University of Florence, University of Perugia, INdAM, CIAFM consortium

DOCTORAL PROGRAMME
IN MATHEMATICS, COMPUTER SCIENCE AND STATISTICS

CURRICULUM IN COMPUTER SCIENCE, MATHEMATICS E STATISTICS
XXXV CYCLE

Administrative Headquarters: University of Florence
Coordinator: Ph.D. Prof. Matteo Focardi

New methods based on Computational Intelligence and Quantum Computing

Academic Discipline (SSD) INF/01

Doctoral candidate
Marco Simonetti

Supervisors:
Ph.D. Prof. Osvaldo Gervasi

Ph.D. Prof. Sumi Helal

Coordinator
Ph.D. Prof. Matteo Focardi

Years 2019/2022

Contents

Introduction . . . **ix**

1 Computing Mega Structures . . . **1**
- 1.1 Introduction . . . 1
- 1.2 High Performance Computing approaches to solving complex problems . . . 3
  - 1.2.1 Cloud containers . . . 3
  - 1.2.2 Containers insights . . . 4
  - 1.2.3 GPGPU Computing . . . 8
  - 1.2.4 GPGPU insights . . . 9
  - 1.2.5 GPGPU and Neural Networks . . . 11
- 1.3 Techniques enabling the solution of complex problems based on Computational Intelligence . . . 12
  - 1.3.1 Approaches based on Machine Learning . . . 13
  - 1.3.2 Machine learnings insights . . . 15
- 1.4 Big Number and AI . . . 17
  - 1.4.1 IoT and AI . . . 19
- 1.5 Conclusion . . . 20

2 Quantum Computing . . . **22**
- 2.1 Introduction . . . 22
- 2.2 Quantum Computing to treat high complex scenarios . . . 24
- 2.3 Quantum mechanical postulates . . . 27
- 2.4 Quantum Computing . . . 29
  - 2.4.1 Bits and Qubits . . . 30
  - 2.4.2 Bloch's Sphere . . . 34

2.4.3 Quantum Transformations and Quantum Gates . . . 40
2.4.4 Quantum Register and Circuits 48
2.4.5 An example of quantum circuit: the *entangler circuit* 53
2.5 Conclusions 58

3 Optical Quantum Computing **60**
3.1 Introduction 60
3.2 Qubits and Qumodes 62
3.2.1 CV Quantum Gates 63
3.2.2 DV Quantum Circuits 66
3.3 The *entangler circuit* in Single Photon Mode 73
3.4 The *entangler circuit* in Dual Rail Mode 78
3.5 Conclusions 87

4 Quantum Computing - Platforms and Algorithms **90**
4.1 Introduction 90
4.2 Quantum Computing insights 93
4.3 Relevant Quantum Algorithms 104
4.3.1 The Deutsch–Jozsa algorithm 105
4.3.2 The Grover's algorithm 105
4.3.3 The Shor's algorithm 109
4.4 Grover's Algorithm 111
4.4.1 Ideal Grover's Circuit implementation 112
4.4.2 Grover's Circuit with IBM Quantum Experience . . . 116
4.4.3 Grover's Circuit with Quandela Cloud 120
4.5 Conclusions 125

5 Quantum Machine Learning **128**
5.1 Introduction 128
5.2 Quantum models for Machine Learning 131
5.2.1 The architecture of the system 132
5.2.2 Data extraction and processing 134
5.2.3 Model construction and validation 134
5.2.4 Analysis of results 137
5.3 Photonic Quantum models for Machine Learning 139

5.3.1 The system architecture 144
5.3.2 Data extraction and processing 146
5.3.3 Models' validation 148
5.4 Conclusion 159

Conclusions **164**

Bibliography **166**

List of Figures

1.1 Final Architecture . . . 7
1.2 CPU vs GPU, performance in GFLOPS . . . 9
1.3 Supervised learning . . . 13
1.4 Ada Structure Complexed With Deoxycoformycin from PDB (left hand side), Orthogonal Axonometry representation (right hand side) . . . 16

2.1 Bit and qubit . . . 31
2.2 Quantum Teleportation . . . 33
2.3 Qubit $|\psi\rangle$ on the Bloch's sphere . . . 34
2.4 The Bloch's sphere, Pauli's axes and their bases for quantum computation. . . . 40
2.5 Identity Gate. . . . 42
2.6 X Gate. . . . 42
2.7 Y Gate. . . . 43
2.8 Z Gate. . . . 43
2.9 H Gate. . . . 44
2.10 Phase Shift Gate. . . . 45
2.11 CNOT Gate. . . . 46
2.12 Rotation of an angle θ around axis: (a) X (b) Y (c) Z . . . 48
2.13 An example of quantum circuit . . . 49
2.14 An example of series circuit . . . 50
2.15 An example of parallel circuit . . . 52
2.16 Entangler Circuit . . . 54
2.17 Circuit for Entangler using Qiskit . . . 55
2.18 Probability Amplitudes of the output state in the Entangler 56

2.19 Technical characteristics for *ibm_manila* system 57
2.20 Probability Distribution of Samples (8,192 shots) on *ibm_manila* system 57

3.1 Phase Shifter gate 69
3.2 Beam Splitter gate 70
3.3 A possible base for block submatrices. 75
3.4 A possible optical implementation for the Entangler 75
3.5 Results obtained by simulating the optical entangler in Perceval 76
3.6 Probability obtained by simulating the optical entangler in Perceval with input state $|1, 0, 0, 0\rangle$ 76
3.7 Optical equivalent gates: H (in pink) and X (in green) 79
3.8 Proposed Optical CNOT schema 80
3.9 Entangler built from the two previous blocks 80
3.10 Simplified proposed Entangler 81
3.11 Output state from simulation on Perceval of Dual-Rail Entangler Circuit 86
3.12 Probability distribution for state $|\psi_{out}\rangle$ on Perceval of Dual-Rail Entangler Circuit 87
3.13 Probability distribution for state $|\psi''_{out}\rangle$ on Perceval of Dual-Rail Entangler Circuit 87

4.1 IBM quantum composer example 95
4.2 StrawberryFields Python's library example 98
4.3 Quantum Inspire composer example 101
4.4 Quirk quantum composer example 102
4.5 Perceval Python's Framework example 103
4.6 Quandela Cloud: various available platforms to run optical quantum algorithms 104
4.7 Quandela Cloud: usage of single platforms 104
4.8 Deutsch-Jozsa quantum circuit to implement the algorithm . 105
4.9 Quantum circuit to implement the Grover's algorithm 109
4.10 Quantum circuit scheme to implement the Grover's algorithm 113
4.11 Oracle 113
4.12 CCZ gate 114

4.13 Diffusion Operator 114
4.14 Quantum circuit scheme to implement the Grover's algorithm 117
4.15 Probability Amplitudes for the output state 118
4.16 Probability Distribution for the output state 118
4.17 Decomposition of the circuit using the base gates for *IBM_manila* platform. 119
4.18 Job info 120
4.19 Probability Distribution for the output state from *IBM_manila* platform. 120
4.20 Grover's algorithm implemented using an Optical Quantum Processor 122
4.21 Probability Amplitudes for the output state in Grover's algorithm, implemented with Perceval 123
4.22 Probability Distribution for the output state in Grover's algorithm, implemented with Perceval 123
4.23 QPU:Ascella characteristics 124
4.24 Job executed on QPU:Ascella 124
4.25 Probability Distribution on QPU:Ascella 125

5.1 Scheme 130
5.2 Architecture of the quantum neural network 133
5.3 Distribution of pattern points on the plane in relation to Gaussian Noise 135
5.4 Quantum Neural Network Models utilised 136
5.5 Metrics for TFQ model 137
5.6 Metrics for PQML model 138
5.7 Metrics for an analogue FFNN 140
5.8 Circuital block diagram for a single layer, single q-mode, related to an optical-quantum layer-based FFNN 142
5.9 Circuital block diagram for a single layer, two q-modes, related to an optical-quantum layer-based FFNN 142
5.10 Circuital block diagram for a single layer, four q-modes, related to an optical-quantum layer-based FFNN 143

5.11 Common structure for the proposed OQFFNN (m is the input dimension) . . . 145
5.12 First type of layer . . . 145
5.13 Second type of layer . . . 146
5.14 Third type of layer . . . 146
5.15 Dataset: Distribution of pattern points on the plane . . . 147
5.16 Dataset: Distribution of pattern points on the plane in polar form . . . 147
5.17 (a) Number of training parameters; (b) Model 1 - layers' scheme . . . 149
5.18 Loss and Validation Loss for Model 1 . . . 149
5.19 Accuracy and Validation Accuracy for Model 1 . . . 150
5.20 (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve . . . 151
5.21 (a) Number of training parameters; (b) Model 2 - layers' scheme 151
5.22 Loss and Validation Loss for Model 2 . . . 152
5.23 Accuracy and Validation Accuracy for Model 2 . . . 152
5.24 (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve . . . 153
5.25 Loss and Validation Loss for Model 2 with phases . . . 154
5.26 Accuracy and Validation Accuracy for Model 2 with phases . 154
5.27 (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve . . . 155
5.28 Number of training parameters . . . 155
5.29 Loss and Validation Loss for Model 3 . . . 156
5.30 Accuracy and Validation Accuracy for Model 3 . . . 156
5.31 (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve . . . 157
5.32 (a) An interesting variant for Model 3; (b) Loss and Validation Loss; (c) Accuracy and Validation Accuracy . . . 159

Introduction

The computing world is rapidly evolving and advancing, with new ground-breaking technologies emerging. This dissertation explores three technologies: Computing Mega Structures, Quantum Computing, and Quantum Machine Learning, placing a strong emphasis on the latter two. These technologies have opened up new possibilities, providing unprecedented computational power and problem-solving capabilities while offering a deeper understanding of complex systems.

Throughout history, humankind has sought to comprehend the world and cosmos by relying on thought patterns and mathematical modelling. However, this perception of a unified certainty began to falter in the early 1800s with the discovery of non-Euclidean geometries and antinomies that originated from the foundations of mathematics. The theory of relativity and quantum mechanics, which emerged in the early 1900s, further debunked the idea of a complete understanding of physical reality, revealing a complexity beyond our wildest imagination. As a result, new technologies and methods are essential for managing the vast amount of information required to make real-time decisions in our society.

This work aims to introduce one such technology: quantum computing, which utilises quantum particles to perform complex calculations in less time than traditional calculators. While the topic is broad, this work unavoidably provides only a partial exposition.

The first chapter discusses current technologies used in massive computing, as well as methods and strategies for managing vast amounts of data.

The second chapter introduces the postulates of quantum mechanics and the necessary physical adaptations to perform calculations with particles.

The third chapter explains the use of photons as special particles in assembling efficient quantum computation systems, highlighting their advantages and disadvantages compared to more mature technologies, such as superconductivity or the ion trap.

The fourth chapter presents various platforms offered by companies working in the sector, along with the design and implementation of valuable algorithms and quantum circuits.

The fifth chapter demonstrates two noteworthy applications of quantum computing in machine learning, one using fermionic particles and the other utilising photons to analyse the feasibility and robustness of a quantum statistical classifier.

It is now necessary to proceed with a more detailed description of what has just been introduced, which will then be outlined and presented in the individual chapters.

Computing Mega Structures is the first topic covered, focusing on remarkable advancements in constructing massive computing infrastructures. These structures employ distributed computing techniques, cloud computing, data centres, supercomputers, and high-performance computing clusters to integrate thousands or millions of computational units seamlessly. They can process vast amounts of data, run complex simulations, and solve intricate problems that were once considered impossible. The chapter has extensively explored computing mega structures, showcasing their potential in solving complex problems. These mega structures offer transformative solutions across various domains by leveraging high-performance computing approaches and computational intelligence techniques. The examination of high-performance computing has revealed the benefits of cloud containers, which provide scalable and flexible computing resources. The insights gained from container utilisation highlight portability, reproducibility, and resource optimisation advantages. Additionally, GPGPU computing has demonstrated the power of parallel processing capabilities, resulting in significant improvements in computational performance. The intersection of GPGPU computing and neural networks has revolutionised artificial intelligence, propelling advancements in computer vision, natural language processing, and data analytics. Furthermore, computational intelligence techniques, particularly those based on machine learning, have proven effective in solving complex problems. Machine learning algorithms, which learn from data, adapt, and evolve, have found diverse applications and provided valuable insights across various domains. The convergence of big data and AI has also been a critical aspect of computing mega-structures. The exponential growth of data necessitates innovative approaches, and AI techniques play a pivotal role in extracting valuable insights. The combination of AI and the Internet of Things (IoT) enables the harnessing of data generated by interconnected devices, leading to advancements in smart cities, healthcare, and manufacturing. Computing mega structures represent a significant advancement in problem-solving capabilities. Integrating high-performance computing approaches, computational intelligence techniques, and the synergy between big data and AI holds immense potential for addressing complex challenges and driving innovation across industries. Further exploration and research in these areas will undoubtedly unlock new frontiers and contribute to a more advanced and interconnected world.

Beyond classical computing, Quantum Computing is explored, which operates on the principles of superposition and entanglement. By utilising qubits, quantum computers can perform calculations at an exponential scale, offering the potential to solve problems currently unsolvable with classical computers. Quantum Computing promises to revolutionise numerous domains, from cryptography and optimisation to drug discovery and material science. Chapter 2 explores the intricacies of quantum computing, its application in treating high-complexity scenarios, and the fundamental concepts that underpin its functionality. At the heart of quantum computing are the postulates of quantum mechanics, which govern the behaviour of quantum systems and differentiate quantum computing from classical computing. Central to quantum computing is the notion of bits and qubits. While classical computing relies on classical bits to encode and manipulate information, quantum computing harnesses the power of qubits. Understanding the properties and characteristics of qubits is crucial for grasping the transformative potential of quantum computing. Bloch's sphere provides a geometric representation to visualise and interpret the states of qubits, facilitating the comprehension of quantum superposition and aiding in elucidating the quantum transformations that transpire during quantum computations. The exploration of Bloch's sphere unveils the intricate nature of quantum states and their relevance in quantum computing. Quantum transformations and quantum gates form the building blocks of quantum computation, enabling the manipulation of qubits and paving the way for complex computations and the development of quantum algorithms. By delving into various quantum transformations and gates, we gain insight into the diverse set of tools at our disposal within the realm of quantum computing. A fundamental component of quantum computing is the quantum register, a collection of qubits operating collectively to perform computations. Quantum circuits, consisting of interconnected quantum gates, govern the flow of information and operations within the quantum register. A comprehensive understanding of quantum registers and circuits is pivotal in harnessing the full potential of quantum computing. Chapter 3 provides a comprehensive exploration of optical quantum computing, covering topics such as optical qubits and qumodes, CV quantum gates, DV quantum circuits, and the entangler circuit in both single photon and dual rail modes. Foundational elements of optical quantum computing are qubits and qumodes. Qubits represent discrete states, while qumodes harness continuous variables. Understanding the interplay between these distinct quantum elements is crucial for developing efficient and scalable optical quantum computing architectures. Specialised quantum gates tailored to manipulate qubits and qumodes are necessary for harnessing the potential of quantum information. In continuous-variable quantum computation, CV quantum gates play a pivotal role. These gates facilitate the manipulation and transformation of quantum states by performing operations such as squeezing, displacement, and rotation, offering a rich repertoire for implementing advanced quantum algorithms and applications. Discrete-variable quantum circuits are also essential in optical quantum computing. These circuits employ a discrete set of operations, including quantum logic gates, to manipulate qubits. DV quantum circuits offer a versatile toolbox for quantum algorithm design by harnessing the principles of superposition and entanglement. Creating entangled states is a central challenge in optical quantum computing, as they underpin entanglement-based operations and quantum communication protocols. This chapter delves into two types of entangler circuits: single photon and dual rail modes. The entangler circuit in single photon mode fosters entanglement generation between qubits or qumodes by harnessing the unique properties of individual photons. Conversely, the entangler circuit in dual rail mode utilises two orthogonal modes to establish entanglement by capitalising on the wave-particle duality of photons. A profound comprehension of these entangler circuits is indispensable for constructing robust and efficient optical quantum computing systems. Chapter 4 begins by exploring the various platforms and technologies that act as the foundation of quantum computing, from superconducting qubits to trapped ions and topological qubits. Each platform has unique strengths and challenges. The chapter examines these platforms' fundamental principles and key components, providing insights into the complex engineering and control required to realise functional quantum computers. After laying this groundwork, the chapter turns its attention to the quantum algorithms that underpin the power and potential of quantum computing. These specialised algorithms, designed to harness the capabilities of quantum systems, unlock the ability to solve problems at an accelerated pace, revolutionising the realm of computation. The chapter explores several pivotal algorithms that have shaped the quantum landscape, including the Deutsch-Jozsa algorithm, Grover's algorithm, and Shor's algorithm. The Deutsch-Jozsa algorithm is an early quantum algorithm that showcases the prowess of quantum computing in solving Boolean function problems with remarkable speedup compared to classical counterparts. Grover's algorithm is a groundbreaking advancement in search algorithms, enabling quick searches within unsorted databases. Shor's algorithm is renowned for its capacity to factor large numbers exponentially faster than classical algorithms and has significant implications for secure communications and encryption protocols. Finally, the chapter revisits Grover's algorithm, implementing two possible circuital solutions.

Accompanying Quantum Computing is the emerging field of Quantum Machine Learning. This fusion of quantum mechanics and machine learning techniques offers the potential to develop quantum algorithms that can learn from data and make predictions. Quantum Machine Learning can enhance the efficiency of classical machine learning algorithms and explore new avenues of pattern recognition, optimisation, and data analysis. By exploring these three topics and examining their underlying principles, cutting-edge applications, and potential implications, it is possible to understand their transformative power comprehensively. Together, they offer a glimpse into a future where computational prowess and intelligent systems converge, propelling humanity toward new frontiers of knowledge, discovery, and innovation. Chapter 5 discusses two specific cases of Quantum Machine Learning, showing how the kernel and supervised training techniques used in machine learning are related to the approaches employed in the quantum variational approach. Both explain how data is handled by explicitly connecting it to vectors in real or complex hyperspaces. The application of this similarity in quantum machine learning is particularly advantageous. In these systems, the data must first be connected to the quantum system's physical states, acting as an analogue adaptation to the nature of the problem. This procedure, which allocates data to quantum states and then trains the system by changing its physical characteristics, is essentially equivalent to one-to-one correspondence. Furthermore, the results obtained using two quantum neural networks will be compared. The first one employs a "classical" quantum approach, using a discrete variable model based on the spin properties of some particles of a fermionic nature. The second utilises photons, and the model will consist of continuous variables: particle position and momentum. A specific dataset (two-moons dataset), simple and effective simultaneously, was used to evaluate the performance of different Machine Learning methods.

During my three-year study, I delved into key topics in information science. I explored and wrote about various subjects like artificial intelligence, cloud computing, telerehabilitation, and quantum computing. Quantum computing has been a particularly enriching area of study for me, as it has allowed me to explore concepts related to mathematics, physics, and information science. This field will continue to grow in importance, attracting more interest as laboratory experimentation progresses and its applications become more widespread. I am grateful to my supervisors for their guidance and to my family, Luisa, Giulia, and Paolo, for their unwavering support throughout this exciting journey.

Chapter 1

Computing Mega Structures

1.1 Introduction

The handling and administration of enormous volumes of data in ever-shrinking time frames have been made possible by the sudden and unrelenting computerisation during the past 40 years. Additionally, the utilisation of computing resources required an exceedingly sophisticated level of elaborateness due to the complexity of scientific and technological concerns [1], as well as the automation of industrial and corporate operations [2]. That necessitated a variety of solutions, including the creation of CPUs, GPUs, TPUs, and other more potent hardware elements, as well as the construction of computer clusters that could tackle the same problem concurrently (see, for example, [3, 4, 5]).

In many application problems in the fields of engineering and logistics, [6, 7, 8, 9], classical analytical methods for the efficient search of the maximum or minimum values of an objective function have been supported by very effective heuristics. The study of algorithms that are less greedy for computational resources can significantly contribute to making information structures more efficient.

In other instances, the issues are intractable despite the use of extraordinary resources due to an intrinsic complexity, such as when factorising exceedingly big numbers or solving a certain class of PDEs (Partial Differential Equations) [10, 11, 12].

Artificial intelligence technologies are essential in fields with high computational complexity, such as chemistry and mathematical-scientific modelling [13]; they are also important in the identification of specific patterns in images or other data aggregates, such as the identification of tumours or the direct identification of emotion from images or photos [14, 15, 16]. The computation framework presupposes an automated, and in some cases, intelligent, adaptation for managing scenarios where the system variables are many and vary quickly [17, 18, 19]. Humankind has recently been able to integrate types of automation and relational empathy with machines at previously imagined levels thanks to the rapid advancement of artificial intelligence technology [20, 21]. There is no doubt that such extensive technological advancement has a significant impact on society: smart cities, which open their doors to extraordinary views and scenarios for the possibilities of services provided to their citizens, to legitimate concerns about their environmental impact; smart schools and educational institutions, which have made significant investments in technology equipment to improve teaching efficacy, whose real consequences and adverse effects are only now becoming clear [22, 23, 24, 25, 26, 27]; smart working is a new way of working that technology has made simpler and more efficient. It has proven effective in some working scenarios. However, its actual value in situations involving a high level of complexity and unpredictable outcomes has yet to be determined [28, 29].

Therefore, one might say that the extraordinarily complex nature of modern society compels us to look for novel answers, even though their important consequences may not be immediately evident. New models should be created to help to understand the ramifications of extremely swift changes, interpret signals, and manage routine and emergency circumstances. Such models need a lot of research, which has to be supported by quick and robust technological support.

1.2 High Performance Computing approaches to solving complex problems

The pandemic is only the most recent and scorching example of a multi-chaotic situation with a devastating impact at planetary level, which in a very short time, thanks to the extraordinary research carried out before and during the period under consideration, by various companies and research groups, has managed to produce different types of vaccines that will hopefully bring us out of the stagnant waters in which we have had to stay for an extended period of time.

This section present a number of technologies that can be fundamental resources for modern and near future research and will enable us to reduce the complexity inherent in various future scenarios.

1.2.1 Cloud containers

Cloud Computing technologies have advanced dramatically in recent years, progressing from the supervision of physical computers to the virtualisation of environments and, eventually, the use of containers, resulting in improved application administration and environment separation.

Containers facilitate cloud infrastructures, and even personal computer resources, to be more flexible by enabling horizontal scalability and, as a result, allowing the infrastructure to be tuned to the application demand. These considerations are critical in order to best calibrate the resources required to address the complexities of the present challenges.

The tested collection of products and technologies may be customised and adapted to the demands of any organisation or corporation that wants to manage data and infrastructure by establishing a private or hybrid Cloud environment (i.e. able to use computing resources from a provider of commercial Cloud services) [30].

In order to showcase the potential of open-source components, the following sections outline the essential procedures involved in establishing a cluster comprising nodes distributed across various company offices. The system can provide seamless scalability and improved resource utilisation using container-based technology, specifically the Docker Engine. The distributed system architecture is designed to provide high availability and fault tolerance by incorporating redundant nodes and load-balancing techniques. Additionally, robust security measures, such as network segmentation, encryption, and access control policies, are implemented to ensure the system's confidentiality, integrity, and availability. The aim is to demonstrate how open-source components can be leveraged to construct a highly dependable and scalable distributed system with robust security features. The system will also have the capacity to be easily extended to commercial Clouds for future growth and expansion. Overall, the sections seek to comprehensively understand how open-source components and container-based technology can be utilised to build a highly dependable and secure distributed system that can quickly scale to meet future business demands.

1.2.2 Containers insights

Data volatility is a significant concern when deploying replicated containers. Containers, in reality, are stateless, and the data contained within them is lost the instant the container is destroyed. The destruction of a container might also occur in an unforeseen manner; for example, it is conceivable that autoscaling identifies low system usage and, as a consequence, decreases the number of active nodes. In order to tackle this challenge, it is critical to provide a network-distributed file system that is subsequently used as storage by containers. That ensures the permanence and consistency of information. XFS[1] is one of the file systems that may be utilised, and its maintenance may be delegated to the GlusterFS program. GlusterFS also allows provisioning a list of dedicated servers. These will create a trustworthy pool, which will be used to share available disk space across nodes. Persistent data will uniformly be mounted on each node. An enterprise-grade fibre connection between sites with a speed of at least 1 Gbps will be required.

[1]XFS (Extents File System) is a 64-bit, high-performance journaling file system for Linux. It was initially created by Silicon Graphics for its IRIX OS, but the code was later donated to Linux. XFS works exceptionally well with large files and is known for its robustness and speed. XFS supports filenames of up to 255 bytes, files of up to 8 EB and file systems of up to 16 EB.

If the services require a database, optimising it will be the next step. The most successful method entails the establishment of several containers to which specific tasks are assigned. One will be the DB container, with the role of master, while the others will be slaves. The master is used for writing and is the only one that can change the structure and data in the database. Instead, slaves will be containers with only the necessary rights to perform reading queries.

The significant advantage is that one can add as many slave nodes as needed, improving our ability to satisfy customer demands linearly. In general, individuals who explore a website perform far more reading actions than writing actions.

With this perspective, one can also create a database autoscaling mechanism that raises the number of reader nodes based on various factors or metrics, such as the average database usage percentage or the average number of active requests per second. There are two critical points to consider.

First, there is some delay across the slave nodes (readers) and the master (writer) in this design. It means that there may be instances where readings on newly written data provide old and out-of-date results. When a database is configured correctly, latency is very low, often less than 100ms and mostly closer to 30ms or 40ms.

The second thing to remember is that queries are not automatically sorted across master and slave nodes. Two methods exist to distribute the workload among the writer and readers. The first is to act on the program code that utilises the database, which requires creating two access points to the database and sorting queries that only perform reading operations on the access point to the slaves and queries that only execute writing operations on the access point to the master. However, this solution is only sometimes feasible; it is perfectly plausible that the program used is legacy or closed source.

The second method, which arises as an alternative to this problem, requires the definition of an additional container that acts as a load balancer and automatically sorts requests across nodes. In the case of a MariaDB database, for instance, a Maxscale proxy can be used. In this regard, it is allowed to publish the MariaDB MaxScale REST API port, an HTTP interface that generates data in JSON format, offering visual management tools. Access to the proxy implies adding a new rule to the NAT Network port forwarding table. MariaDB MaxScale divides requests such that writing queries are sent to the master container while reading queries are handled and balanced through the $slave1, slave2, \ldots, slaveN$ containers. It is also feasible to include the master in the pool of nodes capable of reading. Then, in the master, a special user with `GRANT ALL` access must be defined for MaxScale. MariaDB MaxScale is released under a BSL (Business Source License) license and is capable of much more than simply load balancing: it can also handle failover and switchover. A few stages are involved in the configuration: establishing the servers, creating the monitoring service, defining traffic routing using the Read-Write-Split mode, and lastly, configuring the listener agents using the *mariadbclient* protocol and its TCP port.

The other container to analyse is the proxy for web services. It exposes port 80 for the HTTP protocol and port 443 for the HTTPS protocol. Its task will be to receive and sort requests from the network and redirect them to the proper web servers (such as Apache, NodeJS or Nginx). To appropriately configure the proxy, the documentation given by the HAProxy official site is consulted - The Reliable, High-Performance TCP/HTTP Load Balancer[2], where it is explained the use of the software in a Docker Swarm environment. The DNS resolver definitions are very critical elements to consider. These are required for identifying the containers that comprise the backend. Furthermore, proper request timeout settings must be defined and calculated to reduce the chance that some blocked processes will bind connections, causing difficulties that render services inaccessible to users. The HTTP mode setting enables HAproxy to make sophisticated decisions, such as routing traffic within a particular group of servers depending on the URL route requested by clients and routing traffic depending on the HTTP headers received. This parameter allows HAproxy to work as a level 7 load balancer. This mode is essential for load balancing.

It is also necessary to specify the relevant certificates for using the HTTPS protocol. Alternatively, one can rely on other containers that deal with the automatic generation of certificates, such as those developed by

[2]See the URL: `https://hub.docker.com/_/haproxy`

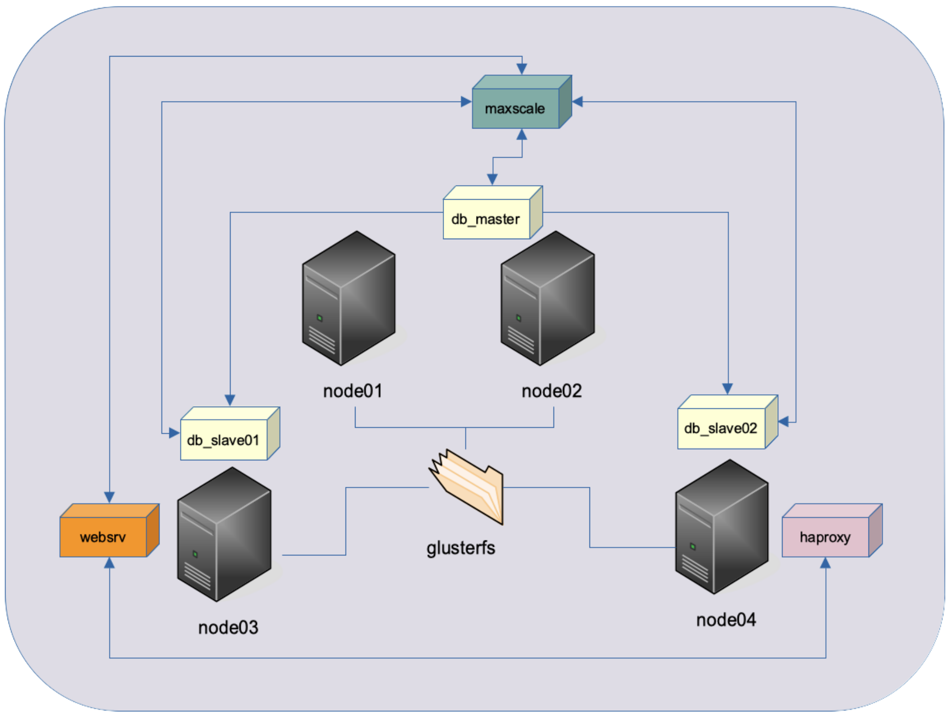


Figure 1.1: Final Architecture

EFF Let's Encrypt [31]. In the backend unit, it is established the algorithm for balancing requests to Web servers. There are several algorithms; the most common is *Round Robin*, which selects the recipient servers one by one in a cyclic way.

Alternatively, the *LeastConn* algorithm may be used: it chooses the Web server that will handle the request, depending on the number of still active connections and always selects the Web server with the minimum load. With this final container, the entire infrastructure may be completed.

The completed infrastructure is depicted in Figure 1.1. One can check the status and distribution of all containers built using the container orchestration tool in swarm mode, Docker service, which is only executable by the management node. Finally, it is vital to note that Portainer[3] may be used to manage and administrate all of the containers that have been built.

[3]Portainer is a universal container management tool. It works with Kubernetes, Docker, Docker Swarm and Azure ACI. It enables the management of containers without needing to know platform-specific code. See the URL https://www.portainer.io/.

It has a web interface that makes long-term system maintenance easier and access to logs, status graphs, and tools for restarting and restoring individual containers.

1.2.3 GPGPU Computing

GPGPU is an abbreviation that stands for General-Purpose Computing on Graphics Processing Units. This term refers to the usage of graphics accelerators, which are essential components of computer graphics, for generalised mathematical operations and calculations that result in the acceleration of computations. With the advent of customizable shaders and support for floating point computations in 2001, GPGPU computing grew in popularity. The capacity to carry easily out SIMD (Single Instruction on Multiple Data) calculations, is one of the features of these architectures.

GPGPU computing became readily helpful even in the consumer environment in 2006, owing to the launch by NVIDIA of the 8800 series of graphics cards with G80 processor, which could benefit from CUDA[4], an architecture capable of executing highly efficient programs for parallel computing on GPU. In 2006, an 8800GTX with the G80 featured 128 unified shaders (compute units) working at a frequency of 576 MHz, which guaranteed 345.6 GFLOPS. Today, 15 years later, the top-of-the-line card named RTX 3090 for desktop PCs created by nVidia delivers 8704 compute units at 1440Mhz that promise 35581 GFLOPS: 102.95 times larger than what was supplied by the finest video card in 2006.

The graph in Figure 1.2 depicts the rise in the computing capability of CPUs and GPUs over time. On the Y axis, the number of billions of floating point operations per second (GFLOPS) is reported. The performance disparity is increasing yearly, as seen by the logarithmic scale. GPUs become programmable architectures, similar to CPUs, thanks to both proprietary (CUDA) and Open Frameworks (OpenCL[5] and SYCL[6]), and can operate

[4]CUDA is a proprietary framework released by NVIDIA™

[5]OpenCL™ (Open Computing Language) is an open, royalty-free standard for cross-platform, heterogeneous, parallel programming devices (CPUs, GPUs, FPGAs, DPSs

[6]SYCL is a royalty-free, cross-platform abstraction layer that enables code for heterogeneous processors to be written using standard ISO C++ with the host and kernel code

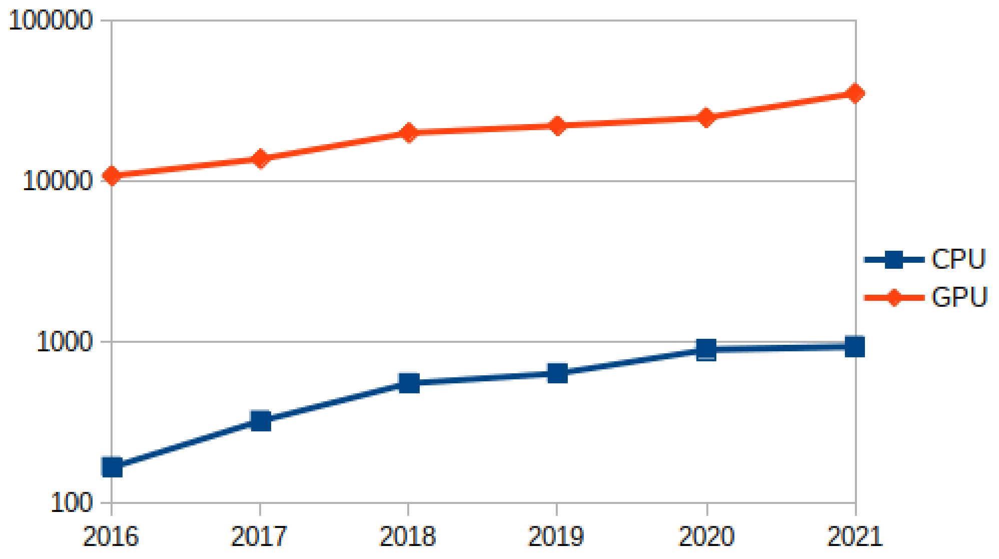


Figure 1.2: CPU vs GPU, performance in GFLOPS

hundreds of threads at the same time. More appropriately, since we are considering Compute Units, we can name this approach Single Instruction Multiple Thread (a name NVIDIA gave to this architecture, SIMT).

CUDA and both OpenCL and SYCL have a strong relationship. The latter is the Open Source alternative to CUDA and enables building parallel, particularly for a diverse collection of GPUs, including Intel, NVIDIA, and AMD. However, the performance is different, and OpenCL is often slower than CUDA because of the feature that allows it to operate on many types of hardware. In 2011, the performance gap was around 16% [32, 33], and to date, it appears to have gotten even more pronounced [34].

1.2.4 GPGPU insights

As mentioned in previous sections, GPGPU computing can significantly speed up many algorithms. Examples include cryptography, image and sound manipulation and analysis, multimedia coding and decoding, neural networks, and Natural Language Processing applications that are crucial in many areas today. However, using these techniques to represent the state

for an application contained in the same source file.

of the art of research requires adopting specific guidelines to produce the expected results.

The architecture of the GPGPU involves running a host program on the CPU and a kernel program on the innumerable Compute Units of the GPU. The host program has the function of allocating the necessary resources, monitoring the execution status of the kernels and collecting the final results. Each Kernel will execute the required algorithm on a Compute Unit, acting on a portion of the input data. The input data must be moved from the central memory to the GPU memory before Kernels are executed. At the end of the kernel computation, the output results have to be copied from the GPU memory to the central memory. This data movement must be optimised in the sense that has to be minimised the input-output requests outside the GPU memory. Furthermore, it is of fundamental importance to allocate data in such a way as to make maximum use of the memory areas close to the Compute Units, in line with the Memory model implemented by OpenCL. The data transfer has a computational cost in terms of time and machine cycles and can have a heavy impact on the measured performance. We can conclude that using these technologies is optimal in the case of medium to extensive input data [35].

Although using these graphics accelerators results in a considerable increase in power, the performance-per-watt ratio[7] is in favour of GPUs over CPUs [36]. For example, the Intel i9-11900K processor that hit the market in Q1 2021 has a Thermal Design Power (TDP[8]) of 125 Watts and is capable of delivering 996.0 GFLOPS in SGEMM calculations. The NVIDIA 3090 has a TDP of 350 Watts and is capable of delivering 35581 GFLOPS. When calculating the performance-per-watt ratio, it is got 996/125=7.968 GFLOPS/watt for the CPU and 35581/350=101.66 GFLOPS/watt for the GPU.

[7]It evaluates the amount of calculation a computer can perform for every watt of power consumed.

[8]The maximum amount of heat created by a computer chip or component is known as the thermal design power (TDP), which is also known as the thermal design point.

1.2.5 GPGPU and Neural Networks

The field of machine learning has witnessed a boom of interest in recent years since several studies have demonstrated the efficacy of neural networks in various tasks that were previously thought to be extremely difficult. Experts have access to massive volumes of data, and modern accelerators have enormous computing capability [37].

Developing code that solves a problem and makes it worthwhile on many graphics card architectures, on the other hand, has become incredibly challenging, especially from an algorithmic standpoint. Indeed, while a GPU gets programmed, it must declare how memory and registers are to be used and allocated in great detail. Even graphics cards of the same brand and generation (but different models) might have significant variances. Auto-tuning approaches have reduced the difficulty of performance portability by customising memory structures and loops to a specific architecture. Today, some auto-tuning software and libraries have been developed to cater to researchers' powerful instruments and attempt to mitigate this problem. Various approaches can allow to optimise kernel parameters for various GPU architectures, i.e. using a deterministic approach and the "Generate and Test" technique. In more recent times, through machine learning, neural networks have been created that can predict the best parameters to be used in a GPU code and optimise final performance.

GPGPU computing is currently used to accelerate many of the most common deep learning frameworks, such as TensorFlow, PyTorch, Caffe, Matlab, and others. It is also used in linear algebra, Data analysis, cryptography, affective computing and so on [38]. Some specific neural networks work with images, and convolution is one of the most fundamental operations. Assume we have a picture saved on our device. In most cases, a picture is represented using three colour channels: red, green, and blue (RGB), which are represented in memory with three distinct matrices with the same dimensions. The number of rows will be equal to the height of the image, while the number of columns will be equal to the image's width. Each matrix element will contain the integer that quantifies the amount of Red, Green, Blu, respectively. Because most images use 8 bits to encode each pixel's brightness value (grey tone), these values are usually between 0 and 255. This results in 16 million colours being represented as results combining 255 possible red tones, 255 possible blue tones, and 255 possible green tones.

Convolutional Neural Networks are a type of Deep Neural Network that performs well on data organised in a grid topology, such as time series and visual inputs and are one of the most popular machine learning architectures for image-based classification. Image recognition, segmentation, detection, and retrieval are examples of applications where CNNs have achieved state-of-the-art results. Convolutional effects can be varied, such as enhancing edges, increasing contrast, dilating or eroding the area occupied by the objects in the image, and so on. Winograd, Image to Column + GEMM, and Direct Convolution are the three main algorithmic techniques for performing convolution with digital images. These three algorithms obtain the same output but have different execution times. Arrays of different sizes and parameters may be faster with the Winograd convolution than with the Direct convolution and vice versa. We demonstrated that it is possible to identify the best algorithm for convolutional operations based on the previously described arrays [39].

1.3 Techniques enabling the solution of complex problems based on Computational Intelligence

Solving complex problems has always been challenging for organisations and businesses in every sector, solving complex problems has always been a challenge. However, with the increased processing power of computers and advancements in Artificial Intelligence techniques, Computational Intelligence techniques, including Machine Learning, can now be used to tackle these challenges effectively and efficiently. Machine Learning, one of the most advanced AI techniques, is revolutionising the resolution of complex problems in various sectors by learning from data. Machine Learning can provide even more advanced and sophisticated solutions when combined with other Computational Intelligence techniques, such as artificial neural networks, evolutionary optimisation, and fuzzy logic. This paragraph explores how Machine Learning and other Computational Intelligence techniques can face complex problems and their advantages over traditional methods.

1.3.1 Approaches based on Machine Learning

There are different types of Machine Learning methods, one of which is Supervised Machine Learning. This method involves creating a dataset of items to train the classifier, focusing on accurately classifying them into different categories. The dataset is then split into two parts: a training set and a test set. The training set is used to teach the classifier, while the test set is used to verify the results. The goal is for the classifier to recognise input items accurately. After the training is completed, the classifier's performance is evaluated by presenting images that were not included in the training dataset. An example is shown in Figure 1.3.

Some of the most commonly used machine learning techniques are as follows.

Decision trees

Decision trees (DT) are a non-parametric, supervised machine learning method for classification and regression. DTs generate *white box* models that are easily interpretable as *if-then-else* expressions. Minor data modifications, on the other hand, might result in the generation of wholly distinct trees.

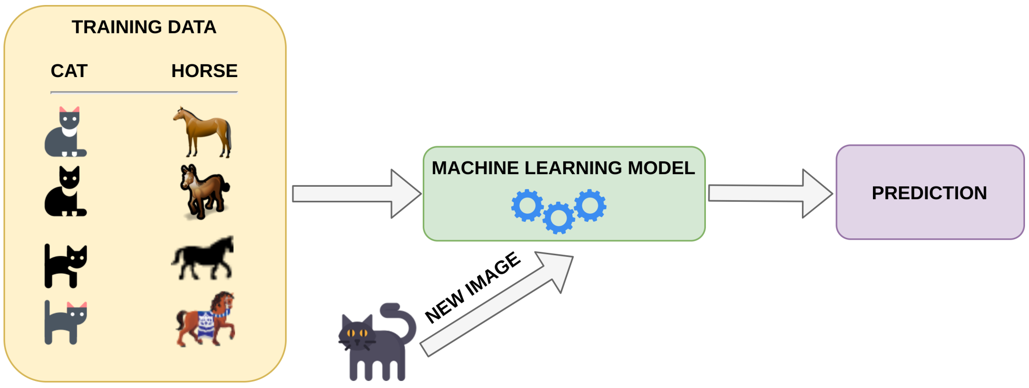


Figure 1.3: Supervised learning

Random Forest

Random Forest fits many independent trees on various subsets of the dataset to reduce variance and over-fitting. Gradient Tree Boosting is another notable tree-based classifier that uses gradient descent to grow a weighted ensemble of DTs gradually.

Bayesian classifier

The simplest variant of a Bayesian network is the naive Bayesian classifier (NBC). An NBC assumes that all features are conditionally independent given the class variable. While this assumption is frequently incorrect, it has proven to be highly effective in practice, obtaining good, simple models with little training.

Logistic Regression

Logistic Regression (LoR) is a simple type of regression analysis in which the classes' independent variables and log odds are assumed to have a linear relationship.

K-Nearest Neighbours

K-Nearest Neighbours(KNN) finds the K data points closest to the query feature vector and polls their assigned labels to determine the query vector's label. KNN is a non-generalising method; it means that instead of learning a model's parameters, it "remembers" the training points and eventually stores them in a suitable data structure, such as a Ball Tree (i.e. metric tree), to speed up inference.

Support Vector Machine

Another machine learning technique is the Support Vector Machine. The data is separated using a support vector machine (SVM) to draw hyperplanes in the feature space. In order to maximise the separation between classes, the points closest to the hyper-planes are used as a reference. The *kernel trick*, which entails defining alternative inner products for the data points, can implicitly embed the problem in higher-dimensional spaces, allowing for complex separating surfaces.

Multi Layer Perceptron

The simplest feed-forward neural network is the Multi-Layer Perceptron (MLP). There are at least three node layers: an input layer, a hidden layer, and an output layer. Each node, except for the input nodes, uses a nonlinear activation function.

Convolutional neural network

CNNs are a type of deep, feed-forward artificial neural network that is commonly used for image recognition. They comprise a series of layers, each of which applies a set of learned convolution filters to the previous layer's results (*activations*). The final layer is usually a classifier with a loss function to back-propagate gradients through the network and update the filter values (*weights*). Non-linearity is typically added after each convolution layer and other layers like pooling operators and fully-connected layers. CNNs convert an input image's original pixel values into final class scores.

1.3.2 Machine learnings insights

Galileo Galilei (1564-1642) introduced the scientific method, characterising and shaping science for centuries to come. The scientific method is based on the repeatability of experiments, which ensures that researchers and scientists can verify the validity of results produced by others. Machine learning, and in particular neural network learning, on the other hand, produces different results each time learning is carried out. This is due to the fact that the weights of the neurons that make up the neural network and that are to be trained are instantiated at random values and the search for values that optimise the objective function can lead to different search paths at each iteration [40]. Because of these characteristics, machine learning makes it possible to address issues, studies and research in a completely new way. In recent times, a research group has initiated a study aimed at achieving protein recognition through the application of machine learning techniques [13, 41]. The study commenced with a method that involved protein classification, which was accomplished by analysing the list of proteinogenic amino acids that constitute them. The analysis was conducted using a one-dimensional convolutional neural network that was able to differentiate between genuine and fake proteins. Afterwards, the researchers endeavoured to enhance the approach by analysing the three-dimensional structure of the proteinogenic amino acid chain, which involved incorporating a list of X, Y, and Z coordinates of various elements in the analysis. The resulting analysis produced a four-dimensional matrix since each element comprised three spatial coordinates and the amino acid type, encoded as an integer. However, the researchers encountered a challenge when trying to test the quality of the analysis using a convolutional neural network. Most neural networks employed in the literature accept three-dimensional matrices as input. Therefore, a method was developed to transform the 4D-encoded proteins into 3D-encoded proteins. A graphical representation based on orthogonal axonometry was employed to accomplish this.

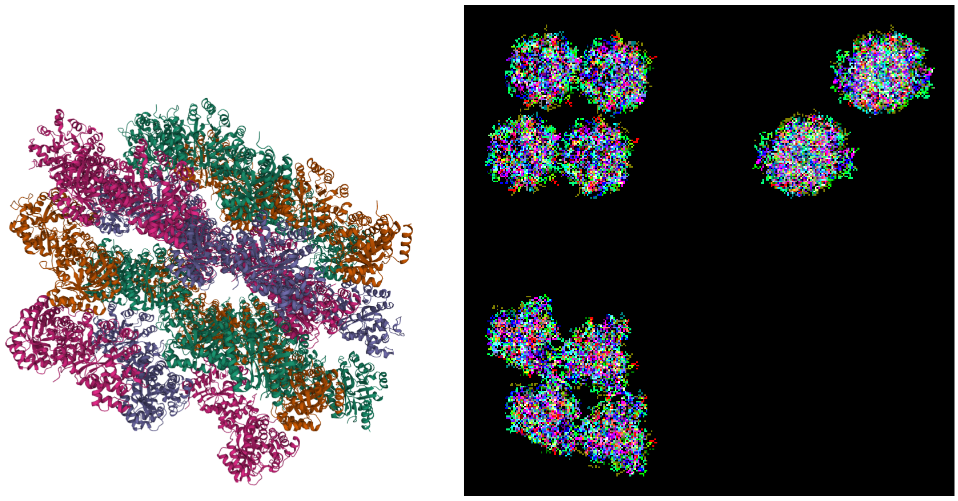


Figure 1.4: Ada Structure Complexed With Deoxycoformycin from PDB (left hand side), Orthogonal Axonometry representation (right hand side)

The image was initially empty and subsequently partitioned into four segments. The proteins were then inserted into the segments, and the color of each pixel was encoded using the type of amino acid, while the XY position was encoded using a transformation function that employed photographs of the protein from three different perspectives. Consequently, each image contained a protein, which was represented using cavalier axonometry. Convolutional neural networks were then employed to analyze the images, and a higher performance was achieved compared to previous work. Figure 1.4 presents an example, which involves the 3D representation of the *Ada Structure Complexed With Deoxycoformycin* provided by the Protein Data Bank (PDB) on the left-hand side and our representation using Orthogonal Axonometry on the right-hand side.

The programming of applications that make use of machine learning is greatly facilitated by a number of frameworks that allow developers to implement high-level algorithms that benefit the computational capabilities of machine learning approaches and that use a number of already optimised libraries. In Table 1.1 are summarised the main Frameworks available.

Framework	**Main characteristics**	**Released by**
Keras	Open Source (MIT License), written in Python	François Chollet
Scikit-learn	Open Source (BSD-3-Clause License), written in Python, Cython, C and C++	David Cournapeau
Tensorflow	Open Source (Apache2 License), written in C++ and CUDA	Google Brain Team
pyTorch	Open Source (BSD License), written in Python, C++ and CUDA	Facebook's AI Research lab (FAIR)
Microsoft Cognitive Toolkit	Open Source (MIT License), written in C++	Microsoft Research
H2O	Open Source (Apache2 License), written in Java	H20.ai

Table 1.1: Main frameworks for machine learning

1.4 Big Number and AI

Computing large numbers has always been a challenge for scientists in all fields. In many fields, such as physics, astronomy, chemistry and engineering, calculations of large numbers are essential for research and development.

However, the complex nature of these calculations often requires a great deal of time, resources and specialised technical knowledge. Fortunately, Artificial Intelligence (AI) is revolutionising the resolution of these complex problems.

Thanks to its ability to learn from data and adapt to situations autonomously, AI can automate many processes to calculate large numbers. For example, AI can be used for data classification, prediction modelling, and simulation of complex scenarios. Furthermore, AI can be used to optimise algorithms and parallelise computations, making it possible to solve problems that would otherwise be impossible or impractical.

One of the most notable examples of AI's application to large numbers computations is using artificial neural networks (ANNs) in prediction modelling. ANNs are machine learning algorithms that are capable of learning from data in a similar way to the human brain. When applied to forecasting modelling, ANNs can quickly and efficiently analyse large amounts of data and identify complex patterns quickly and efficiently. That means that ANNs can be used to predict future events, such as weather, market trends, or disease risk.

In addition, AI can be used for scientific data analysis, such as analysing experimental data, anomaly detection, and data visualisation. This type of data analysis can help scientists identify new relationships in data, reveal hidden patterns, and uncover new insights. For example, AI can be used for analysing data from environmental sensors to identify factors influencing climate change or for analysing genomic data to identify factors causing certain diseases.

Furthermore, AI can be used for the simulation of complex scenarios, such as the simulation of physical phenomena or the simulation of industrial processes. This type of simulation can help scientists evaluate the effectiveness of new theories or technologies, predict the effects of environmental or policy changes, and identify areas for improvement in industrial processes. For example, AI can simulate the propagation of sound waves in air or water to help design better noise control devices. Furthermore, AI can be used to simulate the diffusion of chemicals in the air or water to identify possible environmental effects.

Finally, AI can be used for algorithm optimisation and computational parallelisation. That means that calculations of large numbers can be broken into smaller pieces that can be performed simultaneously on multiple processors or computers, significantly reducing the time required to solve them. Algorithmic optimisation and computational parallelisation can be used in many fields, such as pharmaceutical research, financial analysis, and electronic device design.

In summary, AI is revolutionising the solving of complex problems based on large numbers in multiple scientific and industrial fields. AI can model predictions, analyse scientific data, simulate complex scenarios and optimise algorithms, and parallelise computations. This fact has led to a considerable improvement in the speed and efficiency of calculating large numbers, paving the way for new scientific and technological breakthroughs. However, it is also important to note that AI cannot wholly replace human experts in many fields but rather work in synergy with them to achieve better and more accurate results.

1.4.1 IoT and AI

The Internet of Things (IoT) and Artificial Intelligence (AI) are two technologies revolutionising how people interact with the world around them. In particular, the IoT allows the connection of everyday objects to the network, while AI allows the analysis of the collected data to obtain valuable information. When these two technologies are combined, the result obtained is a highly sophisticated home monitoring system that offers numerous benefits.

One of the main benefits of using IoT and AI for home monitoring is the ability to detect and prevent potential security issues. For example, IoT sensors can detect the presence of smoke or toxic gases, while AI can be used to analyse the collected data and determine if it is a real threat. Furthermore, IoT and AI-based security systems can detect any intrusion into the home and notify the proper authorities in an emergency.

Another benefit of using IoT and AI to monitor home environments is the ability to monitor energy consumption. Sensors can detect the presence of people inside the house and automatically adjust the temperature and lighting according to individual needs. Furthermore, IoT and AI-based monitoring systems can be used to monitor the energy consumption of individual electronic appliances, allowing the house's occupants to identify any waste and take measures to reduce consumption.

However, there are also some issues to consider when using IoT and AI for home monitoring. For example, privacy can be an issue, as sensors collect data about the activities of the home's occupants. It is essential to ensure that data is protected and only used for its intended purpose. Additionally, there can be compatibility issues between various IoT devices, which can make it challenging to integrate sensors with other systems.

1.5 Conclusion

Computing's mega structures have become a crucial tool for solving complex problems across a wide range of industries. High-performance computing approaches have enabled the processing of massive amounts of data, simulation of complex systems, and optimisation of intricate processes that would otherwise be impossible with traditional computing methods. The development of computing structures has come a long way since the first supercomputer was built in the 1960s. Today, high-performance computing systems are faster, more powerful, and more accessible than ever. Advances in hardware and software technologies have paved the way for new approaches to solving complex problems, such as artificial intelligence, machine learning, and deep learning. The use of high-performance computing approaches has significantly impacted various fields, including scientific research, healthcare, energy, finance, and manufacturing. For example, in scientific research, HPC systems have enabled simulations of complex systems, such as weather patterns, protein folding, and astrophysical events, which have led to groundbreaking discoveries and advancements in various fields. In healthcare, HPC systems have facilitated the analysis of large datasets from clinical trials and genomics research, leading to personalised medicine and better patient outcomes. In energy, HPC systems have enabled the optimisation of energy production, storage, and distribution, leading to more efficient and sustainable systems. In finance, HPC systems have facilitated the analysis of market trends and risk management, leading to more informed investment decisions. In manufacturing, HPC systems have enabled the simulation and optimisation of production processes, improving quality and efficiency. Despite the numerous benefits of high-performance computing approaches, challenges still need to be addressed. One of the biggest challenges is the cost of building and maintaining HPC systems. The high costs of hardware, software, and energy consumption can make it difficult for smaller organisations to invest in HPC infrastructure. Additionally, the complexity of HPC systems and the need for specialised expertise can make it challenging for organisations to implement and maintain HPC solutions. Another challenge is the need for scalable and efficient algorithms to take advantage of HPC systems' parallel processing capabilities. Developing efficient algorithms that can handle massive datasets and complex systems is an ongoing research area that requires collaboration between computer scientists, mathematicians, and domain experts. In conclusion, high-performance computing approaches have revolutionised how we solve complex problems. As technology advances, it can be expected to see even more innovative and impactful uses of computing mega structures in the future.

Chapter 2

Quantum Computing

2.1 Introduction

The idea of using quantum mechanics in the world of computation was born around the 1950s following the statement by R. Feynman about the possibility of simulating nature efficiently and effectively. Some years later, he described [42] the possibility of defining physical laws through a computer, by analyzing their probabilistic aspect: the important conclusion was that classical computation could not effectively simulate physical processes, that would only be reachable through a quantum computer.

In the late 1970s classical probabilistic computation became extremely important in computer science and as a result the first non-deterministic algorithms were implemented, giving rise to doubts about Church-Turing's thesis [43]. This was followed by the studies of David Deutsch who in his *"Quantum theory, the Church–Turing principle and the universal quantum computer"* [44] lays the foundations to define how to apply quantum principles to the Turing machine. He first talked about a quantum Turing Machine, that is, an abstract model that allowed to simulate a quantum computer.

In 1982, physicist Paul Benioff was able to demonstrate that the classical Turing machine could simulate certain physical phenomena without incurring an exponential slowdown in its performance. Three years later David Deutsch hypothesized that, since the laws of physics were ultimately approximated to those of quantum mechanics, a device based on the principles of quantum mechanics could be used to efficiently simulate an arbitrary physical system. The true potential of this new science was highlighted in the early 1990s by Richard Jozsa, who in 1991, after describing the functions that cannot be solved by quantum parallelism, collaborated with Deutsch proposing the first problem that a quantum machine could solve more quickly than a deterministic one.

Later, in 1994, a mathematician named Peter Shor developed an algorithm that could find the factors of large numbers much more efficiently than the best classical algorithm. It was so powerful that it put modern cryptography at risk: in fact, encryption algorithms that until then (and even in the present day) were considered state-of-the-art, with the advent of this new type of computation would have soon become obsolete. It is thought that, given an integer N, Shor's algorithm can factorize it in a time of $O(log(N))$, while on a classical computer the time is exponential in N. This means that the quantum algorithm could easily pierce the best modern encryption algorithm, in a very short time.

Another notable algorithm was thought by the researcher Lov Grover, who was able to solve a search problem in a database of N unsorted items in $O(\sqrt{N})$ using $O(logN)$ as storage space. This is a real incredible result compared to classical search algorithm that operates in $O(N)$.

These advances contributed to the current definition of quantum computers, which are computers that use quantum physics and mechanics to provide computational power much superior to that of a classical computer for some types of problems.

The study of these machines has given rise to a new field of theoretical research in computer science and physics called *quantum computation*, which will completely overturn the processing of information, managing to solve currently unsolvable scientific problems. Despite the fact that the premises are very promising, the actual technical implementation for quantum computers has not helped to achieve the desired results yet; in fact, managing a large number of qubits, which is required to solve problems of varying nature and structure, has demonstrated a significant level of implementation complexity. In addition, the realization of a quantum algorithm necessitates an ad-hoc configuration of the algorithm itself, due to the peculiarity of quantum transformations compared to classical ones, and the implementation of the corresponding quantum circuit. Nonetheless, the time and resources required to design and construct such systems will be amply repaid with the ability to solve computationally very complex problems, which are currently difficult, if not impossible, to solve with existing hardware architectures.

2.2 Quantum Computing to treat high complex scenarios

The actual quantum race, which spans the years 2000 to the present, starts in 2001 when IBM completed the most difficult quantum calculation. They built a seven-qubit quantum computer using billions of molecules, solving a short variant of the mathematical conundrum that underlies many modern cryptographic data security solutions.

Moreover, in 2016, IBM made the first quantum computer accessible via the IBM cloud. From then on, the world's largest IT companies started to confront, culminating in 2019 with Google's assertion of quantum supremacy [45]. They conducted a sampling experiment known as a random quantum circuit and published the results in Nature. Their Sycamore 53 qubits-quantum computer managed to provide a solution in around 3 minutes, a billion times faster than the 10,000 years that a traditional supercomputer would need.

Despite Google's landmark achievement, the result just obtained by the Hefei USTC (University of Science and Technology of China) team is very interesting [46]. The photon system developed would be able to perform the calculation ten trillion times faster than the most powerful Chinese supercomputer. The operation in practice refers to Galton's machine, where there are photons instead of beads. At the same time, instead of the pegs, prisms and mirrors divert the particles' path. This system of mirrors and prisms is called an interferometer. In the case of Galton's machine, it is sufficient to count the beads that fall into each column at the base. However, when using photons, it is necessary to use devices capable of detecting the arrival of light particles. Based on Gaussian Boson Sampling's concepts, this new experiment obtained incredible results, demonstrating how a calculation performed with this system of interferometers was enormously more efficient than the one performed on a supercomputer. These results conclude that this approach can take a considerable step towards a new way of conceiving computation.

Although it will be some time before there is a commercially viable quantum system, several business companies, which are currently developing quantum processors, are working together to assess the potential uses of quantum in daily life.

A quantum computer would be able to manage a large quantity of data, making it highly beneficial for logistics and transportation and for all those tasks that call for a significant database. Quantum algorithms manage to be significantly more efficient than classical ones. Then, given its capacity for problem-solving, it will undoubtedly be employed in both the financial and health sectors to advance artificial intelligence.

Quantum computing offers the potential to solve problems exponentially faster than classical computers, but its development and implementation come with special and distinctive challenges in processing information.

1. *Scalability Factor*:
    - *Physical Scalability*: Quantum computers differ from classical computers as they use qubits instead of bits to process information. Creating stable qubits that can maintain their quantum state (coherence) for a sufficient amount of time is challenging. As we increase the number of qubits in a system, ensuring they all interact correctly and maintain coherence becomes progressively more complicated.
    - *Error Correction*: Quantum computers are highly susceptible to errors due to their delicate nature. To scale up, we need quantum error correction techniques to correct these mistakes. Implementing these techniques requires additional qubits, making large-scale, fault-tolerant quantum computing a significant challenge.
2. *Energy Factors*:

- *Cooling Needs*: Superconducting qubits, a popular approach to quantum computing, require temperatures near absolute zero to function correctly. This cooling is energy-intensive and becomes more challenging as systems grow.
- *Power Consumption*: As quantum systems scale, their power consumption could become significant, especially when considering error correction and other overheads.

3. *Dealing with Security Threats (Quantum-Safe Cryptography)*:

    - *Quantum Supremacy and Encryption*: The most immediate security threat is the potential for quantum computers to break widely used cryptographic schemes. For instance, RSA and ECC, popular public-key cryptographic methods, can be compromised by a sufficiently powerful quantum computer using Shor's algorithm.
    - *Transition to Quantum-Safe Algorithms*: While quantum computers threaten existing cryptographic techniques, they also pave the way for new quantum-safe algorithms. Transitioning to these new methods will require global cooperation, rigorous testing, and significant changes to existing IT infrastructures.
    - *Quantum Key Distribution (QKD)*: QKD offers provably secure encryption based on the principles of quantum mechanics. However, it currently has limitations in terms of distance and requires specialized infrastructure, such as quantum repeaters, to make it viable for broader applications.

4. *Other Challenges*:

    - *Software and Algorithms*: Quantum computing requires a new programming paradigm. Developing software that can harness the potential of quantum hardware is still in its early stages.
    - *Interfacing with Classical Systems*: Quantum computers will likely work in tandem with classical systems for many applications. Efficient communication and data transfer between the two is crucial.

While quantum computing has immense potential, it presents unique challenges requiring interdisciplinary expertise and cooperation. As technology advances, addressing these challenges will be critical to realizing its promise and mitigating its risks.

Therefore, the quantum will undoubtedly be a subject of discussion for years to come and a driver of investments and innovations.

2.3 Quantum mechanical postulates

Scientists have derived some assumptions that lead to what is known as the postulates[1] of quantum mechanics in order to comprehend quantum mechanics more thoroughly. These are, in fact, the presumptions we must make to comprehend the relationship between the mathematics of quantum physics and measurable reality.

1. **Postulate 1 - The Wave Function Postulate**: The state of a quantum mechanical system is completely specified by a function $\Psi(\mathbf{r}, t)$, called *the wave function* or *state function*, that depends on the coordinates of the particle(s) and on time.

   As a single particle has a probability equal to 1 of being discovered somewhere in space, the normalisation condition must be applied

$$\int_{-\infty}^{\infty} \Psi^*(\mathbf{r}, t)\Psi(\mathbf{r}, t) d\tau = 1 \tag{2.1}$$

2. **Postulate 2 - Experimental Observables**: a linear, Hermitian operator in quantum mechanics corresponds to each observable in classical mechanics. If we demand that the expected value of an operator $\hat{A}$ be real, as it should be, then this postulate is obligatory.

3. **Postulate 3 - Individual Measurements**: the eigenvalues that match the eigenvalue equation (Eq.2.2) are the only values that may

[1] https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/The_Live_Textbook_of_Physical_Chemistry_(Peverati)/23%3A_Postulates_of_Quantum_Mechanics

ever be detected in any measurements of the observable related to operator A:

$$\hat{\mathcal{A}}\,|\Psi\rangle = \alpha\,|\Psi\rangle \tag{2.2}$$

This postulate expresses the essential tenet of quantum mechanics: dynamical variable values can be quantised. However, it is possible to have a continuum of eigenvalues, as in the case of unbound states in photonics. Every measurement of the quantity $\hat{\mathcal{A}}$ will always result in α if the system is in an eigenstate of $\hat{\mathcal{A}}$ with eigenvalue α. Even though measurements must always return an eigenvalue, the starting state need not be an eigenstate of $\hat{\mathcal{A}}$.

4. **Postulate 4 - Expectation Values and Collapse of the Wavefunction**: the average value of the observable corresponding to $\hat{\mathcal{A}}$ in a system characterised by a normalised wave function is given by:

$$\langle A\rangle = \int_{-\infty}^{\infty} \Psi^* \hat{A} \Psi d\tau \tag{2.3}$$

The wave function instantaneously *collapses* into the associated eigenstate Ψ_i following measurement of returns some eigenvalue α_i. This is an essential implication of the fourth postulate. Measurement, thus, has an impact on the system's state.

5. **Postulate 5 - Time Evolution**: The wave function of a system evolves in time according to the time-dependent Schrödinger equation:

$$\hat{H}\Psi(\mathbf{r},t) = i\hbar\frac{\partial\Psi}{\partial t} \tag{2.4}$$

6. **Postulate 6 - Pauli Exclusion Principle**: The total wave function of a system with n *spin* $-\frac{1}{2}$ particles (also called *fermions*) must be antisymmetric concerning the interchange of all coordinates of one particle with another. For *spin-1* particles (also called *bosons*), the wave function is symmetric:

$$\begin{aligned} \Psi\left(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_n\right) &= -\Psi\left(\mathbf{r}_2,\mathbf{r}_1,\ldots,\mathbf{r}_n\right) \quad \textit{fermions} \\ \Psi\left(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_n\right) &= +\Psi\left(\mathbf{r}_2,\mathbf{r}_1,\ldots,\mathbf{r}_n\right) \quad \textit{bosons} \end{aligned} \tag{2.5}$$

To describe the properties of the qubit, we must necessarily refer to a reformulated subset of the postulates of quantum mechanics.

1. A suitable Hilbert space $\mathcal{H}$ is associated to every physical system. The state of this system is described by a vector $|\psi\rangle \in \mathcal{H} \mid \langle\psi|\psi\rangle = 1$ (unitary norm). If two physical systems 1 and 2, with Hilbert spaces $\mathcal{H}$ and $\mathcal{H}$ are combined in a unique system (for example by making them interact in some way), the space of Hilbert of the composite system is given by the tensor product of $\mathcal{H}_1$ by $\mathcal{H}_2$ ($\mathcal{H}_1 \otimes \mathcal{H}_2$).

2. Each observable quantity **A** is associated with a linear and self-adjoint operator $\hat{\mathcal{A}}$ in space. The set of possible values for measuring a quantity is given by the spectrum of the operator associated with it.

3. If the physical system is in a state $|\psi\rangle$ the probability that the observation of a quantity **A** gives as result α is directly proportional to $|\langle\alpha_i|\psi\rangle|^2$.

4. The measure of the observable **A** on the state $|\psi\rangle$, assuming we got α as a result, projects $|\psi\rangle$ on the eigenspace of α.

2.4 Quantum Computing

Quantum computing is a field of computing that uses quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data [47, 48]. Unlike classical computers, which use bits that can only be in one of two states (either 0 or 1), quantum computers use *quantum bits*, or *qubits* (rarely also written "qbits"), that can exist in multiple states simultaneously, allowing them to perform calculations much faster than classical computers. A qubit is a two-level quantum system that can be in a state of 0, 1, or a superposition of both states. In other words, a qubit can be considered a unit of information in a quantum computer. Unlike classical bits, which are either 0 or 1, qubits can exist in a linear combination of the two states, known as a superposition. In addition to superposition, qubits also have the ability to be entangled with one another. Entanglement is a quantum phenomenon in which the states of two or more qubits are correlated in a way that classical physics cannot explain. That allows quantum computers to perform certain operations faster than classical computers, such as factorisation and searching large databases. While quantum computing has the potential to revolutionise computing as known so far, it is still in its infancy. It faces many challenges, such as decoherence (loss of quantum information due to interaction with the environment), error correction, and scalability. Nevertheless, researchers continue to progress in this exciting field, and quantum computers are already used for specific tasks in industry and research.

2.4.1 Bits and Qubits

The bit is the smallest unit of information; it is also the foundational element of Claude Shannon's classical theory of information, and it is physically equivalent to a two-state system with only two possible values: 0 and 1. Instead, the quantum computer employs qubits (quantum bits), a physical property of an elementary particle that obeys quantum physics by simultaneously existing in two states: it can be both 1 and 0 at the same time, or even in a superposition of them, that is a linear combination of them. A classical system could be compared to a region in real three-dimensional space made up of two distinct points, whereas a qubit could be compared to one of the infinite points on the surface of an unitary sphere (Figure 2.1). This quantum physical device was defined by Benjamin Schumacher [49] as being simpler, more flexible, and powerful than the digital one. To comprehend the essence of this new kind of machine and how it differs from its traditional counterpart, the bit, we must consider the rules that govern the actions and development of a particular physical structure, and therefore the transmission of the knowledge found within it. It has become possible to extend the field of operation of a machine by looking at it from a different point of view, thanks to some quantum mechanics postulates. As a result, the space filled by classic binary sequences (bit registers) will contain all infinite variations (*principle of superposition of states*) as well as their non-classical relations (*phenomenon of interference*), and all of these will affect the final outcome (*principle of measurement*).

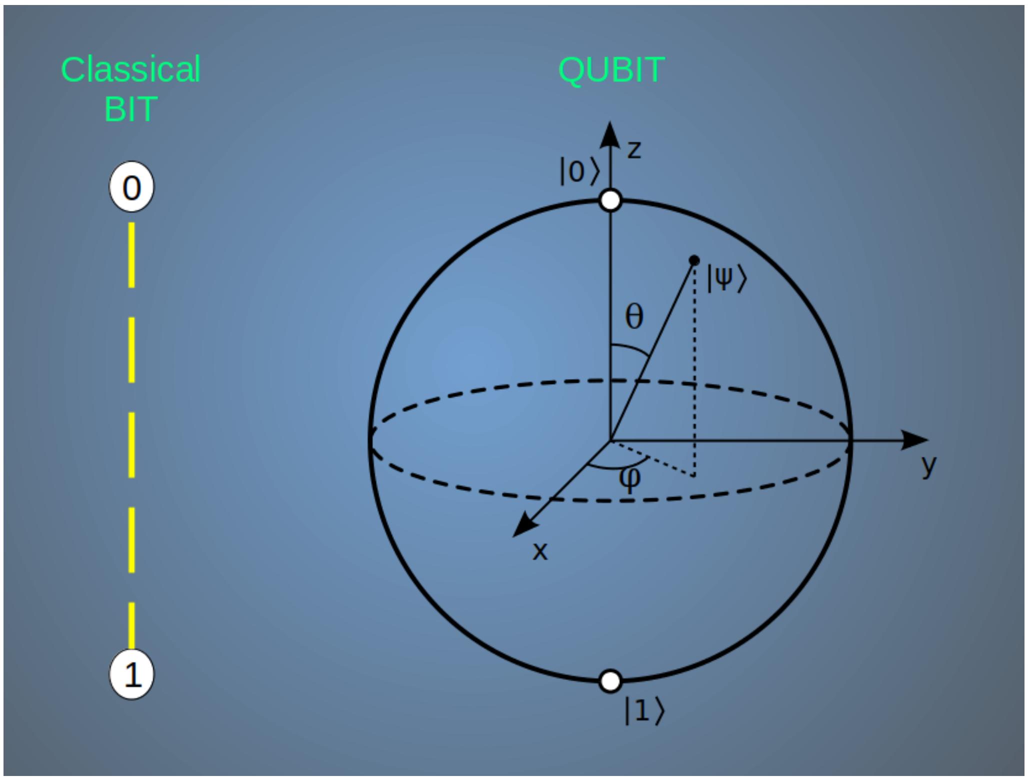


Figure 2.1: Bit and qubit

Always referring to quantum physics we can identify the qubit as a vector ψ in the two-dimensional complex space $\mathbb{C}^2$ (Hilbert space) defined in the form

$$\begin{gathered}|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{pmatrix}0\\1\end{pmatrix} + \beta\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix}, \\ \textit{with } (\alpha,\beta) \in \mathbb{C}^2 \wedge |\alpha|^2 + |\beta|^2 = 1\end{gathered} \tag{2.6}$$

where the scalars α and β are complex numbers expressing the probability amplitude of the state $|0\rangle$ and $|1\rangle$, respectively; $|0\rangle$ and $|1\rangle$ states constitute the computational basis for the space under consideration. The probability that a qubit can be 0 or 1 is totally without guarantees and is dependent only on situations of uncertainty. These intermediate states of the qubit are considered *superpositions of the states* and can be viewed as a coexistence of them in certain proportions. The probability that a quantum state $|\psi\rangle$ can be detected in state $|\varphi\rangle$ when it is measured in relation to a basis that also contains $|\varphi\rangle$ is equal to

$$P_{|\varphi\rangle}(\psi) = |\langle\varphi|\psi\rangle|^2 \tag{2.7}$$

The individual probabilities, with respect to the computational basis, for the state $|\psi\rangle$ can be calculated as in the equations Eq.2.8.

$$\begin{aligned} P_{|0\rangle}(\psi) &= |\langle 0|\psi\rangle|^2 = |\alpha\langle 0|0\rangle + \beta\langle 0|1\rangle|^2 = |\alpha|^2 \\ P_{|1\rangle}(\psi) &= |\langle 1|\psi\rangle|^2 = |\alpha\langle 1|0\rangle + \beta\langle 1|1\rangle|^2 = |\beta|^2 \end{aligned} \tag{2.8}$$

Projective measurements are the name for this kind of measurement. For *pure states* [50, 51], to satisfy the normality condition, the components of the state $|\psi\rangle$ must meet the equation Eq.2.9.

$$\begin{aligned} 1 = ||\psi\rangle|^2 = \langle\psi|\psi\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)\cdot(\alpha|0\rangle + \beta|1\rangle) = \\ = \alpha^*\alpha\langle 0|0\rangle + \alpha^*\beta\langle 0|1\rangle + \beta^*\alpha\langle 1|0\rangle + \beta^*\beta\langle 1|1\rangle = |\alpha|^2 + |\beta|^2 \end{aligned} \tag{2.9}$$

In order to create a computational model, which allows the automatic execution of the calculation, it is necessary to be able to carry out some elementary transformations on a single qubit or groups of qubits. These transformations respond to precise temporal evolutions of the wave function and can be mathematically represented through particular operators with their specific algebra or, equivalently, by unitary matrices. From a functional point of view, these transformations are called *Quantum Gates* (section 2.4.3).

Comprehending quantum processes applies the notion of *entanglement*, which refers to the correlation between two or more quantum systems. This correlation exists even when the sub-systems are not communicating directly or indirectly and are related in a cause-and-effect relationship. An essential and direct concept correlated is *quantum teleportation*, which involves the instantaneous transmission of a quantum state from one stage to another regardless of the distance between the particles. See Figure 2.2 and Algorithm 1 for more information. It is necessary to produce an entangled quantum state or a Bell's state (the four states that can be created when two qubits are maximally entangled - Eq.2.10) for the qubit to be transmitted so that

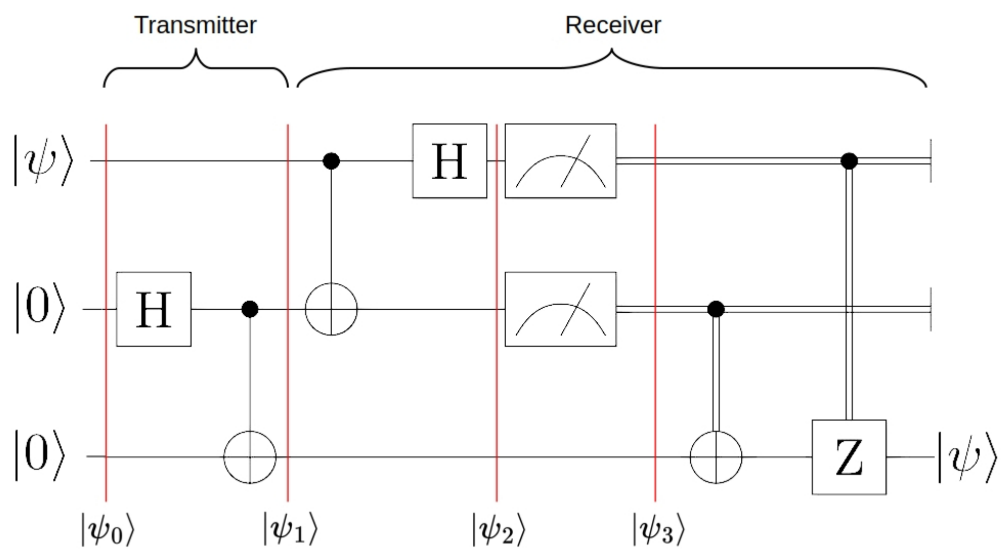


Figure 2.2: Quantum Teleportation

effective teleportation can be performed.

$$
\begin{aligned}
\left|\Phi^+\right\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \\
\left|\Phi^-\right\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}} \\
\left|\Psi^+\right\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}} \\
\left|\Psi^-\right\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}}
\end{aligned}
\tag{2.10}
$$

In order to manipulate the entangled quantum state, the sender then prepares the particle with information in the qubit and joins it with one of the entangled particles in the intermediate state. The entangled particle's modified state is subsequently transferred to an instrument that tracks the alteration. Information can be *teleported* or transmitted between two persons in separate places if the change in measurement enables the destination to reproduce the original information the sender possessed. The no-cloning theorem[2] [52] is preserved as the information is regenerated from the entangled

[2]The *no-cloning theorem* states that it is impossible to implement a circuit that will perfectly copy an unknown quantum state

state and not copied during teleportation. That happens since the original quantum information is definitively lost when it becomes a component of the entanglement state.

2.4.2 Bloch's Sphere

Equation 2.6 emphasises that a qubit may be represented as a vector of $\mathbb{C}^2$ and, more specifically, as a linear combination of two computational basis eigenstates. The system of equations characterising the state is lowered by one degree by the constraint that the state's norm must be equal to 1, decreasing the overall system's degrees of freedom to three. That allows visualising a qubit in three dimensions using a proper geometrical representation called Bloch's sphere (Fig.2.3).

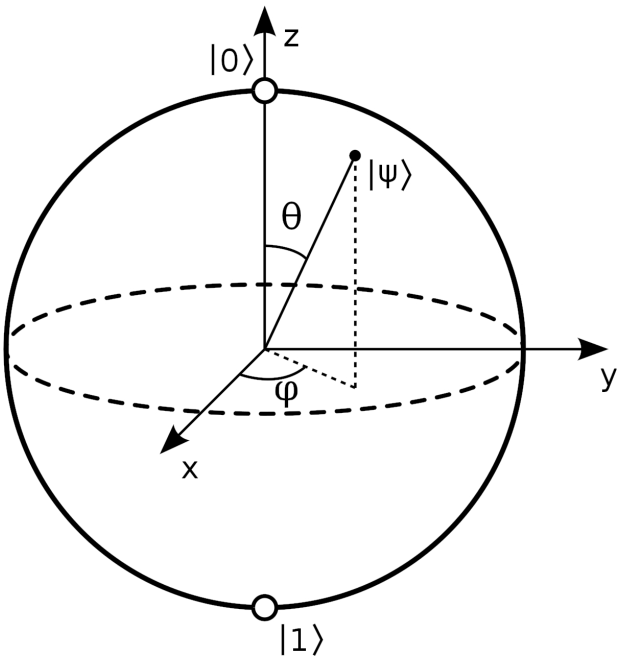


Figure 2.3: Qubit $|\psi\rangle$ on the Bloch's sphere

So, two antipodal points on this unit 2-sphere type correspond to a pair of mutually orthogonal state vectors. North and South Poles in the Bloch's sphere are commonly set to match the conventional basis vectors $|0\rangle$ and $|1\rangle$, which may represent an electron's spin-up and spin-down states correspondingly. Other representations of the qubit are possible, all equivalent. Starting

Algorithm 1 Quantum Teleportation - Part 1

```
1: In Reference to Fig.2.2, the initial state $|\psi_0\rangle$ is created from three distinct
   particles. The first from the top (least significant qubit - LSQB) is in
   the generic $\alpha|0\rangle + \beta|1\rangle$ state, while the others are in the zero-state.
   The LSQB, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, contains the information to be sent.
   Mathematically:
   $$|\psi_0\rangle = |0\rangle \otimes |0\rangle \otimes |\psi\rangle = |0\rangle \otimes |0\rangle \otimes (\alpha|0\rangle + \beta|1\rangle) = \alpha|000\rangle + \beta|001\rangle$$
2: The subcircuit characterising stage 2 of the transmission circuit is the
   **entangler**, capable of creating a Bell state, starting from two distinct
   ground-states $|0\rangle$. In this specific case, the Bell's state generated is
   $$|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$
   So, $|\psi_1\rangle = |\Phi^+\rangle \otimes |\psi\rangle$.
   Now the two most significant qubits are *entangled* and can be
   individually distributed to the sender and receiver. The initial state $|\psi_0\rangle$
   will then be transformed into $|\psi_1\rangle$.
   $$\begin{aligned} |\psi_1\rangle &= \Big(\Big(\mathbb{I} \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1|\Big) \otimes \mathbb{I}\Big) \cdot \Big(\mathbb{I} \otimes \mathbb{H} \otimes \mathbb{I}\Big) \cdot |\psi_0\rangle = \\ &= \frac{\sqrt{2}}{2}\alpha|000\rangle + \frac{\sqrt{2}}{2}\beta|001\rangle + \frac{\sqrt{2}}{2}\alpha|110\rangle + \frac{\sqrt{2}}{2}\beta|111\rangle \end{aligned}$$
3: Once the state $|\psi_1\rangle$ has reached the receiver, it is converted to the state
   $|\psi_2\rangle$ employing a disentangling circuit
   $$\begin{aligned} |\psi_2\rangle &= \Big(\mathbb{I} \otimes \mathbb{I} \otimes \mathbb{H}\Big) \cdot \Big(\mathbb{I} \otimes \Big(\mathbb{I} \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1|\Big)\Big) \cdot |\psi_1\rangle = \\ &= \frac{\alpha}{2}|000\rangle + \frac{\alpha}{2}|001\rangle + \frac{\beta}{2}|010\rangle - \frac{\beta}{2}|011\rangle + \\ &+ \frac{\beta}{2}|100\rangle - \frac{\beta}{2}|101\rangle + \frac{\alpha}{2}|110\rangle + \frac{\alpha}{2}|111\rangle \end{aligned}$$
```

```
Algorithm 1 Quantum Teleportation - Part 2
4: Four possible results are possible after measuring the two least significant qubits
     1. measurements outcomes are equal to 00, then the state $|\psi_3\rangle$ is
        $$|\psi_3\rangle = \frac{\alpha}{2}|000\rangle + \frac{\beta}{2}|100\rangle$$
     2. measurements outcomes are equal to 01, then the state $|\psi_3\rangle$ is
        $$|\psi_3\rangle = \frac{\alpha}{2}|001\rangle - \frac{\beta}{2}|101\rangle$$
     3. measurements outcomes are equal to 10, then the state $|\psi_3\rangle$ is
        $$|\psi_3\rangle = \frac{\beta}{2}|010\rangle + \frac{\alpha}{2}|110\rangle$$
     4. measurements outcomes are equal to 11, then the state $|\psi_3\rangle$ is
        $$|\psi_3\rangle = -\frac{\beta}{2}|011\rangle + \frac{\alpha}{2}|111\rangle$$
5: At this point, it will be sufficient to use two quantum gates controlled by the collapsed qubits, such as a cX and a cZ, to obtain always the original state $|\psi\rangle$ on the most significant qubit.
```

from Eq.2.6, it is also possible to describe the qubit in Hilbert Space as follows mathematically:

1. Let

$$\alpha = Ae^{i\varphi_1} \wedge \beta = Be^{i\varphi_2}$$
$$\textit{with} \;\; A \in \mathbb{R}_0^+ \wedge B \in \mathbb{R}_0^+$$
$$\textit{with} \;\; \varphi_1 \in \; [0, 2\pi) \subset \mathbb{R} \wedge \varphi_2 \in \; [0, 2\pi) \subset \mathbb{R}$$

2. From constraints in Eq.2.9, it is obtained

$$|\alpha|^2 + |\beta|^2 = 1 \Rightarrow \left|Ae^{i\varphi_1}\right|^2 + \left|Be^{i\varphi_2}\right|^2 = 1 \Longrightarrow A^2 + B^2 = 1$$

3. From constraints in 1. and 2., it is possible to write

$$A = \cos\frac{\theta}{2} \wedge B = \sin\frac{\theta}{2}\,, \qquad \textit{with}\ \ \theta \in\ [0,\pi] \subset \mathbb{R}$$

4. Constraints in 1. , 2. and 3. implies

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = e^{i\varphi_1}\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi_2}\sin\left(\frac{\theta}{2}\right)|1\rangle =$$
$$= e^{i\varphi_1}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi_2}\sin\left(\frac{\theta}{2}\right)|1\rangle\right)$$

5. From a purely physical point of view, the multiplicative factor $e^{i\varphi_1}$ (called *global phase*) is not measurable and therefore

$$|\psi_1\rangle = e^{i\varphi_1}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right)$$

   and
$$|\psi_2\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$
   are two indistinguishable states (with $\varphi = \varphi_2 - \varphi_1$).

6. This new mathematical description (Eq.2.11) presents two freedom degrees and explicitly highlights that a single qubit can be imagined as a three-dimensional vector of unit norm. These angles can be interpreted geometrically as Euler angles in the unit sphere (Fig.2.3). Each point on the sphere surface can be reached through the following definition:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.11}$$

   with $\theta \in [0,\pi] \subset \mathbb{R}$ and $\varphi \in [0,2\pi] \subset \mathbb{R}$.

An equivalent mathematical form to effectively represent a qubit is given by its density matrix. Assuming that this matrix (Eq.2.12) can represent any

2x2 unitary transformation:

$$U = \begin{pmatrix} a & b \\ -e^{i\omega}b^* & e^{i\omega}a^* \end{pmatrix} \tag{2.12}$$

with $a \in \mathbb{C} \wedge b \in \mathbb{C} \wedge \omega \in [0; 2\pi) \subset \mathbb{R} \wedge |a|^2 + |b|^2 = 1$. The outer product $|\psi\rangle\langle\psi|$ gives the density matrix (Eq.2.13) ρ of the qubit $|\psi\rangle$ (Eq.2.11).

$$\begin{aligned} \rho = |\psi\rangle\langle\psi| &= \cos^2\frac{\theta}{2}|0\rangle\langle 0| + e^{-i\varphi}\cos\frac{\theta}{2}\sin\frac{\theta}{2}|0\rangle\langle 1| + \\ &+ e^{i\varphi}\cos\frac{\theta}{2}\sin\frac{\theta}{2}|1\rangle\langle 0| + \sin^2\frac{\theta}{2}|1\rangle\langle 1| = \\ &= \begin{pmatrix} \cos^2\frac{\theta}{2} & e^{-i\varphi}\cos\frac{\theta}{2}\sin\frac{\theta}{2} \\ e^{i\varphi}\cos\frac{\theta}{2}\sin\frac{\theta}{2} & \sin^2\frac{\theta}{2} \end{pmatrix} = \\ &= \frac{1}{2}\begin{pmatrix} 1+\cos\theta & e^{-i\varphi}\sin\theta \\ e^{i\varphi}\sin\theta & 1-\cos\theta \end{pmatrix} = \\ &= \frac{1}{2}\begin{pmatrix} 1+\cos\theta & \cos\varphi\sin\theta - i\sin\varphi\sin\theta \\ \cos\varphi\sin\theta + i\sin\varphi\sin\theta & 1-\cos\theta \end{pmatrix} \end{aligned} \tag{2.13}$$

Moreover, Equations 2.13 and 2.14 connect several equivalent qubits' descriptions: the density matrix, Pauli's Spinors and Bloch's vector $\vec{r} = (r_x, r_y, r_z) \in \mathbb{R}^3$.

$$\begin{aligned} &= \frac{1}{2}\begin{pmatrix} 1+r_z & r_x - ir_y \\ r_x + ir_y & 1-r_z \end{pmatrix} = \\ &= \frac{1}{2}\left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\cdot r_x + \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\cdot r_y + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\cdot r_z\right] = \\ &= \frac{1}{2}\Big[\mathbb{I} + \sigma_x \cdot r_x + \sigma_y \cdot r_y + \sigma_z \cdot r_z\Big] = \frac{1}{2}\Big[\mathbb{I} + \vec{\sigma}\cdot\vec{r}\Big] \end{aligned} \tag{2.14}$$

where $(r_x, r_y, r_z) = (\cos\theta, \cos\varphi\sin\theta, \cos\varphi\sin\theta)$ is the mathematical characterisation of a point on a unit-sphere.

In quantum mechanics and group theory, the matrices $\mathbb{I}$ (Identity) and

σ_x, σ_y, σ_z (Pauli's matrices) constitute the so-called Pauli's Group G_1

$$G_1 \overset{\text{def}}{=} \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} \equiv \langle X, Y, Z \rangle \tag{2.15}$$

where

$$\begin{array}{ll} \mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, & X = \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ Y = \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, & Z = \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{array} \tag{2.16}$$

The eigenvectors (eigenstates) derived from the individual Pauli's matrices form a tern of orthonormal bases helpful in describing the qubit's state. The preferred basis for the computation is given by the eigenstates of the Z-matrix, which are positioned as two antipodal points of the Bloch's sphere and on its Z-axis (Fig.2.4); they are labelled $|0\rangle$ (North Pole) and $|1\rangle$ (South Pole) and named *Computational Basis.*

Alternative bases are the *Hadamard Basis* on the X-axis and the *Circular Basis* on the Y-axis (Eq.2.17 and Table 2.1).

$$\begin{array}{l} \{|+\rangle\,, |-\rangle\} = \left\{ \frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle\,, \frac{\sqrt{2}}{2}|0\rangle - \frac{\sqrt{2}}{2}|1\rangle \right\} \\ \{|\circlearrowright\rangle\,, |\circlearrowleft\rangle\} = \left\{ \frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}i|1\rangle\,, \frac{\sqrt{2}}{2}|0\rangle - \frac{\sqrt{2}}{2}i|1\rangle \right\} \end{array} \tag{2.17}$$

Basis	Name	Generating Axis
$|0\rangle\,, |1\rangle$	Computational	Z
$|+\rangle\,, |-\rangle$	Hadamard	X
$|\circlearrowright\rangle\,, |\circlearrowleft\rangle$	Circular	Y

Table 2.1: Basis vectors and their generating axes on the Bloch's sphere

Any point on the Bloch's sphere can be reached, starting from state $|0\rangle$ (North Pole), through the composition of three rotations around two of the three axes X, Y, Z. The rotation matrices [53, 54] around the axes are

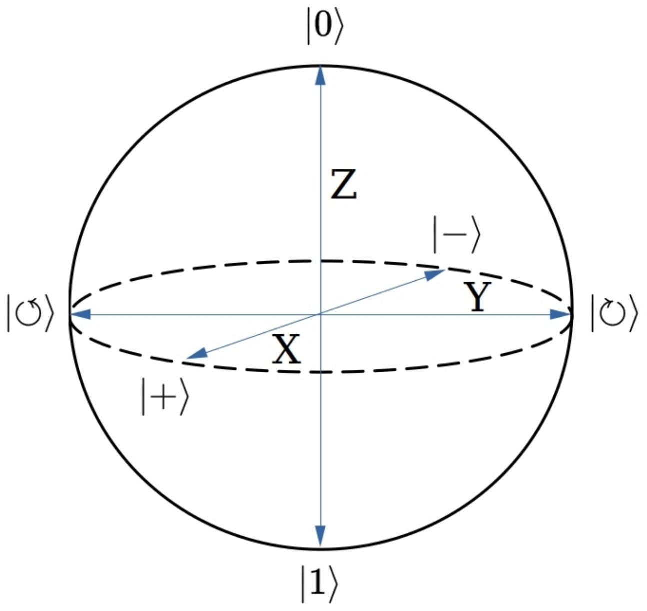


Figure 2.4: The Bloch's sphere, Pauli's axes and their bases for quantum computation.

respectively given in Eqs.2.18, 2.19, 2.20.

$$R_X(\theta) = e^{-i\frac{\theta}{2}\cdot X} = \exp\left(-i\frac{\theta}{2}\cdot\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\right) = \begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right)\\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix} \tag{2.18}$$

$$R_Y(\theta) = e^{-i\frac{\theta}{2}\cdot Y} = \exp\left(-i\frac{\theta}{2}\cdot\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\right) = \begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right)\\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix} \tag{2.19}$$

$$R_Z(\varphi) = e^{-i\frac{\varphi}{2}\cdot Z} = \exp\left(-i\frac{\varphi}{2}\cdot\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\right) = \begin{pmatrix}e^{-i\frac{\varphi}{2}} & 0\\ 0 & e^{i\frac{\varphi}{2}}\end{pmatrix} \tag{2.20}$$

with θ and φ like in Eq.2.11. More precisely, in minimal form, a series of gates R_Z, R_X, R_Z, or equivalently R_Z, R_Y, R_Z, are sufficient to mathematically describe each quantum state on the Bloch sphere, except for a global phase.

2.4.3 Quantum Transformations and Quantum Gates

A rudimentary quantum circuit using a few qubits is known as a quantum logic gate (or just a quantum gate). Classical logic gates are the building blocks of traditional digital circuits, so they are the building blocks of quantum circuits [55, 56, 57]. Taking as reference a basis of the Hilbert space, almost always the computational basis, quantum gates are unitary operators and are expressed as unitary matrices. Eq.2.12 can be equivalently rewritten in the following forms:

$$\begin{aligned}
U &= e^{i\frac{\varphi}{2}} \begin{pmatrix} e^{i\varphi_1}\cos\theta & e^{i\varphi_2}\sin\theta \\ -e^{-i\varphi_2}\sin\theta & e^{-i\varphi_1}\cos\theta \end{pmatrix} \\
U &= e^{i\frac{\varphi}{2}} \begin{pmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} e^{i\Delta} & 0 \\ 0 & e^{-i\Delta} \end{pmatrix} \\
U &= \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} e^{i\xi} & 0 \\ 0 & e^{i\zeta} \end{pmatrix} \cdot \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix}
\end{aligned} \tag{2.21}$$

Apart from the real values to be assigned to the various parameters in the individual equations in Eq.2.21, it is interesting to note that a unitary matrix can be decomposed, or factorised, into simpler unitary matrices, which depend singularly on one parameter. That suggests the actual possibility of realising any quantum transformation on a qubit through the use and composition of a set of elementary transformations.

The following proposes a collection of the most representative and primarily used quantum gates in quantum computation.

Identity Gate

The Identity gate does not perform any changes on the qubit state. Its only purpose is to mathematically express the results of different gate operations, especially in multi-qubit circuits. In practical use, it can act as a delay stage, aiding in the synchronisation of quantum transformations that take place in both parallel and series.

Its graphical representation is given in Fig.2.5, while its mathematical descriptions are in Eq.2.22 (expressed in Dirac's notation and matrix notation, respectively).

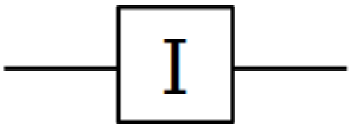


Figure 2.5: Identity Gate.

$$\begin{aligned}\mathbb{I} &= |0\rangle\langle 0| + |1\rangle\langle 1| \\ \mathbb{I} &= \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\end{aligned} \tag{2.22}$$

X Gate

Gate X performs an exchange operation of the components of the state vector, i.e. it transforms $|0\rangle$ into $|1\rangle$ and vice versa, or $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in $\beta|0\rangle + \alpha|1\rangle$ (Eq.2.23).

$$\begin{aligned} X|0\rangle &= \Big(|0\rangle\langle 1| + |1\rangle\langle 0|\Big)|0\rangle = |0\rangle\langle 1|0\rangle + |1\rangle\langle 0|0\rangle = |1\rangle \\ X|1\rangle &= \Big(|0\rangle\langle 1| + |1\rangle\langle 0|\Big)|1\rangle = |0\rangle\langle 1|1\rangle + |1\rangle\langle 0|1\rangle = |0\rangle \\ X|\psi\rangle &= X\Big(\alpha|0\rangle + \beta|1\rangle\Big) = \alpha X|0\rangle + \beta X|1\rangle = \beta|0\rangle + \alpha|1\rangle \end{aligned} \tag{2.23}$$

Its graphical representation is given in Fig.2.6, while its mathematical descriptions are in Eq.2.24.

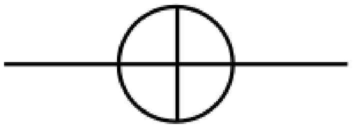


Figure 2.6: X Gate.

$$\begin{aligned} X &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ X &= \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\end{aligned} \tag{2.24}$$

Y Gate

Gate Y performs a swap operation of the components of the state vector and displaces state component $|1\rangle$ of π, unless a global phase of $\frac{\pi}{2}$, i.e. it transforms $|0\rangle$ into $i|1\rangle$ and $|1\rangle$ into $-i|0\rangle$, or $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in $i\beta|0\rangle - i\alpha|1\rangle$ (Eq.2.25).

$$\begin{aligned} Y|0\rangle &= i\Big(|1\rangle\langle 0| - |0\rangle\langle 1|\Big)|0\rangle = i|1\rangle\langle 0|0\rangle - i|0\rangle\langle 1|0\rangle = i|1\rangle \\ Y|1\rangle &= i\Big(|1\rangle\langle 0| - |0\rangle\langle 1|\Big)|1\rangle = i|1\rangle\langle 0|1\rangle - i|0\rangle\langle 1|1\rangle = -i|0\rangle \\ Y|\psi\rangle &= Y\Big(\alpha|0\rangle + \beta|1\rangle\Big) = \alpha Y|0\rangle + \beta Y|1\rangle = i\beta|0\rangle - i\alpha|1\rangle \end{aligned} \tag{2.25}$$

Its graphical representation is given in Fig.2.7, while its mathematical descriptions are in Eq.2.26.

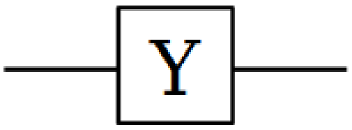


Figure 2.7: Y Gate.

$$Y = i\Big(|1\rangle\langle 0| - |0\rangle\langle 1|\Big)$$
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{2.26}$$

Z Gate

Gate Z performs a sign inversion on the quantum state's component $|1\rangle$ while leaving the component $|0\rangle$ unchanged, or $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in $\alpha|0\rangle - \beta|1\rangle$ (Eq.2.27).

$$\begin{aligned} Z|0\rangle &= \Big(|0\rangle\langle 0| - |1\rangle\langle 1|\Big)|0\rangle = |0\rangle\langle 0|0\rangle - |1\rangle\langle 1|0\rangle = |0\rangle \\ Z|1\rangle &= \Big(|0\rangle\langle 0| - |1\rangle\langle 1|\Big)|1\rangle = |0\rangle\langle 0|1\rangle - |1\rangle\langle 1|1\rangle = -|1\rangle \\ Z|\psi\rangle &= Z\Big(\alpha|0\rangle + \beta|1\rangle\Big) = \alpha Z|0\rangle + \beta Z|1\rangle = \alpha|0\rangle - \beta|1\rangle \end{aligned} \tag{2.27}$$

Its graphical representation is given in Fig.2.8, while its mathematical descriptions are in Eq.2.28.

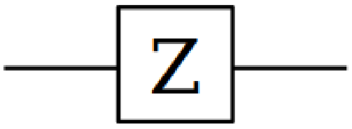


Figure 2.8: Z Gate.

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|$$
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.28}$$

H Gate

Gate H, also called *Hadamard Gate*, maps the computational basis vectors $|0\rangle$ and $|1\rangle$ into the basis vectors $|+\rangle$ and $|-\rangle$, respectively (Eq.2.29). Moreover, if the computational basis state is provided, it can produce an equivalent *superposition state.*

$$\begin{aligned}
H|0\rangle &= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|\Big)|0\rangle = \\
&= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0|0\rangle + |1\rangle\langle 0|0\rangle + |0\rangle\langle 1|0\rangle - |1\rangle\langle 1|0\rangle\Big) = \\
&= \frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle \\
H|1\rangle &= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|\Big)|1\rangle = \\
&= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0|1\rangle + |1\rangle\langle 0|1\rangle + |0\rangle\langle 1|1\rangle - |1\rangle\langle 1|1\rangle\Big) = \\
&= \frac{\sqrt{2}}{2}|0\rangle - \frac{\sqrt{2}}{2}|1\rangle \\
H|\psi\rangle &= H\Big(\alpha|0\rangle + \beta|1\rangle\Big) = \alpha H|0\rangle + \beta H|1\rangle = \\
&= \frac{\sqrt{2}}{2}\alpha|0\rangle + \frac{\sqrt{2}}{2}\alpha|1\rangle + \frac{\sqrt{2}}{2}\beta|0\rangle - \frac{\sqrt{2}}{2}\beta|1\rangle = \\
&= \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle
\end{aligned} \tag{2.29}$$

Its graphical representation is given in Fig.2.9, while its mathematical descriptions are in Eq.2.30.

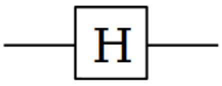


Figure 2.9: H Gate.

$$\begin{aligned}
H &= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|\Big) \\
H &= \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} = \frac{\sqrt{2}}{2}\Big(X + Z\Big)
\end{aligned} \tag{2.30}$$

Some other remarkable properties can thus be derived:

- $H \cdot H = \mathbb{I}$
- $H \cdot Z \cdot H = X$
- $H \cdot X \cdot H = Z$
- $H|0\rangle = |+\rangle \ \wedge \ H|1\rangle = |-\rangle$
- $H|+\rangle = |0\rangle \ \wedge \ H|-\rangle = |1\rangle$

Phase Shift Gates

Phase Shift Gates are a family of quantum gates that performs a phase shift on the quantum state's component $|1\rangle$ while leaving the component $|0\rangle$ unchanged, or $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in $\alpha|0\rangle + e^{i\varphi}\beta|1\rangle$ (Eq.2.31). The phase-shift angle is $\varphi \in [0, 2\pi) \subset \mathbb{R}$.

$$\begin{aligned}
P(\varphi)|0\rangle &= \Big(|0\rangle\langle 0| + e^{i\varphi}|1\rangle\langle 1|\Big)|0\rangle = |0\rangle\langle 0|0\rangle + e^{i\varphi}|1\rangle\langle 1|0\rangle = |0\rangle \\
P(\varphi)|1\rangle &= \Big(|0\rangle\langle 0| + e^{i\varphi}|1\rangle\langle 1|\Big)|1\rangle = |0\rangle\langle 0|1\rangle + e^{i\varphi}|1\rangle\langle 1|1\rangle = e^{i\varphi}|1\rangle \\
P(\varphi)|\psi\rangle &= P(\varphi)\Big(\alpha|0\rangle + \beta|1\rangle\Big) = \alpha P(\varphi)|0\rangle + \beta P(\varphi)|1\rangle = \\
&= \alpha|0\rangle + e^{i\varphi}\beta|1\rangle
\end{aligned} \tag{2.31}$$

Its graphical representation is given in Fig.2.10, while its mathematical descriptions are in Eq.2.32.

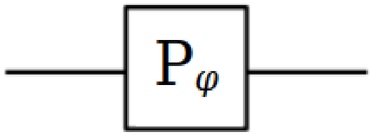


Figure 2.10: Phase Shift Gate.

$$\begin{aligned}
P(\varphi) &= |0\rangle\langle 0| + e^{i\varphi}|1\rangle\langle 1| \\
P(\varphi) &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}
\end{aligned} \tag{2.32}$$

Gate $P(\varphi)$ is a significant gate because it may be used to produce some particularly noteworthy gates for establishing a universal base of elementary gates for the design of compound quantum circuits:

$$\begin{aligned}
S &\overset{\text{def}}{=} P\left(\frac{\pi}{2}\right) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = \sqrt{Z} \\
T &\overset{\text{def}}{=} P\left(\frac{\pi}{4}\right) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} = \sqrt{S} = \sqrt[4]{Z}
\end{aligned} \tag{2.33}$$

In addition, it is essential to notice that

$$P(\pi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z$$

That justifies $S = \sqrt{Z}$ and $T = \sqrt[4]{Z}$.

CNOT Gate

The CNOT gate (Controlled NOT) acts on two qubits, where one qubit acts as a control input while the other is a conditioned input, meaning the output response depends on both input qubits. Tab.2.2 shows the input-output relationship for the CNOT gate. It is an irreplaceable component for obtaining a minimal basis of gates, acting on one or two qubits, that guarantee the universality of computation.

Input Qubits		**Output Qubits**	
Control	*Target*	*Control*	*Target*
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Table 2.2: Input-output relationship for the CNOT gate

More generally, following the convention that the most significant qubit is the control qubit and the other the target, CNOT gate applies this map: $|00\rangle \mapsto |00\rangle$, $|01\rangle \mapsto |01\rangle$, $|10\rangle \mapsto |11\rangle$, $|11\rangle \mapsto |10\rangle$; or, equally, it transforms the generic state $|\psi_{in}\rangle$ into $|\psi_{out}\rangle$, where

$$\begin{aligned} &|\psi_{in}\rangle = c_0\,|00\rangle + c_1\,|01\rangle + c_2\,|10\rangle + c_3\,|11\rangle \\ &|\psi_{out}\rangle = c_0\,|00\rangle + c_1\,|01\rangle + c_3\,|10\rangle + c_2\,|11\rangle \\ &\text{with } (c_0\,,\ c_1\,,\ c_2\,,\ c_3) \in \mathbb{C}^4 \quad \wedge \quad \sum_{j=0}^{3} |c_j|^2 = 1 \end{aligned} \tag{2.34}$$

Its graphical representation is given in Fig.2.11, while its mathematical descriptions are in Eq.2.35.

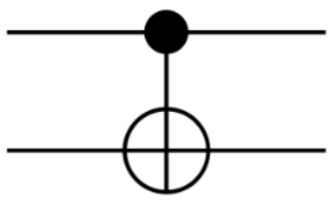


Figure 2.11: CNOT Gate.

$$\begin{aligned} CX &= |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X \\ CX &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned} \tag{2.35}$$

The CNOT gate is, in addition, particularly essential in quantum computation for three other reasons:

1. It is a fundamental item of several universal quantum gate sets, such as the group $\{CX, H, S, T\}$ or $\Big\{R_X(\theta), R_Y(\theta), R_Z(\varphi), CX\Big\}$.

2. Combined with the Hadamard gate, it has the capability to generate the so-called Bell's states, in which the two input qubits are closely interrelated and set in entanglement (see *quantum teleportation* in Section 2.4.1).

3. From a purely binary point of view, the CNOT on basis states behaves as a reversible XOR logic gate, with two inputs and two outputs, whose descriptive equation is $CX\Big(|a, b\rangle\Big) = |a, a \otimes b\rangle$. It is an essential component for implementing reversible boolean logic and arithmetic circuits.

Rotation Gates

The rotation operator gates perform the transformation on a single qubit given by Equations 2.18, 2.19, 2.20. They rotate the qubit's state around axes of the Bloch's sphere. They can be described by special unitary matrices, whose rotational angle is a continuous parameter with a period equal to 4π; however, it can be restricted when represented on Bloch's sphere, as in Eq.2.11. Their graphical representations are shown in Fig.2.12. Referring to Figure 2.12, the most important properties of these gates can be summarised as follows:

(a) **Rotation about X-axis**: $R_X\left(\theta\right)$

1) *exponential form*: $\exp\left(-iX\frac{\theta}{2}\right)$

2) *matrix form*: $\begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$

(b) **Rotation about Y-axis**: $R_Y\left(\theta\right)$

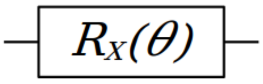
1) *exponential form*: $\exp\left(-iY\frac{\theta}{2}\right)$

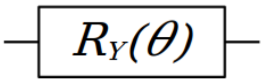
2) *matrix form*: $\begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$

(c) **Rotation about Z-axis**: $R_Z(\theta)$

1) *exponential form*: $\exp\left(-iZ\frac{\theta}{2}\right)$

2) *matrix form*: $\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} = e^{-i\frac{\theta}{2}} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} = P(\theta)$

(a) 

(b) 

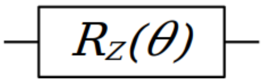
(c) 

Figure 2.12: Rotation of an angle θ around axis: (a) X (b) Y (c) Z

Moreover, it is always possible to derivate most of the main one-qubit gates from the composition of rotation gates; some simple examples are reported

$$\begin{aligned} &R_X(\pi) = -iX, \quad R_Y(\pi) = -iY, \quad R_Z(\pi) = -iZ \\ &R_Y\left(\frac{3}{2}\pi\right) \cdot X \cdot R_Y\left(\frac{\pi}{2}\right) = Z \\ &Z \cdot R_Y\left(\frac{3}{2}\pi\right) = H \end{aligned} \tag{2.36}$$

2.4.4 Quantum Register and Circuits

It is possible to generate four possible states with two classic bits: 00, 01, 10, 11. In general, 2^n distinct states can be constructed with n bits. Each normalized vector in the state space created by a system of n qubits has dimension 2^n and represents a potential computational state, which we will refer to as the n qubit quantum register.

This exponential increase in the size of state space means that a quantum computer could process data at a rate that is exponentially faster than a classical computer.

A quantum register with n qubits is formally a 2^n dimensional Hilbert space element, $\mathbb{C}^{2^n}$, with a computational basis generated by 2^n registers with n qubits.

$$|i_{n-1}\rangle \otimes |i_{n-2}\rangle \otimes .... \otimes |i_1\rangle \otimes |i_0\rangle \qquad i_j \in \{0,1\} \;\wedge\; 0 \leq j \leq n-1.$$

$|i_{n-1}\rangle|i_{n-2}\rangle....|i_0\rangle$, or more simply $|i_{n-1}i_{n-2}....i_0\rangle$, is the name for a base vector (their set is known as *computational basis*). As a result, using two qubits and the vectors $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, we can create the computational foundation of the state space. The vector $|01\rangle$ can be written as $|0\rangle \otimes |1\rangle$, that is the tensor product of $|0\rangle$ e $|1\rangle$, as shown before.

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \qquad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \tag{2.37}$$

For instance, a two-qubit quantum register is characterised by a complex superposition of states, as seen in the equation below:

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle \qquad \textit{with} \quad \sum_{j=0}^{3} |\alpha_j|^2 = 1$$

It is precisely with the superposition of states that quantum parallelism occurs in all its power.

Designing a quantum algorithm means constructing a quantum circuit formed by the series-parallel combination of elementary quantum gates: an example circuit is shown in Fig.2.13. A quantum circuit uses different

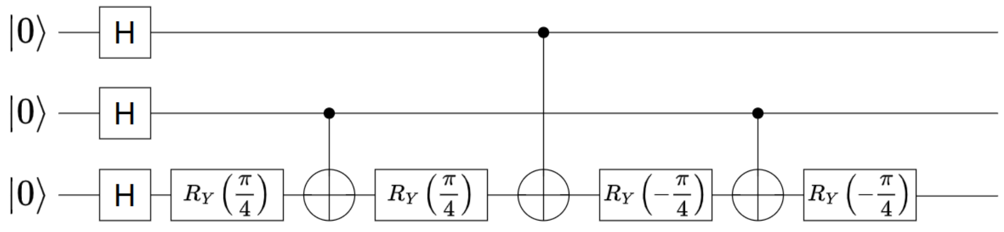

$|0\rangle$
$|0\rangle$
$|0\rangle$
H
H
H
$R_Y\left(\frac{\pi}{4}\right)$
$R_Y\left(\frac{\pi}{4}\right)$
$R_Y\left(-\frac{\pi}{4}\right)$
$R_Y\left(-\frac{\pi}{4}\right)$

Figure 2.13: An example of quantum circuit

models to be solved: gates are quantum operators that can be described mathematically. The two most straightforward ways to handle operators and the transformations they perform on qubits are the *algebra of Dirac's operators* and the *algebra of matrices*. In both descriptions, the operators are linear.

Series Circuits

A simple series circuit presents an ordered sequence of gates acting on a single qubit, as in Fig.2.14.

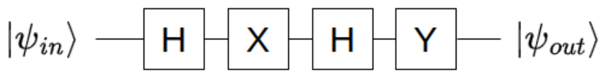


Figure 2.14: An example of series circuit

The global evolution of the system, from the initial qubit's state to the final transformation, according to the two mathematical perspectives, is given below.

- **Dirac's notation**: Let $|\psi_{in}\rangle = \alpha|0\rangle + \beta|1\rangle$, with α and β as in Equation 2.6,

$$\begin{aligned}
|\psi_{out}\rangle &= Y \cdot H \cdot X \cdot H\,|\psi_{in}\rangle = Y \cdot H \cdot X \cdot H(\alpha|0\rangle + \beta|1\rangle) = \\
&= Y \cdot H \cdot X\left(\frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle\right) = && \text{(from Eq.2.29)} \\
&= Y \cdot H\left(\frac{\alpha+\beta}{\sqrt{2}}X|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}X|1\rangle\right) = \\
&= Y \cdot H\left(\frac{\alpha-\beta}{\sqrt{2}}|0\rangle + \frac{\alpha+\beta}{\sqrt{2}}|1\rangle\right) = && \text{(from Eq.2.23)} \\
&= Y\left(\frac{\alpha-\beta}{\sqrt{2}}H|0\rangle + \frac{\alpha+\beta}{\sqrt{2}}H|1\rangle\right) = \\
&= Y\left(\frac{\alpha-\beta}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{\alpha+\beta}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = \\
&= Y\left(\frac{\alpha-\beta}{2}|0\rangle + \frac{\alpha-\beta}{2}|1\rangle + \frac{\alpha+\beta}{2}|0\rangle - \frac{\alpha+\beta}{2}|1\rangle\right) = \\
&= Y(\alpha|0\rangle - \beta|1\rangle) = \alpha Y|0\rangle - \beta Y|1\rangle = \\
&= i\alpha|1\rangle + i\beta|0\rangle = i\beta|0\rangle + i\alpha|1\rangle = && \text{(from Eq.2.25)} \\
&= i(\beta|0\rangle + \alpha|1\rangle) = e^{i\frac{\pi}{2}} \cdot X\,|\psi_{in}\rangle
\end{aligned}$$

So, $|\psi_{out}\rangle = i\beta\,|0\rangle + i\alpha\,|1\rangle$, while the block $Y \cdot H \cdot X \cdot H$ behaves as a single gate X except for an inconsequential global phase, $e^{i\frac{\pi}{2}}$.

- **Matricial notation**: Let $|\psi_{in}\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$, with α and β as in Eq.2.6,

$$\begin{aligned}
|\psi_{out}\rangle &= Y \cdot H \cdot X \cdot H \cdot |\psi_{in}\rangle = Y \cdot H \cdot X \cdot H \cdot \begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \\
&= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{pmatrix} \cdot \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{pmatrix} \cdot \begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \\
&= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{pmatrix} \cdot \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\alpha+\beta}{\sqrt{2}}\\ \frac{\alpha-\beta}{\sqrt{2}}\end{pmatrix} = \\
&= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{pmatrix} \cdot \begin{pmatrix}\frac{\alpha-\beta}{\sqrt{2}}\\ \frac{\alpha+\beta}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \cdot \begin{pmatrix}\alpha\\ -\beta\end{pmatrix} = \\
&= \begin{pmatrix}i\beta\\ i\alpha\end{pmatrix} = i \cdot \begin{pmatrix}\beta\\ \alpha\end{pmatrix}
\end{aligned}$$

As in the previous notation, it can be shown that the block $Y \cdot H \cdot X \cdot H$ behaves as a single gate X except for an inconsequential global phase, $e^{i\frac{\pi}{2}}$.

$$\begin{aligned}
Y \cdot H \cdot X \cdot H &= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{pmatrix} \cdot \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \cdot \begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\end{pmatrix} = \\
&= \begin{pmatrix}0 & i\\ i & 0\end{pmatrix} = i \cdot \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} = i \cdot X = e^{i\frac{\pi}{2}} \cdot X
\end{aligned}$$

Parallel Circuits

A simple parallel circuit presents an ordered vertical sequence of gates acting at the same instant of time on two or more qubits, as in Fig.2.15. It is essential to determine which should be the most significant qubit in the binary string that is to be represented by the circuit, thus deciding whether to start from the top or the bottom, as in Fig.2.15. The global evolution of the system, from the initial qubits' state to the final transformation, is given below. The convention followed in this example is to have the most significant qubit at the bottom of Fig.2.15, that is the most significant qubit is $|\psi_1\rangle$.

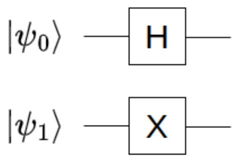


Figure 2.15: An example of parallel circuit

- **Dirac's notation**: Let $|\psi_0\rangle = \alpha|0\rangle + \beta|1\rangle$, $|\psi_1\rangle = \gamma|0\rangle + \delta|1\rangle$, with $(\alpha,\beta,\gamma,\delta) \in \mathbb{C}^4$ and $|\alpha|^2 + |\beta|^2 = 1 \ \wedge \ |\gamma|^2 + |\delta|^2 = 1$.

$$|\psi_{out}\rangle = (X|\psi_1\rangle) \otimes (H|\psi_0\rangle) = (X \otimes H) \cdot (|\psi_1\rangle \otimes |\psi_0\rangle)$$

where

$$\begin{aligned} X \otimes H &= \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|) \otimes (|0\rangle\langle 1| + |1\rangle\langle 0|) = \\ &= \frac{1}{\sqrt{2}}(|00\rangle\langle 10| + |00\rangle\langle 11| + |01\rangle\langle 10| - |01\rangle\langle 11| + \\ &+ |00\rangle\langle 10| + |00\rangle\langle 11| + |01\rangle\langle 10| - |01\rangle\langle 11|) \end{aligned}$$

and

$$\begin{aligned} |\psi_1\rangle \otimes |\psi_0\rangle &= \Big(\gamma|0\rangle + \delta|1\rangle\Big) \otimes \Big(\alpha|0\rangle + \beta|1\rangle\Big) = \\ &= \alpha\gamma|00\rangle + \beta\gamma|01\rangle + \alpha\delta|10\rangle + \beta\delta|11\rangle \end{aligned}$$

Considering that $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ is an orthonormal basis in Hilbert space $\mathbb{H}^{\otimes 2}$, the result of the computation is

$$\begin{aligned} |\psi_{out}\rangle &= (X \otimes H) \cdot (|\psi_1\rangle \otimes |\psi_0\rangle) = \\ &= \frac{1}{\sqrt{2}}\Big(\alpha\gamma|10\rangle + \alpha\gamma|11\rangle + \beta\gamma|10\rangle - \beta\gamma|11\rangle + \\ &+ \alpha\delta|00\rangle + \alpha\delta|01\rangle + \beta\delta|00\rangle - \beta\delta|01\rangle\Big) = \\ &= \frac{\alpha\delta + \beta\delta}{\sqrt{2}}|00\rangle + \frac{\alpha\delta - \beta\delta}{\sqrt{2}}|01\rangle + \frac{\alpha\gamma + \beta\gamma}{\sqrt{2}}|10\rangle + \frac{\alpha\gamma - \beta\gamma}{\sqrt{2}}|11\rangle \end{aligned}$$

- **Matricial notation**: Let $|\psi_1\rangle = \begin{pmatrix}\gamma \\ \delta\end{pmatrix}$ and $|\psi_0\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix}$, with α and β, with $(\alpha,\beta,\gamma,\delta) \in \mathbb{C}^4$ and $|\alpha|^2 + |\beta|^2 = 1 \ \wedge \ |\gamma|^2 + |\delta|^2 = 1$.

So,

$$|\psi_{out}\rangle = (X \otimes H) \cdot (|\psi_1\rangle \otimes |\psi_0\rangle) =$$

$$= \left[\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}\right] \cdot \left[\begin{pmatrix} \gamma \\ \delta \end{pmatrix} \otimes \begin{pmatrix} \alpha \\ \beta \end{pmatrix}\right] =$$

$$= \begin{pmatrix} 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha\gamma \\ \beta\gamma \\ \alpha\delta \\ \beta\delta \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}\alpha\delta}{2} + \frac{\sqrt{2}\beta\delta}{2} \\ \frac{\sqrt{2}\alpha\delta}{2} - \frac{\sqrt{2}\beta\delta}{2} \\ \frac{\sqrt{2}\alpha\gamma}{2} + \frac{\sqrt{2}\beta\gamma}{2} \\ \frac{\sqrt{2}\alpha\gamma}{2} - \frac{\sqrt{2}\beta\gamma}{2} \end{pmatrix} =$$

$$= \begin{pmatrix} \frac{\alpha\delta+\beta\delta}{\sqrt{2}} \\ \frac{\alpha\delta-\beta\delta}{\sqrt{2}} \\ \frac{\alpha\gamma+\beta\gamma}{\sqrt{2}} \\ \frac{\alpha\gamma-\beta\gamma}{\sqrt{2}} \end{pmatrix}$$

This result perfectly fits with what was previously obtained with the Dirac's notation, confirming the equivalence of the two notations.

2.4.5 An example of quantum circuit: the *entangler circuit*

The **entangler** (Fig.2.16) is an example of a circuit/algorithm used to generate a specific element of Bell's base (Eq.2.10) from state $|00\rangle$. Equation 2.38 explains how qubits behave based on the chosen ordering convention for this circuit. In this case, the convention utilised is the *Most Significant Qubit at the top.*

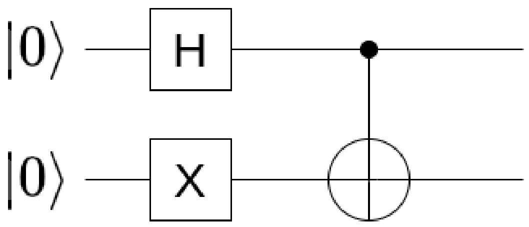


Figure 2.16: Entangler Circuit

$$
\begin{aligned}
|\psi_{out}\rangle = CX \cdot \Big(H \otimes X\Big) \cdot |00\rangle = \\
&= \Big(|0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X\Big) \cdot \Big(H \otimes X\Big) \cdot |00\rangle = \\
&= \Big(|0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X\Big) \cdot \Big(H \otimes X\Big) \cdot (|0\rangle \otimes |0\rangle) = \\
&= \Big(|0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X\Big) \cdot \Big(H|0\rangle \otimes X|0\rangle\Big) = \\
&= \Big(|0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X\Big) \cdot \frac{1}{\sqrt{2}}\Big((|0\rangle + |1\rangle) \otimes |1\rangle\Big) = \\
&= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0|0\rangle + |0\rangle\langle 0|1\rangle\Big) \otimes \Big(\mathbb{I}|1\rangle\Big) + \\
&+ \frac{1}{\sqrt{2}}\Big(|1\rangle\langle 1|0\rangle + |1\rangle\langle 1|1\rangle\Big) \otimes \Big(X|1\rangle\Big) = \\
&= \frac{1}{\sqrt{2}}\Big(|0\rangle \otimes |1\rangle\Big) + \frac{1}{\sqrt{2}}\Big(|1\rangle \otimes |0\rangle\Big) = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle
\end{aligned}
\tag{2.38}
$$

An utterly equivalent result can be obtained using matrix algebra (Eq.3.18). Table 2.3 shows the probability distribution obtained after a measurement process on the two qubits for the quantum state in Eq.2.38.

$\|\psi\rangle$	**Probability**	
component	**amplitude**	**distribution**
$\|00\rangle$	0	0
$\|01\rangle$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$
$\|10\rangle$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$
$\|11\rangle$	0	0

Table 2.3: Component States of $|\psi_{out}\rangle$ and their probability distribution

In order to verify the correctness of the theoretical results, the quantum entangler circuit is implemented, first using a Python programming language library dedicated to quantum circuit simulation (Qiskit[3]) and then IBM's cloud environment, IBM Quantum Experience[4], which among other things, allows theoretical results to be verified by giving access to real quantum computers. Section 4.2 describes some cloud platforms for quantum circuit simulation, including IBM Quantum Experience. Using the Qiskit's conventions, the Entangler circuit is depicted in Fig.2.17. In Fig.2.18a, the

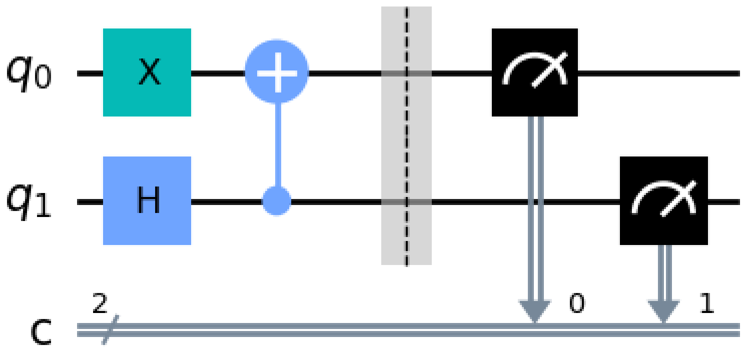


Figure 2.17: Circuit for Entangler using Qiskit

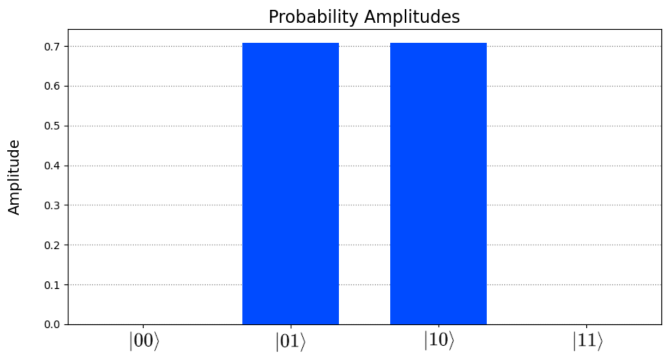
components of the output state can be seen (their phases are indicated by the colours given by the bar in Fig.2.18c). Fig.2.18b shows the probability distributions on the states of the computational basis obtained from the measurement process on observable Z; the number of samples used is 8,192.

The same circuit was subsequently loaded onto IBM's quantum computer, named *ibmq_manila*, with 5 qubits, Quantum Volume (QV) equal to 32, Circuit Layer Operation per Second (CLOPS) equal to 2.8K, quantum processor model *Falcon r5.11L*; in order to better understand the impressive performance of the *ibm_manila* platform, Figure 2.19 displays some technical data about this small quantum computer, which consists of only five qubits.

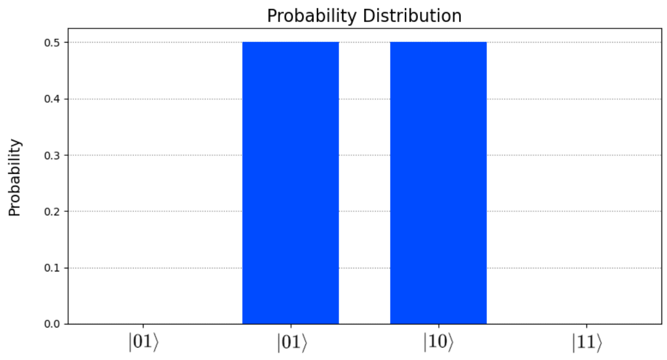
The Probability Distribution is shown in Fig.2.20.

Executing a job on a real quantum computer is not an ideal process, so measured samples inevitably appear on components of the computational

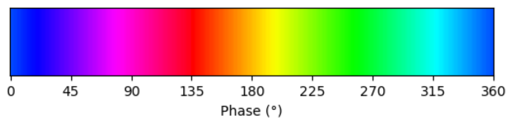
[3]`https://qiskit.org/`
[4]`https://quantum-computing.ibm.com`



a) Probability Amplitudes



b) Probability Distribution of Samples (8,192 shots)



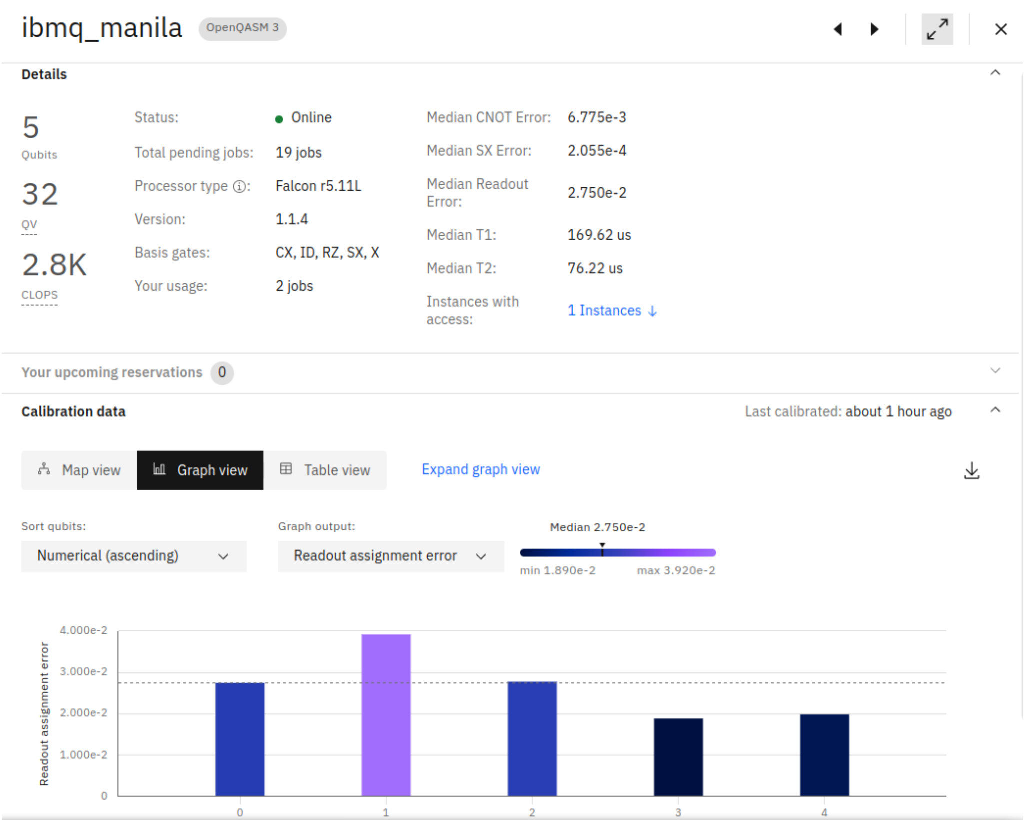
c) Phase Bar

Figure 2.18: Probability Amplitudes of the output state in the Entangler

basis that theoretically should not be present (in the specific case of Fig.2.20, there are minor spurious measurements on $|00\rangle$ and $|11\rangle$).

In Chapter 4, some of the most common quantum algorithms are presented, and three quantum algorithms able to take an exponential advantage from the superposition of states will be analysed in detail:

- the Deutsch–Jozsa's algorithm
- the Grover's algorithm
- the Shor's algorithm



ibmq_manila OpenQASM 3

Details

5 Qubits

32 QV

2.8K CLOPS

Status:	● Online	Median CNOT Error:	6.775e-3
Total pending jobs:	19 jobs	Median SX Error:	2.055e-4
Processor type ⓘ:	Falcon r5.11L	Median Readout Error:	2.750e-2
Version:	1.1.4	Median T1:	169.62 us
Basis gates:	CX, ID, RZ, SX, X	Median T2:	76.22 us
Your usage:	2 jobs	Instances with access:	1 Instances ↓

Your upcoming reservations 0

Calibration data Last calibrated: about 1 hour ago

Map view | Graph view | Table view | Expand graph view

Sort qubits: Numerical (ascending)

Graph output: Readout assignment error

Median 2.750e-2

min 1.890e-2 max 3.920e-2

Figure 2.19: Technical characteristics for *ibm_manila* system

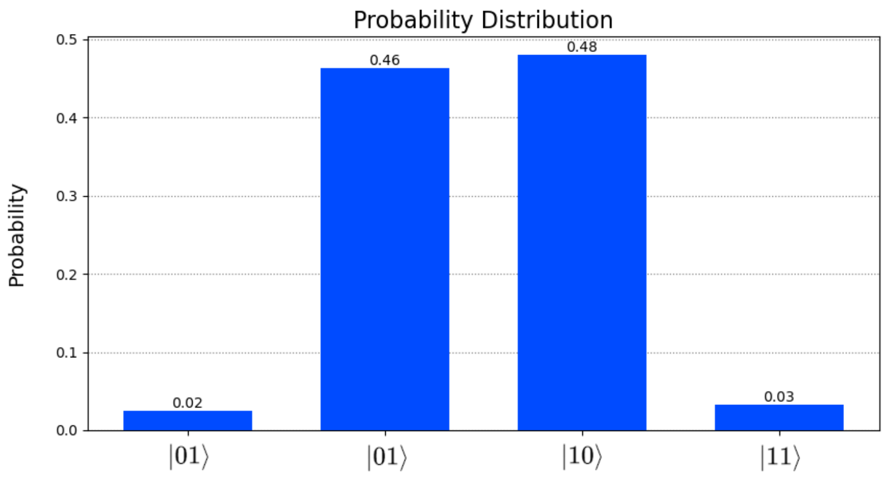


Figure 2.20: Probability Distribution of Samples (8,192 shots) on *ibm_manila* system

2.5 Conclusions

Quantum computing is a rapidly evolving field that aims to harness quantum mechanics' strange and counter-intuitive properties to process information in fundamentally different ways from classical computers. Qubits are quantum two-level systems that can be represented using a vector in a two-dimensional Hilbert space. Unlike classical bits, which can only exist in one of two possible states (0 or 1), qubits can be in a superposition of both states. That means a qubit can be in a form that is a linear combination of the 0 and 1 states. When a qubit is measured, it collapses into one of its two possible states with a probability determined by the superposition coefficients. Another essential property of qubits is entanglement, which occurs when two or more qubits are correlated in such a way that the state of one qubit is dependent on the state of the other qubit. Entanglement is a non-local property of quantum systems that allows for the possibility of quantum teleportation and other exotic phenomena. Qubits can be implemented using various physical systems, such as trapped ions, superconducting circuits, and nitrogen vacancy centres in diamonds. Each of these systems has its advantages and challenges, and researchers are actively exploring different qubit implementations to determine which will be the most practical for large-scale quantum computing. One of the most promising applications of qubits is in quantum computing. Quantum computers are based on quantum circuits built from gates that manipulate the state of one or more qubits. Quantum circuits can be used to perform a variety of operations, such as factorization and searching, that are exponentially faster than the best-known classical algorithms. Another notable application of qubits is in quantum simulation. Quantum simulation involves using a quantum computer to emulate the behaviour of quantum systems that are too complex to be simulated on classical computers. It has applications in fields such as chemistry, where quantum simulations can study the behaviour of complex molecules and materials. In addition to quantum computing and simulation, qubits have other potential applications in fields such as cryptography and metrology. For example, qubits can be used to implement quantum key distribution, a method for secure communication that is impossible to eavesdrop on without disturbing the quantum state. Qubits can also be used to implement quantum metrology, which involves using quantum systems to make precise measurements of physical quantities such as time and magnetic fields. Despite the tremendous promise of qubits, significant challenges must be overcome before large-scale quantum computing can become a reality. One of the biggest challenges is decoherence, which occurs when the quantum state of a qubit is disturbed by its environment. Decoherence can cause errors in quantum circuits and limit the number of qubits that can be reliably operated. Researchers are exploring diverse techniques to mitigate decoherence, such as error correction and fault-tolerant quantum computing.

To sum up, qubits are the basic units of quantum computing and work differently than classical computers. They can be in multiple states simultaneously and connected to each other. This unique function allows qubits to perform specific calculations exponentially faster than classical computers.

Chapter 3

Optical Quantum Computing

3.1 Introduction

Photon chips are one of the most promising platforms for quantum computing. They have several advantages over other physical systems, such as ions or superconducting qubits, that make them particularly suitable for specific quantum computing tasks.

The first advantage is there is no need for ultra-cold environments: indeed, superconducting or ion-trap qubits require extremely low temperatures, almost close to absolute zero, or very extreme low pressure to operate. This requires complex and costly cooling systems. On the other hand, photonic quantum systems can function at room temperature, which makes them more practical for some applications.

Photons' inherent ability to travel long distances without significant loss of coherence is an important advantage. That makes them ideal for use in quantum communication, where quantum information is transmitted over large distances through optical fibres or free-space channels. Photons are already a well-established technology for quantum communication, with several commercial applications such as quantum key distribution and secure transmission. Another benefit of photons is their ability to be manipulated using linear optical devices such as beam splitters, phase shifters, and polarizers. The main difficulty in using photons for quantum computing is their lack of strong interactions with each other. So, to perform entangling operations between two photons, it is necessary to use a non-linear optical element, which is typically weak and hard to control. However, recent advances in materials science have led to the development of new types of non-linear optical materials, such as diamond nitrogen-vacancy centres, that can be used to implement high-fidelity two-qubit gates between photons.

Moreover, photonic systems offer a pathway to scalability through quantum frequency combs or integrated photonic circuits. While scaling up remains a challenge for all quantum systems, photons provide a range of possibilities.

Performing quantum computation with photons faces the challenge of efficiently detecting single photons. Reliable detection of individual photons with high efficiency is necessary to perform accurate quantum computation. Several technologies for photon detection exist, including single-photon avalanche photodiodes (SPADs) and transition-edge sensors (TESs), but each has its strengths and weaknesses. However, the use of photons in quantum computing has already led to several impressive demonstrations of quantum algorithms, including reduced and simplified versions of Shor's algorithm for factoring large numbers and Grover's algorithm for database search. In addition to quantum computing and communication, photons are also being explored in other quantum technologies, such as quantum sensing and metrology.

It is essential to comprehend that each type of quantum technology - whether it is photonic, ion-trap or superconducting - has its own set of advantages and limitations. Depending on the application or task, one of these technologies may be better suited than the others. For example, superconducting qubits have made remarkable progress in terms of coherence times and error rates, while ion traps provide a high degree of control over individual qubits. Therefore, although photonic quantum technology offers numerous benefits, it does not render the other technologies useless. The future of quantum technology and computing may involve a hybrid approach that combines the strengths of each platform.

This chapter will mainly focus on photonics technology for the reasons mentioned above and due to the author's curiosity and interest in them. It should be noted that the advent of hybrid quantum systems can indeed offer one potential solution to the problems that the different technologies provide.

3.2 Qubits and Qumodes

While a qubit is a quantum mechanical system that can exist in two states, commonly referred to as $|0\rangle$ and $|1\rangle$, or in a superposition of them, a *qumode* is a quantum mechanical system that can exist in a continuum of states. In other words, instead of having discrete states like qubits, qumodes can have an infinite number of possible states (Eq.3.1).

$$\begin{aligned} &\textit{qubit} && |\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \\ &\textit{qumode} && |\psi\rangle = \int dx\ \psi(x)\,|x\rangle \end{aligned} \tag{3.1}$$

The most common example of a qumode is the electromagnetic field, which can be used to implement continuous-variable quantum computing. In *continuous-variable quantum computing* (CV model), qumodes are manipulated using operations that change the amplitude and phase of the electromagnetic field. These operations are typically implemented using optical devices such as beam splitters, phase shifters, and squeezers. By manipulating the electromagnetic field, continuous-variable quantum computers can simultaneously perform operations on multiple qumodes, making them potentially more powerful than discrete-variable quantum computers, such as those based on the qubit. In optics, the *discrete-variable quantum computing* (DV model) employs two *optical modes*, such as optical fibres or orthogonal free propagation, to transmit pairs of photons. The presence or absence of a photon in each fibre or its polarization indicates the two potential states of a qubit. Modifying the phase of one photon in each pair makes it feasible to carry out quantum operations on the equivalent qubit's description.

A possible application of these two models is *Quantum Machine Learning* (QML). Chapter 5 will present two parallel examples of QML applied to the same dataset, one using the *fermionic discrete variable model* and the other using the *bosonic continuous variable model*. In the following paragraphs, some basic concepts for helping to understand the two models will be illustrated: for the CV model, the main gates used for the computation will be listed, but above all, the ones useful for understanding the second part of Chapter 5, where an example of this model applied to Quantum Machine Learning will be shown; the DV model will be illustrated a little more in-depth because of the similarities it has with the model of classical quantum computation, i.e. the one with fermions and qubits.

3.2.1 CV Quantum Gates

A way to encode information in a continuous variable model with photons is by using a technique called *Gaussian encoding*, as it uses the Gaussian states of bosons.

The vacuum state $|0\rangle$ is the primary and essential state from which all other states can be produced. By evolving the vacuum state with a bosonic Hamiltonian H and evolution time t, it is possible to generate all other states expressed as $|\psi\rangle = \exp(-itH)\,|0\rangle$. The resulting states are called Gaussian if the Hamiltonian H is at most quadratic in the operators $\hat{x}$ and $\hat{p}$. Equation 3.2 reports some Gaussian states useful in quantum computation. The *coherent state* and the *squeezed state* can be respectively obtained by applying the displacement operator (Eq.3.5) and the squeezing operator (Eq.3.4) to the vacuum state.

$$\begin{aligned}
&\textit{vacuum state} && |0\rangle = \frac{1}{\sqrt[4]{\pi\hbar}} \int dx\ e^{-x^2/(2\hbar)}\,|x\rangle && \\
&\textit{coherent state} && |\alpha\rangle = D(\alpha)\,|0\rangle && \text{with } \alpha \in \mathbb{C} \\
&\textit{squeezed state} && |z\rangle = S(z)\,|0\rangle && \text{with } z \in \mathbb{C}
\end{aligned} \tag{3.2}$$

In Gaussian encoding, the information is encoded in the amplitude and phase of a coherent state of light, a quantum state that behaves similarly to classical light. The amplitude and phase can represent binary information, with the amplitude corresponding to a '0' or '1' value and the phase related to the sign of the value. Various quantum operations can be applied to the state to manipulate the encoded information, such as squeezing, displacement, and phase shifts. These operations can be performed using linear optical components such as beam splitters and phase shifters or non-linear components such as Kerr Gate or Cubic Phase Gate. The mathematical transformations performed by the main gates used in CV are listed below:

Rotational Gate

Phase space rotational gate (single-mode gate), useful to rotate mean value and variation of the bosonic distribution in the phase space diagram.

$$R(\phi) = \exp\left(i\phi\hat{a}^\dagger\hat{a}\right) = \exp\left(i\frac{\phi}{2}\left(\frac{\hat{x}^2+\hat{p}^2}{\hbar} - \hat{\mathbf{1}}\right)\right) \tag{3.3}$$

where $\phi \in [0; 2\pi) \subset \mathbb{R}$

Squeezing Gate

Phase space squeezing gate (single-mode gate), useful to squeeze and rotate mean value and variation of the bosonic distribution in the phase space diagram.

$$S(z) = \exp\left(\frac{1}{2}\left(z^*\hat{a}^2 - z\hat{a}^{\dagger\,2}\right)\right) = \exp\left(\frac{r}{2}\left(e^{-i\phi}\hat{a}^2 - e^{i\phi}\hat{a}^{\dagger\,2}\right)\right) \tag{3.4}$$

where $z = re^{i\phi}$ with $r \in \mathbb{R} \mid r \geqslant 0 \wedge \phi \in [0; 2\pi) \subset \mathbb{R}$

Displacement Gate

Phase space displacement gate (single-mode gate), useful to shift the mean value and variation of the bosonic distribution in the phase space diagram.

$$D(z) = \exp(za^\dagger - z^*a) = \exp\left(-i\sqrt{\frac{2}{\hbar}}\left(\mathfrak{Re}(z)\hat{p} - \mathfrak{Im}(z)\hat{x}\right)\right) \tag{3.5}$$

where $z = re^{i\phi}$ with $r \in \mathbb{R} \mid r \geqslant 0 \wedge \phi \in [0; 2\pi) \subset \mathbb{R}$

Kerr Gate

The Kerr interaction gate (single-mode gate) is useful for applying a specific non-linear effect on the bosonic distribution.

$$K(\kappa) = \exp\left(i\kappa\hat{n}^2\right) \tag{3.6}$$

with κ real constant for Kerr effect and $\hat{n}$ is the Number Operator.

Beam Splitter Gate

Beamsplitter interaction gate (two-modes gate), helpful for creating a quantum superposition effect.

$$BS(\theta,\phi) = \exp\left(\theta\left(e^{i\phi}\hat{a}\hat{b}^\dagger - e^{-i\phi}\hat{a}^\dagger\hat{b}\right)\right) \tag{3.7}$$

where $\theta \in [0; 2\pi) \subset \mathbb{R} \wedge \phi \in [0; 2\pi) \subset \mathbb{R}$.
Let $|\alpha,\beta\rangle$ a coherent state, the beamsplitter transforms it to a new coherent state $|\alpha',\beta'\rangle$ in such a way that

$$\begin{aligned}\alpha' &= \alpha\cos\theta - \beta e^{-i\phi}\sin\theta = t\alpha - r^*\beta\\ \beta' &= \beta\cos\theta + \alpha e^{i\phi}\sin\theta = t\beta + r\alpha\end{aligned}$$

where $\boldsymbol{t = \cos\theta}$, called *transmittivity* amplitude of the beamsplitter and $\boldsymbol{r = e^{i\phi}\sin\theta}$, named *reflectivity* amplitude of the beamsplitter.

Two-mode Squeezing Gate

Two-mode squeezing interaction (two-modes gate), used to perform some non-linear effects, such as entanglement.

$$S_2(z) = \exp\left(z\hat{a}_1^\dagger\hat{a}_2^\dagger - z^*\hat{a}_1\hat{a}_2\right) = \exp\left(r(e^{i\phi}\hat{a}_1^\dagger\hat{a}_2^\dagger - e^{-i\phi}\hat{a}_1\hat{a}_2\right) \tag{3.8}$$

where $z = re^{i\phi}$ with $r \in \mathbb{R} \mid r \geqslant 0 \wedge \phi \in [0; 2\pi) \subset \mathbb{R}$.

It is also equivalent to two opposite local squeezers sandwiched between two 50% phase-less beamsplitters:

$$S_2(z) = BS^\dagger\left(\frac{\pi}{4}, 0\right) \cdot [S(z) \otimes S(-z)] \cdot BS\left(\frac{\pi}{4}, 0\right)$$

CrossKerr Gate

Two-mode Kerr interaction gate (two-modes gate)

$$CK(\kappa) = \exp\left(i\kappa\hat{n}_1^2\hat{n}_2^2\right) \tag{3.9}$$

with κ real constant for Kerr effect and $\hat{n}_i$ is the Number Operator on each mode.

3.2.2 DV Quantum Circuits

A Discrete Variable quantum circuit may be implemented using an optical circuit. There are several alternative encodings, such as *spatial modes encoding* and *polarisation modes encoding*. Each qubit in a quantum circuit has a pair of spatial qumodes when using spatial modes encoding; in this chapter, the terms modes and qumodes will always be synonymous for quantum optical modes. One photon in one of the spatial modes corresponds to each qubit state in a Fock state[1]: so, the quantum state $|0\rangle$ can be encoded as $|1, 0\rangle$ in spatial mode, where one photon is in the first qumode, and no photons are in the second one; consequently, the quantum state $|1\rangle$ can be encoded as $|0, 1\rangle$. This model is called *Dual Rail Mode*.

In other words, two spatial modes are used to represent each qubit, with one mode corresponding to the logical 0-state (0_L) and the other mode corresponding to the logical 1-state (1_L). The qubit state is then encoded as a superposition of the two spatial modes, with the relative amplitudes of the modes determining the probability of measuring the qubit in the 0 or 1 state [58]. One advantage of using spatial modes for

[1]The Fock space is a mathematical construction used to describe the quantum states of a system of identical particles, such as electrons, photons, or atoms. It is a Hilbert space that is built up from a vacuum state, which represents the absence of any particles, and a set of creation and annihilation operators, which create and destroy particles. The Fock space is a powerful tool for calculating the probabilities of different quantum states of a system of identical particles.

dual rail encoding in optical quantum computing is that the modes can be easily manipulated using optical components such as beam splitters and phase shifters. That allows for implementing quantum gates that can operate on multiple qubits simultaneously, which is essential for scaling up quantum computing systems. Moreover, spatial mode encoding can encode multiple qubits in a single photon, simplifying the hardware requirements for building a quantum computing system. For example, a single photon can be encoded with four spatial modes to represent two qubits, allowing for the implementation of multi-qubit gates with a single photon source [59]. However, implementing spatial mode encoding in optical quantum computing also has some challenges, such as the need for high-quality photon sources, precise control of optical components, the mitigation of errors due to imperfections in the encoding and decoding processes and the extreme difficulty of implementing a deterministic CNOT gate, or a CZ (Controlled Z gate), using only linear optics. Nevertheless, this approach holds promise for developing scalable and fault-tolerant quantum computing systems based on optical technologies.

Another possible coding is related to the prospect of using a single photon to represent a quantum state in a truncated Fock's space (*finite Fock's space*). Such a form of arithmetic representation would be equivalent to writing the natural numbers in base 1 and it is called *Single-Photon Mode*. Thus in a Fock's space with four qumodes, the various states could be described as in Tab.3.1. In the second column of the table, we find the classical representation of the qubit in a Hilbert's space, as already presented in section 2.4.4. In the third and fourth columns, the information is encoded in a Fock's space with four qumodes and one photon, whose position determines the number natural to represent; they represent two equivalent conventions: least significant digit to the left or the right. The most commonly used convention is that relating to the third column, as it is strongly affine to the definition of quantum register in Eq.2.37.

Quantum Optical Linear Circuit (QOLC) is a linear optical circuit that is used to manipulate quantum states of light. It is composed of passive linear optical elements, such as beam splitters and phase shifters, and single-photon detectors. The beam splitter and the phase shifter can be

Number	Qubit Hilbert	qumode Fock	qumode Fock
0	$\lvert 00\rangle$	$\lvert 1,0,0,0\rangle$	$\lvert 0,0,0,1\rangle$
1	$\lvert 01\rangle$	$\lvert 0,1,0,0\rangle$	$\lvert 0,0,1,0\rangle$
2	$\lvert 10\rangle$	$\lvert 0,0,1,0\rangle$	$\lvert 0,1,0,0\rangle$
3	$\lvert 11\rangle$	$\lvert 0,0,0,1\rangle$	$\lvert 1,0,0,0\rangle$

Table 3.1: Example of possible encodings for qubits in Single-Photon Mode

described mathematically through a unitary Gaussian matrix and constitute the reprogrammable part for the optical quantum chip [60]; quadratic Hamiltonians generate them in the creation and annihilation operators $\hat{a}^\dagger$ and $\hat{a}$, respectively. The composition of a QOLC is critical to its operation and performance. The main components of a QOLC are:

- **Input port**: it is a physical port where the quantum state of light is introduced into the circuit. The input port can be a single-mode or multi-mode fibre, depending on the application; it must be designed to minimise losses and maintain the coherence of the quantum state.

  A *single photon source* is always present upstream of the input port. During its propagation within the quantum circuit, its distribution wave is processed to enable computation. Single photon sources are defined by specifying parameters such as *brightness*, *purity* or *indistinguishability* [61, 62, 63]. The model that describes a single photon source is the perfect/imperfect quantum-dot-based single-photon source obtained by a statistical combination of Fock states.

- **Phase shifters**: they are passive linear optical elements that change the phase of a beam of light. Phase shifters are used to adjust the photon's phase to create interference patterns and manipulate the quantum state of light.

  It has one parameter, the phase shift θ, with $\theta \in [0; 2\pi] \subset \mathbb{R}$ and $\hbar$ is the *reduced Planck's constant.*

  The Hamiltonian and the Unitary (Eq.3.3) for this optical element are

$$H = \hbar\theta\hat{a}^\dagger\hat{a} \quad \mapsto \quad U(\theta) = \exp(i\theta\hat{a}^\dagger\hat{a}) \tag{3.10}$$

So, the transformed mode, that is the output qumode, is given by the following equation:

$$\hat{b} = U(\theta)\hat{a}U^{\dagger}(\theta) = \hat{a}e^{i\theta} \tag{3.11}$$

Therefore, the phase shifter is an optical element acting on a single qumode. Figure 3.1 depicts the graphic symbol used in quantum optical circuits for phase shifters.

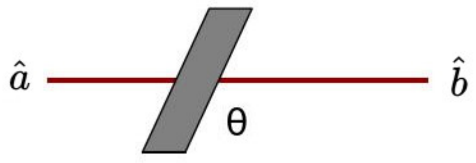


Figure 3.1: Phase Shifter gate

- **Beam splitters**: they are passive linear optical elements that split a beam of light into two or more beams (cf. Section 3.2.1). Beam splitters are the primary building blocks of a QOLC. They are used to separate and recombine photons. By adjusting the reflectivity and transmissivity of the beam splitter, interference patterns can be created to manipulate the quantum state of light. In certain encoding forms, such as Single Photon Mode, they are also used to produce entangled states, a critical resource for quantum communication and quantum computing.

  The Hamiltonian and the Unitary (Eq.3.7, with a natural phase reflection equal to $\frac{\lambda}{4}$, that is $\phi = \frac{\pi}{2}$) for this optical element are

$$H = \hbar\,\frac{\theta}{2}\,\hat{a}_1^{\dagger}\hat{a}_2 + \hbar\,\frac{\theta}{2}\,\hat{a}_2^{\dagger}\hat{a}_1 \quad\mapsto\quad U(\theta) = \exp\left(i\,\frac{\theta}{2}(\hat{a}_1^{\dagger}\hat{a}_2 + \hat{a}_2^{\dagger}\hat{a}_1)\right) \tag{3.12}$$

  So, the transformed mode, that is the output qumode, is given by the following equation:

$$\begin{pmatrix}\hat{b}_1\\ \hat{b}_2\end{pmatrix} = \begin{pmatrix}\cos\frac{\theta}{2} & i\sin\frac{\theta}{2}\\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{pmatrix}\cdot\begin{pmatrix}\hat{a}_1\\ \hat{a}_2\end{pmatrix} \qquad \text{with } \theta\in[0,4\pi]\subset\mathbb{R} \tag{3.13}$$

where $\hat{a}_1^\dagger$ and $\hat{a}_2^\dagger$ are the creation operators for the input photons in ports 1 and 2, respectively. The annihilation operators $\hat{a}_1$ and $\hat{a}_2$ are related to the creation operators by the commutation relation $[\hat{a}_i, \hat{a}_j^\dagger] = \delta_{ij}$. The creation operators $\hat{b}_1^\dagger$ and $\hat{b}_2^\dagger$ represent the creation of a photon in the output ports 1 and 2, respectively. The parameter θ is the angle of the beam splitter and determines the splitting ratio between the input modes.

The action of the beam splitter operator can be understood as follows: a photon entering the beam splitter from input mode 1 has a probability amplitude of $\cos(\theta/2)$ to exit from output mode 1 and a probability amplitude of $i\sin(\theta/2)$ to exit from output mode 2. Similarly, a photon entering the beam splitter from input mode 2 has a probability amplitude of $\cos(\theta/2)$ to exit from output mode 2 and a probability amplitude of $i\sin(\theta/2)$ to exit from output mode 1.

Therefore, the beam splitter is an optical element acting on two qumodes. Figure 3.2 depicts the graphic symbol used in quantum optical circuits for beam splitters. A mainly employed passive optical

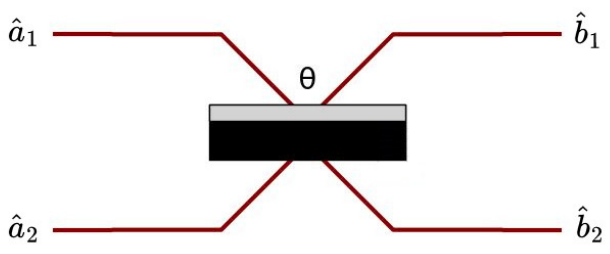


Figure 3.2: Beam Splitter gate

element is the *symmetrical beam splitter*, got when $\theta = \frac{\pi}{2}$. That implies the Eq.3.14 from Eq.3.13.

$$\begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}i \\ \frac{\sqrt{2}}{2}i & \frac{\sqrt{2}}{2} \end{pmatrix} \cdot \begin{pmatrix} \hat{a}_1 \\ \hat{a}_2 \end{pmatrix} \tag{3.14}$$

- **Detectors**: they detect the photons that exit the circuit. Single-photon detectors are typically used in QOLC, as they can detect individual photons. They must also be designed to maximise detection efficiency and minimise noise.

By appropriately combining beam splitters and phase shifters, it is possible to construct a quantum optical circuit that can simulate any unitary transformation in the Single Photon Mode case. This model is straightforward to be used for building quantum optical circuits but has the major disadvantage of very low scalability: in fact, each quantum state must necessarily correspond to a qumode, effectively eliminating the advantage of exponential computability (for example, a system with 5 qubits, with a unitary matrix of size 32x32, the equivalent optical circuit would need 32 qumodes).

Other more scalable solutions, such as the Dual Rail Mode, have other disadvantages: one of the main ones is the possibility of realising the entanglement between two photons using only linear optics due to the tendency of several photons to aggregate on the same quantum state (Pauli's exclusion principle is not valid for bosons). Entanglement is crucial to define a universal basis of gates useful in quantum computing. To this purpose, it is possible to use ancillary qumodes (called *heralds*), whose information helps filter out all the unnecessary states generated by the circuit. Also in this case, however, the number of qumodes required to implement the various CNOT gates will be high.

Over time, various solutions have been proposed to mitigate these drawbacks; taking into account that the measurement process in quantum mechanics introduces a strong non-linearity, the introduction of quantum teleportation and one-way cluster computation mechanisms into the circuit seems to be promising solutions. Also, using active nonlinear gates, which employ weak nonlinear iterations between photons (such as the Cross-Kerr gate - see paragraph 3.2.1), could open a way to manage universal quantum information. Once the photons are fed into the circuit through the source, using only intermediate linear quantum gates, because of the conservation of total energy, the number of input photons will necessarily be equal to the number of output photons. Their redistribution between individual modes and relative counting of them will determine the output state. In the case of n photons on an input port with m qumodes, a possible base output state, using finite Fock's state description, is

$$\begin{aligned}&|\psi\rangle = |k_0, k_1, \dots, k_{m-1}\rangle \\ &\text{with } k_j \in \mathbb{N} \;\; \wedge \;\; 0 \leq j \leq m-1 \;\; \wedge \;\; m \in \mathbb{N} \;\; \wedge \;\; \sum_{j=0}^{m-1} k_j = n\end{aligned} \tag{3.15}$$

The linear combination of several base states will thus give any output state. Therefore, the number of elements making up the canonical basis is granted by the formula $C_n^{m+n-1} = \binom{m+n-1}{n}$, number of n-element combinations of m objects, with repetition.

For example, in a system with 3 qumodes and 2 photons, the base states are $\binom{3+2-1}{2} = 6$ and they are

$$|2,0,0\rangle, |1,1,0\rangle, |1,0,1\rangle, |0,2,0\rangle, |0,1,1\rangle, |0,0,2\rangle$$

The states $|2,0,0\rangle, |0,2,0\rangle, |0,0,2\rangle$ should not cause consternation since bosons, unlike fermions, do not respond to the Pauli's Exclusion Principle and can therefore accumulate in large numbers on the same quantum state (or in the same mode, in the present case).

So, any quantum state in such a system is a linear combination of these six base states. Finally, by sampling the output photons using SPD (Single-Photon Detectors), it is possible to determine the probability of each base state composing the output state.

This approach is based on the *Boson Sampling* concept.

Boson Sampling

Boson Sampling is a quantum computing problem that aims to demonstrate the computational supremacy of quantum computers over classical computers. It is a particular case of the more general problem of computing the permanent of a matrix, known as #P-hard, meaning that it is challenging to solve even with the most powerful classical computers. In Boson Sampling, photons are sent through a complex network of optical devices, such as beam splitters and phase shifters, and then detected at the output [64, 65, 66, 67]. The probability distribution of the photons' final positions is calculated, which can be represented by the probability matrix. The crucial point is that this probability matrix is difficult to calculate efficiently on a classical computer, even for relatively small numbers of photons and optical devices. In contrast, a quantum computer can efficiently simulate this process using a few dozen photons and optical devices, demonstrating the potential for quantum computing to outperform classical computing in specific tasks. Boson Sampling is an essential step towards building large-scale quantum computers that can solve problems beyond the capabilities of classical computers. However, it is worth noting that it is not yet clear whether Boson Sampling can be used to solve practical problems, as the problem it solves is highly specialized and does not seem to have practical applications.

Next sections present examples of implementing quantum gates using spatial modes encoding and linear optical components:

- the entangler circuit in Single Photon Mode
- the entangler circuit in Dual Rail Mode

3.3 The *entangler circuit* in Single Photon Mode

This section will present the circuit implementation of an Entangler (Fig.2.16, Eq.2.38) using linear quantum optics. The circuit was built and tested with the Perceval software[2] [68], and the results obtained were later compared with the outcome got from a true quantum optical computer in the cloud, Quandela Cloud[3]. Consequently, for displaying the various gates in the circuit, the graphical and mathematical conventions given by the software are followed[4].

The label on the graphic symbol indicates the type of transformation operated by the beam-splitter; in the case of **Rx**, it refers to the typical type of beam splitter (Eq.3.13), with a transformation that rotates the input state by a rotation of theta angle about the X-axis in the Poincaré's sphere

[2]`https://perceval.quandela.net/docs/index.html`
[3]`https://cloud.quandela.com/webide/login`
[4]`https://perceval.quandela.net/docs/components.html`

(in optics, the equivalent representation of the Bloch's sphere for the qubits). **Ry** performs a rotation around the analogous Y axis: its transformation matrix is given by Eq.3.16. **H** performs a Hadamard's transformation with a representation matrix described by Eq. 3.17.

$$\begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \cdot \begin{pmatrix} \hat{a}_1 \\ \hat{a}_2 \end{pmatrix} \tag{3.16}$$

$$\begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \cdot \begin{pmatrix} \hat{a}_1 \\ \hat{a}_2 \end{pmatrix} \tag{3.17}$$

The unitary matrix that characterises the transformation is

$$U = CX \cdot (H \otimes X) = \begin{pmatrix} 0 & \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{pmatrix} \tag{3.18}$$

There are several methods for implementing an optical circuit, starting from a unitary matrix defining the transformation of the quantum states under examination [69, 70]. The original matrix U must first be decomposed into products of unitary block matrices (T_j) and a diagonal matrix (D), that is $U = \left(\prod_{j=1}^{m} T_j{}^{\dagger}\right) \cdot D$; the single blocks, normally matrices belonging to the SU(2) group, can be reproduced through universal transformations, such as those of the Equations 2.21, and then implemented with a series of beam splitters and phase shifters. Different optimisations are obviously attainable to improve this procedure to design an actual physical circuit [71, 72]. Suppose one uses the optical components shown in Figure 3.3 (one conventional beam splitter with variable transmissibility angle - θ - and two phase shifters with varying angles - α and β) as the basic block. In that case,

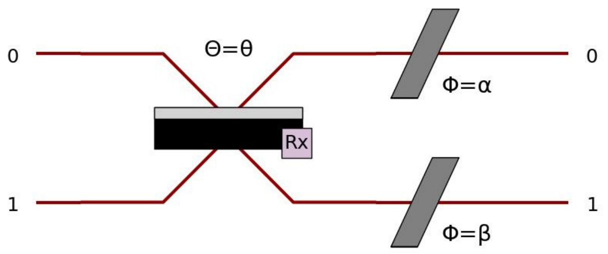


Figure 3.3: A possible base for block submatrices.

the submatrix obtained for the generic block T will be given by Eq.3.19.

$$T = \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\beta} \end{pmatrix} \cdot \begin{pmatrix} \cos\frac{\theta}{2} & i\sin\frac{\theta}{2} \\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \tag{3.19}$$

The final circuit is obtained from the composition of the basic blocks and an initial array of phase shifters, as in Fig.3.4. Furthermore, the values of the individual parameters necessary to obtain the required quantum transformation, described by Eq.3.18, are reported too.

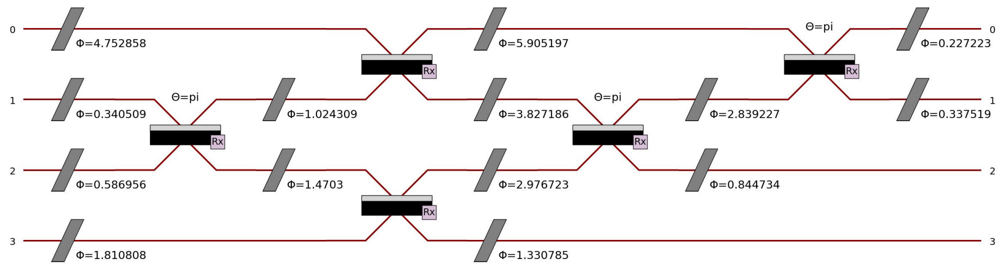


Figure 3.4: A possible optical implementation for the Entangler

Simulations in Perceval (Fig.3.5 and Fig.3.6) confirm theorical outcomes in Eq.3.20. Measurements on output states show that the probability distribution is 50% for both states $|0, 1, 0, 0\rangle$ and $|0, 0, 1, 0\rangle$.

```
Initial State: |1,0,0,0>

|0,1,0,0> (0.7071067811865476+6.7382377455516e-11j)
|0,0,1,0> (0.7071067811865475-6.738248847781846e-11j)

sqrt(2)/2*|0,1,0,0>+sqrt(2)/2*|0,0,1,0>
-----------------------------------------------------------------------------
Initial State: |0,1,0,0>

|1,0,0,0> (0.7071067811865476+1.5155876553762934e-11j)
|0,0,0,1> (0.7071067811865475-1.5155932064914165e-11j)

sqrt(2)/2*|1,0,0,0>+sqrt(2)/2*|0,0,0,1>
-----------------------------------------------------------------------------
Initial State: |0,0,1,0>

|0,1,0,0> (0.7071067811865475+6.738215541091108e-11j)
|0,0,1,0> (-0.7071067811865476+6.738265501127216e-11j)

sqrt(2)/2*|0,1,0,0>-sqrt(2)/2*|0,0,1,0>
-----------------------------------------------------------------------------
Initial State: |0,0,0,1>

|1,0,0,0> (0.7071067811865475+1.51559875760654e-11j)
|0,0,0,1> (-0.7071067811865476+1.5155876553762937e-11j)

sqrt(2)/2*|1,0,0,0>-sqrt(2)/2*|0,0,0,1>
-----------------------------------------------------------------------------
```

Figure 3.5: Results obtained by simulating the optical entangler in Perceval

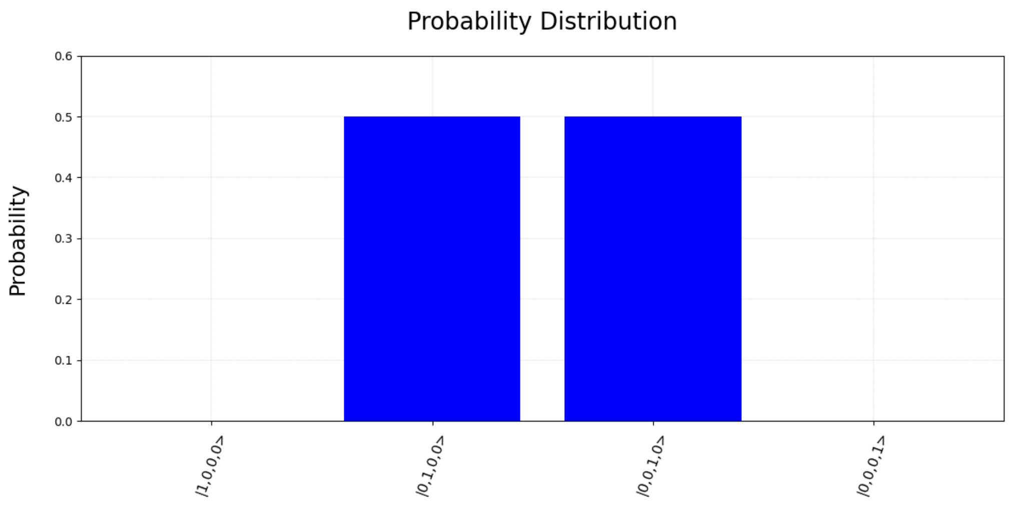


Figure 3.6: Probability obtained by simulating the optical entangler in Perceval with input state $|1, 0, 0, 0\rangle$

$$\begin{aligned}
U\,|1,0,0,0\rangle &\mapsto \frac{1}{\sqrt{2}}\,|0,1,0,0\rangle + \frac{1}{\sqrt{2}}\,|0,0,1,0\rangle \\
U\,|0,1,0,0\rangle &\mapsto \frac{1}{\sqrt{2}}\,|1,0,0,0\rangle + \frac{1}{\sqrt{2}}\,|0,0,0,1\rangle \\
U\,|0,0,1,0\rangle &\mapsto \frac{1}{\sqrt{2}}\,|0,1,0,0\rangle - \frac{1}{\sqrt{2}}\,|0,0,1,0\rangle \\
U\,|0,0,0,1\rangle &\mapsto \frac{1}{\sqrt{2}}\,|1,0,0,0\rangle - \frac{1}{\sqrt{2}}\,|0,0,0,1\rangle
\end{aligned} \tag{3.20}$$

The same circuit was subjected to a real optical quantum computer: the Ascella QPU, on Quandela's Cloud platform. One can see the presence of spurious states (Eq.3.21) due to three types of technical imperfections: the imperfection of the photon source (which can emit spurious photons on some other modes); the imperfection of the output photon detectors (which can fail the correct counting of some photons); the imperfection of the circuit's constituent elements that produce noise. The numbers on the right indicate the frequency of occurrence of the individual states (left-hand labels) following photon counting on the output detectors out of 100,000 initial samples. However, the consistency of the result with those obtained theoretically and with the simulator is perceptible: indeed, state $|0,1,0,0\rangle$ has about 45% probability on the whole sample set, while state $|0,0,1,0\rangle$ has about 55%, consistent with theoretical results.

$$
\begin{aligned}
|1,0,0,0\rangle &: \ 14\\
|0,1,0,0\rangle &: \ 107,571\\
|0,0,1,0\rangle &: \ 136,267\\
|0,0,0,1\rangle &: \ 3,086\\
|1,1,0,0\rangle &: \ 0\\
|1,0,1,0\rangle &: \ 0\\
|1,0,0,1\rangle &: \ 0\\
|0,1,1,0\rangle &: \ 26\\
|0,1,0,1\rangle &: \ 1\\
|0,0,1,1\rangle &: \ 0\\
|1,1,1,0\rangle &: \ 0\\
|1,1,0,1\rangle &: \ 0\\
|1,0,1,1\rangle &: \ 0\\
|0,1,1,1\rangle &: \ 0\\
|1,1,1,1\rangle &: \ 0
\end{aligned}
\tag{3.21}
$$

3.4 The *entangler circuit* in Dual Rail Mode

As said before, a quantum optical spatial dual rail circuit is a type of quantum circuit that operates on photons using a spatial encoding scheme called dual-rail encoding. This scheme divides a single photon into two separate spatial modes: the *signal* and *idler* modes. These two modes can be considered two separate paths. The dual rail encoding scheme helps to perform quantum computations because it allows for the encoding of quantum information in a robust way against certain types of errors. For example, the dual rail encoding can be used to encode a qubit, where the presence or absence of a photon in the signal mode represents the logical state $|0\rangle_L$, and the presence or absence of a photon in the idler mode means the logical state $|1\rangle_L$.

So, in this specific case, the state $|1,0\rangle$ has been chosen for logic state $|0\rangle_L$ and the state $|0,1\rangle$ for logic state $|1\rangle_L$. The first stage of the entangler, i.e. $H \otimes X$, can be implemented as in Fig.3.7, having got their equivalent representation utilising a beam splitter and four phase shifters to constitute gates H and X; that is possible employing the unit matrix decomposition schemes already seen in the previous section (cf. Section 3.3) and, as the basic block, the subcircuit in Fig.3.3.

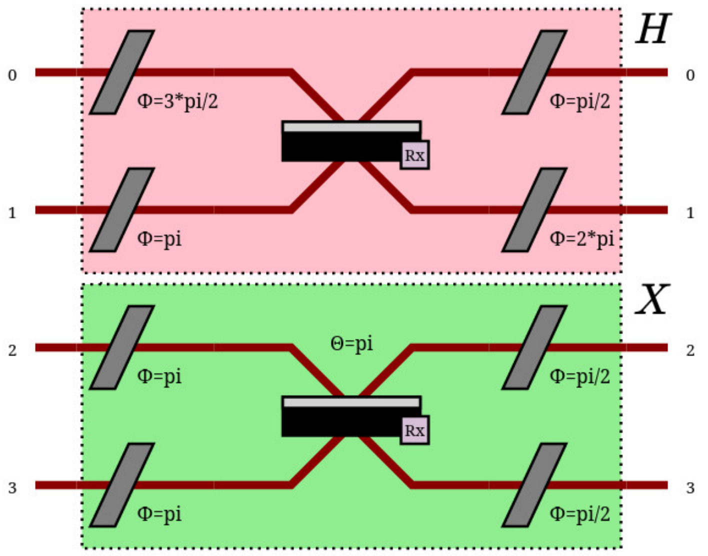


Figure 3.7: Optical equivalent gates: H (in pink) and X (in green)

Relating to the implementation of CNOT, it is essential to empathise that, in dual-rail encoding, deterministic two-qubit gates are impossible, and a chance of failure always exists. This nonsuccess can be detected in two ways: by using additional photons, known as *ancillas*, that are measured independently from the information photons to determine if the gate was successful on the information qubits (these gates are known as *heralded*); or by directly measuring the information qubits and assessing whether the gate was successful based on the result (these gates are known as *postselected*). The CNOT gate operates on two qubits (two couples of modes), a *control* and a *target*, and inverts the value of the target if the control qubit is in the logical state $|1\rangle_L$. Two types of CNOT gates occur most in the literature: the **postselected CNOT** of Ralph [73] and the **heralded CNOT** of the KLM (Knill-Laflamme-Milburn) protocol [74].

A CNOT gate of the first type, which is less sophisticated but much more straightforward to build than the other one, is sufficient to implement

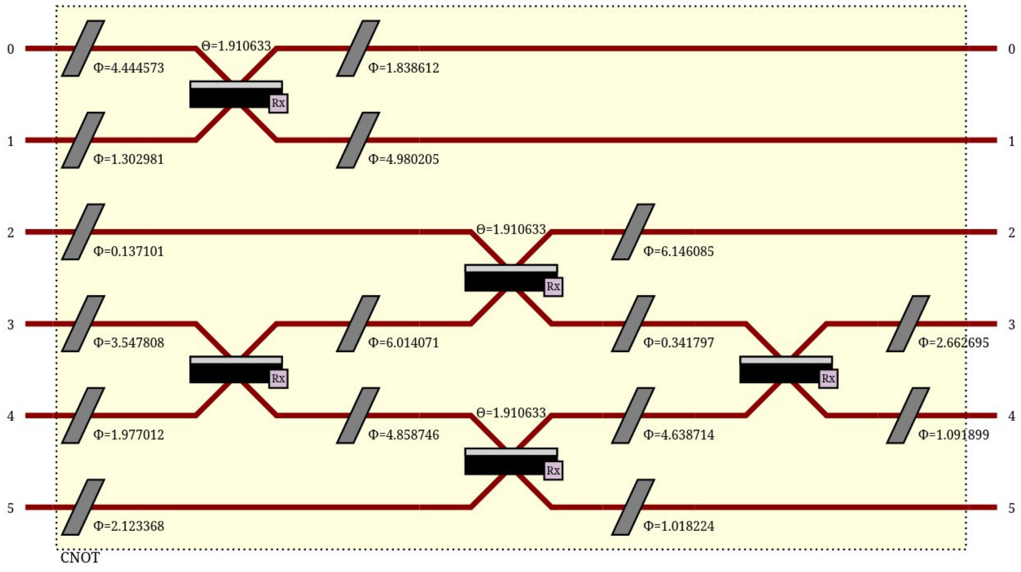


Figure 3.8: Proposed Optical CNOT schema

the proposed entangler circuit (Fig.3.8 and Fig.3.9).

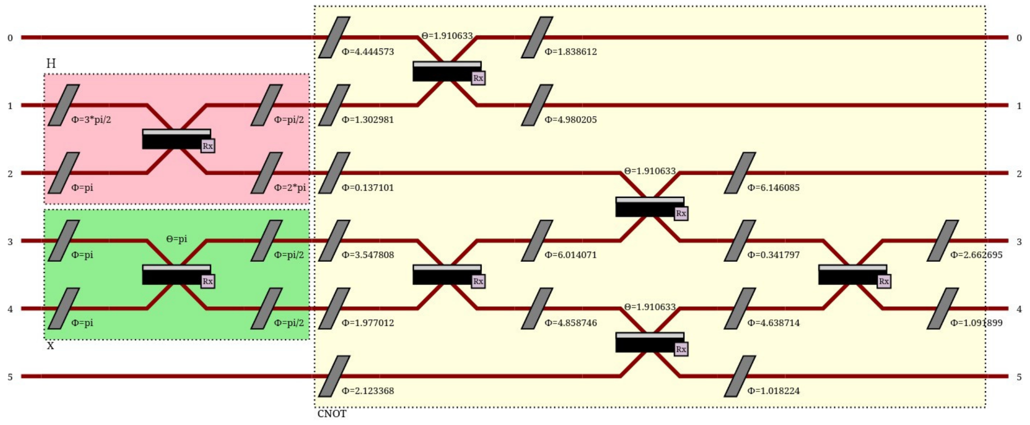


Figure 3.9: Entangler built from the two previous blocks

On the Entangler in Fig.3.9, however, it is possible to operate some significant simplifications, which help to make the physical realisation of the circuit easier and reduce noise due to unnecessary additional components. An optical Entangler like the one in Fig.3.10 is therefore proposed.

Some useful tools provided by the simulation software allow one to verify the equivalence between the circuits in Fig.3.9 and Fig.3.10: both have the

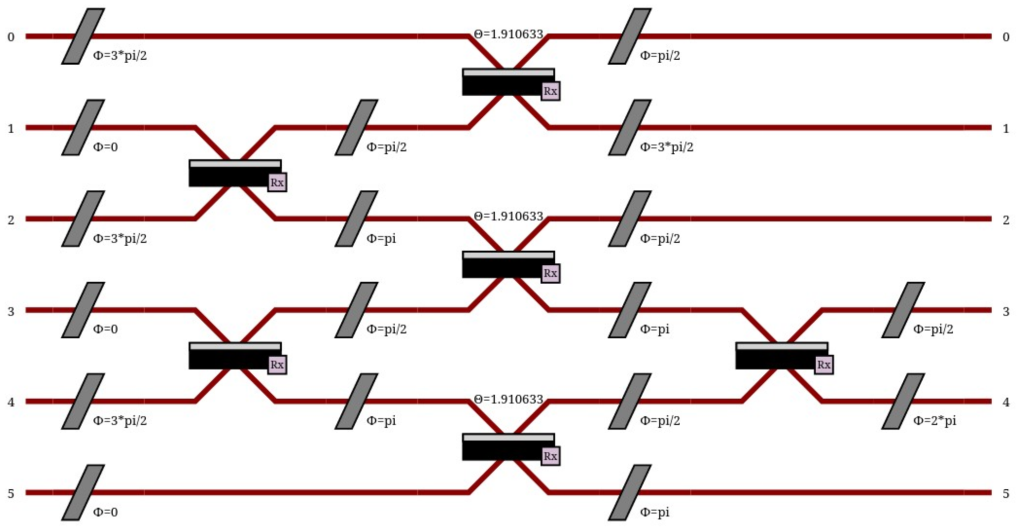


Figure 3.10: Simplified proposed Entangler

same unitary matrix (Eq.3.22).

$$U = \begin{pmatrix} \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}i}{3} & -\frac{\sqrt{3}i}{3} & 0 & 0 & 0 \\ -\frac{\sqrt{6}i}{3} & \frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{6} & 0 & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & -\frac{\sqrt{3}i}{3} & -\frac{\sqrt{3}i}{3} & 0 \\ 0 & -\frac{\sqrt{6}i}{6} & \frac{\sqrt{6}i}{6} & 0 & \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} \\ 0 & -\frac{\sqrt{6}i}{6} & \frac{\sqrt{6}i}{6} & \frac{\sqrt{3}}{3} & 0 & -\frac{\sqrt{3}}{3} \\ 0 & 0 & 0 & -\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \end{pmatrix} \tag{3.22}$$

In Fig.3.9 and Fig.3.10, modes labelled 0 and 5 are the ancillary modes. No photons are provided into them; the output state is guaranteed to be correct only if no photons are detected on the output stage of ancillary modes after a measurement process. The modes labelled 1,2 and 3,4 encode the two classical input qubits $|0\rangle \otimes |0\rangle = |00\rangle$, respectively; further, $|0\rangle$ is $|1, 0\rangle$ and $|1\rangle$ is $|0, 1\rangle$. That means the input state $|\psi_{in}\rangle$ is $|0, 1, 0, 1, 0, 0\rangle$, with

$$|\psi_{in}\rangle = |\overbrace{0}^{ancilla}, \overbrace{1,\ 0}^{qubit_1}, \overbrace{1,\ 0}^{qubit_0}, \overbrace{0}^{ancilla}\rangle$$

From Eq.3.22, it results that the annihilation operators describing the relationships between the input and the output of the circuit are related by the following expressions

$$\begin{pmatrix}\hat{b}_0\\ \hat{b}_1\\ \hat{b}_2\\ \hat{b}_3\\ \hat{b}_4\\ \hat{b}_5\end{pmatrix}=\begin{pmatrix}\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}i}{3} & -\frac{\sqrt{3}i}{3} & 0 & 0 & 0\\ -\frac{\sqrt{6}i}{3} & \frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{6} & 0 & 0 & 0\\ 0 & \frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & -\frac{\sqrt{3}i}{3} & -\frac{\sqrt{3}i}{3} & 0\\ 0 & -\frac{\sqrt{6}i}{6} & \frac{\sqrt{6}i}{6} & 0 & \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3}\\ 0 & -\frac{\sqrt{6}i}{6} & \frac{\sqrt{6}i}{6} & \frac{\sqrt{3}}{3} & 0 & -\frac{\sqrt{3}}{3}\\ 0 & 0 & 0 & -\frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3}\end{pmatrix}\cdot\begin{pmatrix}\hat{a}_0\\ \hat{a}_1\\ \hat{a}_2\\ \hat{a}_3\\ \hat{a}_4\\ \hat{a}_5\end{pmatrix} \tag{3.23}$$

$$\begin{pmatrix}\hat{a}_0^\dagger\\ \hat{a}_1^\dagger\\ \hat{a}_2^\dagger\\ \hat{a}_3^\dagger\\ \hat{a}_4^\dagger\\ \hat{a}_5^\dagger\end{pmatrix}=\begin{pmatrix}\frac{\sqrt{3}}{3} & -\frac{\sqrt{6}i}{3} & 0 & 0 & 0 & 0\\ -\frac{\sqrt{3}i}{3} & \frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{6} & -\frac{\sqrt{6}i}{6} & -\frac{\sqrt{6}i}{6} & 0\\ -\frac{\sqrt{3}i}{3} & \frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{6}i}{6} & \frac{\sqrt{6}i}{6} & 0\\ 0 & 0 & -\frac{\sqrt{3}i}{3} & 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3}\\ 0 & 0 & -\frac{\sqrt{3}i}{3} & \frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3}\\ 0 & 0 & 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3}\end{pmatrix}\cdot\begin{pmatrix}\hat{b}_0^\dagger\\ \hat{b}_1^\dagger\\ \hat{b}_2^\dagger\\ \hat{b}_3^\dagger\\ \hat{b}_4^\dagger\\ \hat{b}_5^\dagger\end{pmatrix}$$

Eqs.3.23 and 3.24 show the procedure for calculating the output state, starting from the initial state. That will allow comparing the correctness of the results obtained with the simulator and with the real optical quantum computer.

$$\begin{cases}\hat{a}_0^\dagger=\frac{\sqrt{3}}{3}\hat{b}_0^\dagger-\frac{\sqrt{6}i}{3}\hat{b}_1^\dagger\\ \hat{a}_1^\dagger=-\frac{\sqrt{3}i}{3}\hat{b}_0^\dagger+\frac{\sqrt{6}}{6}\hat{b}_1^\dagger+\frac{\sqrt{6}}{6}\hat{b}_2^\dagger-\frac{\sqrt{6}i}{6}\hat{b}_3^\dagger-\frac{\sqrt{6}i}{6}\hat{b}_4^\dagger\\ \hat{a}_2^\dagger=-\frac{\sqrt{3}i}{3}\hat{b}_0^\dagger+\frac{\sqrt{6}}{6}\hat{b}_1^\dagger-\frac{\sqrt{6}}{6}\hat{b}_2^\dagger+\frac{\sqrt{6}i}{6}\hat{b}_3^\dagger+\frac{\sqrt{6}i}{6}\hat{b}_4^\dagger\\ \hat{a}_3^\dagger=-\frac{\sqrt{3}i}{3}\hat{b}_2^\dagger+\frac{\sqrt{3}}{3}\hat{b}_4^\dagger-\frac{\sqrt{3}}{3}\hat{b}_5^\dagger\\ \hat{a}_4^\dagger=-\frac{\sqrt{3}i}{3}\hat{b}_2^\dagger+\frac{\sqrt{3}}{3}\hat{b}_3^\dagger+\frac{\sqrt{3}}{3}\hat{b}_5^\dagger\\ \hat{a}_5^\dagger=\frac{\sqrt{3}}{3}\hat{b}_3^\dagger-\frac{\sqrt{3}}{3}\hat{b}_4^\dagger-\frac{\sqrt{3}}{3}\hat{b}_5^\dagger\end{cases} \tag{3.24}$$

$$
\begin{aligned}
&|0,1,0,1,0,0\rangle = \hat{a}_1^\dagger \hat{a}_3^\dagger |0,0,0,0,0,0\rangle \\
&\hat{a}_1^\dagger \hat{a}_3^\dagger = \left(-\frac{\sqrt{3}i}{3}\hat{b}_0^\dagger + \frac{\sqrt{6}}{6}\hat{b}_1^\dagger + \frac{\sqrt{6}}{6}\hat{b}_2^\dagger - \frac{\sqrt{6}i}{6}\hat{b}_3^\dagger - \frac{\sqrt{6}i}{6}\hat{b}_4^\dagger\right)\left(-\frac{\sqrt{3}i}{3}\hat{b}_2^\dagger + \frac{\sqrt{3}}{3}\hat{b}_4^\dagger - \frac{\sqrt{3}}{3}\hat{b}_5^\dagger\right) \\
&\quad = \left(-\frac{1}{3}\hat{b}_0^\dagger\hat{b}_2^\dagger - \frac{i}{3}\hat{b}_0^\dagger\hat{b}_4^\dagger + \frac{i}{3}\hat{b}_0^\dagger\hat{b}_5^\dagger - \frac{\sqrt{2}i}{6}\hat{b}_1^\dagger\hat{b}_2^\dagger + \frac{\sqrt{2}}{6}\hat{b}_1^\dagger\hat{b}_4^\dagger - \frac{\sqrt{2}}{6}\hat{b}_1^\dagger\hat{b}_5^\dagger - \frac{\sqrt{2}i}{6}\hat{b}_2^{\dagger 2} + \right. \\
&\qquad \left. -\frac{\sqrt{2}}{6}\hat{b}_2^\dagger\hat{b}_3^\dagger - \frac{\sqrt{2}}{6}\hat{b}_2^\dagger\hat{b}_5^\dagger - \frac{\sqrt{2}i}{6}\hat{b}_3^\dagger\hat{b}_4^\dagger + \frac{\sqrt{2}i}{6}\hat{b}_3^\dagger\hat{b}_5^\dagger - \frac{\sqrt{2}i}{6}\hat{b}_4^{\dagger 2} + \frac{\sqrt{2}i}{6}\hat{b}_4^\dagger\hat{b}_5^\dagger\right) \\
&|\psi_{out}\rangle = \hat{a}_1^\dagger \hat{a}_3^\dagger |0,0,0,0,0,0\rangle = \\
&= -\frac{1}{3}b_0^\dagger b_2^\dagger |0,0,0,0,0,0\rangle \ - \frac{i}{3}b_0^\dagger b_4^\dagger |0,0,0,0,0,0\rangle \ + \frac{i}{3}b_0^\dagger b_5^\dagger |0,0,0,0,0,0\rangle \ + \\
&- \frac{\sqrt{2}i}{6}b_1^\dagger b_2^\dagger |0,0,0,0,0,0\rangle \ + \frac{\sqrt{2}}{6}b_1^\dagger b_4^\dagger |0,0,0,0,0,0\rangle \ - \frac{\sqrt{2}}{6}b_1^\dagger b_5^\dagger |0,0,0,0,0,0\rangle \ + \\
&- \frac{\sqrt{2}i}{6}b_2^{\dagger 2} |0,0,0,0,0,0\rangle \ - \frac{\sqrt{2}}{6}b_2^\dagger b_3^\dagger |0,0,0,0,0,0\rangle \ - \frac{\sqrt{2}}{6}b_2^\dagger b_5^\dagger |0,0,0,0,0,0\rangle \ + \\
&- \frac{\sqrt{2}i}{6}b_3^\dagger b_4^\dagger |0,0,0,0,0,0\rangle \ + \frac{\sqrt{2}i}{6}b_3^\dagger b_5^\dagger |0,0,0,0,0,0\rangle \ - \frac{\sqrt{2}i}{6}b_4^{\dagger 2} |0,0,0,0,0,0\rangle \ + \\
&+ \frac{\sqrt{2}i}{6}b_4^\dagger b_5^\dagger |0,0,0,0,0,0\rangle
\end{aligned}
$$

Now applying the creation operators $\hat{b}_i^\dagger$ to the appropriate mode i-th of the vector $|0,0,0,0,0,0\rangle$, we obtain the output state of the optical quantum system under consideration:

$$
\begin{aligned}
|\psi_{out}\rangle &= \hat{a}_1^\dagger \hat{a}_3^\dagger |0,0,0,0,0,0\rangle = \\
&= -\frac{1}{3}|1,0,1,0,0,0\rangle \ - \frac{i}{3}|1,0,0,0,1,0\rangle \ + \frac{i}{3}|1,0,0,0,0,1\rangle \ + \\
&- \frac{\sqrt{2}i}{6}|0,1,1,0,0,0\rangle \ + \frac{\sqrt{2}}{6}|0,1,0,0,1,0\rangle \ - \frac{\sqrt{2}}{6}|0,1,0,0,0,1\rangle \ + \\
&- \frac{i}{3}|0,0,2,0,0,0\rangle \ - \frac{\sqrt{2}}{6}|0,0,1,1,0,0\rangle \ - \frac{\sqrt{2}}{6}|0,0,1,0,0,1\rangle \ + \\
&- \frac{\sqrt{2}i}{6}|0,0,0,1,1,0\rangle \ + \frac{\sqrt{2}i}{6}|0,0,0,1,0,1\rangle \ - \frac{i}{3}|0,0,0,0,2,0\rangle \ + \\
&+ \frac{\sqrt{2}i}{6}|0,0,0,0,1,1\rangle
\end{aligned}
\tag{3.25}
$$

As said in Section 3.2.2, the space dimension describing this transformation is given by the formula $C_n^{m+n-1} = \binom{m+n-1}{n}$, where m is the number of modes in system and n is the number of photons in input; here, the dimension is $\binom{6+3-1}{3} = 21$: the components of the state vector given by Eq.3.25 have only 13 non-zero probability amplitudes, while the remaining eight are zero.

Since modes 0 and 5 are heralded (it means 0 photons in them, 0 photons expected out of them), the output state $|\psi_{out}\rangle$ collapses after a measurement process and the relative filtering step to eliminate all the states with mode 0 or mode 5 different from zero; the new output state consequently becomes $|\psi'_{out}\rangle$ (in order to improve readability, the ancillary states have been omitted).

$$\begin{aligned} |\psi'_{out}\rangle = &-\frac{\sqrt{2}i}{4}|1,1,0,0\rangle + \frac{\sqrt{2}}{4}|1,0,0,1\rangle - \frac{i}{2}|0,2,0,0\rangle \\ &- \frac{\sqrt{2}}{4}|0,1,1,0\rangle - \frac{\sqrt{2}i}{4}|0,0,1,1\rangle - \frac{i}{2}|0,0,0,2\rangle \end{aligned} \tag{3.26}$$

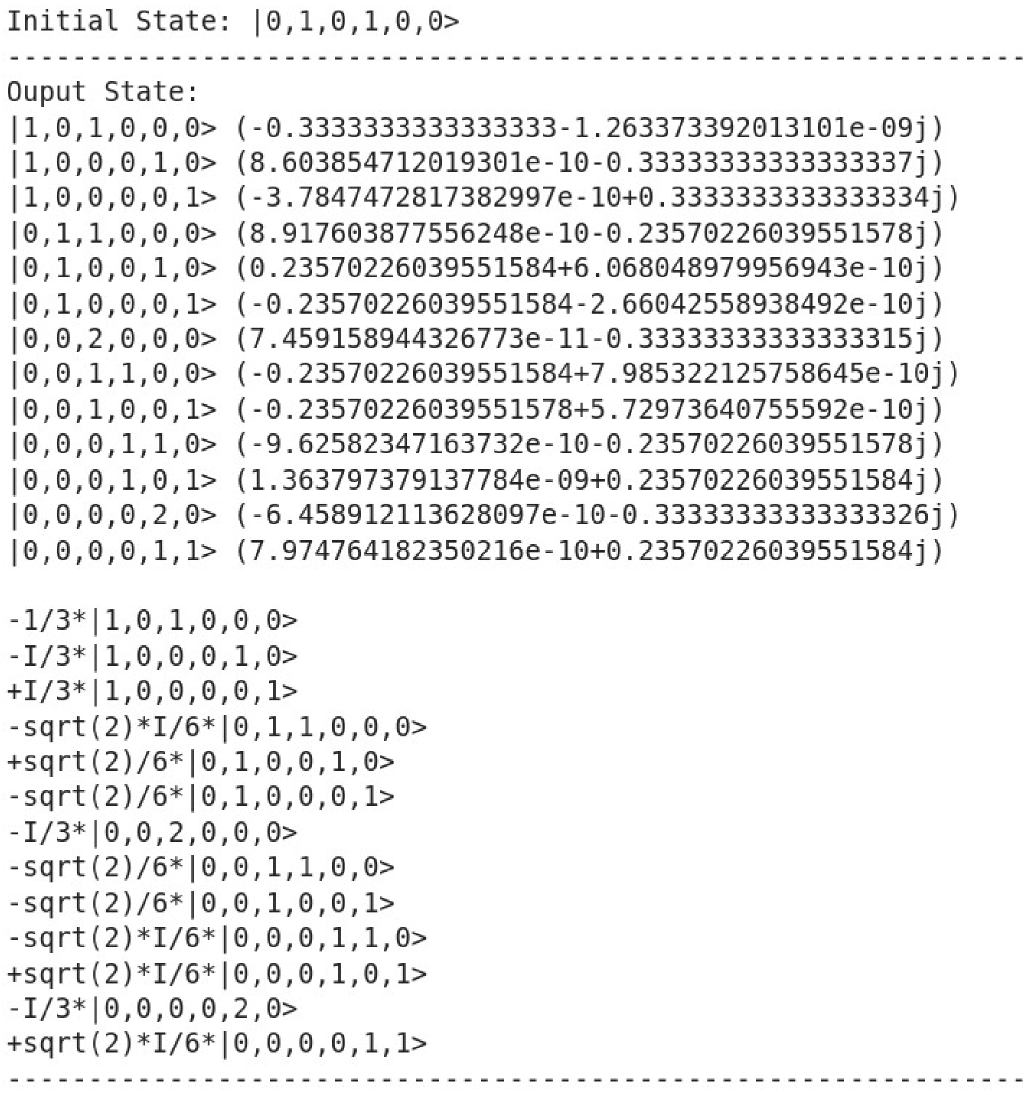
Subsequently, because the modes encoding the individual qubits can only contain a single photon alternatively, all those states that do not respect this constraint are discarded. The final state remaining after this post-selection operation, hence, results $|\psi''_{out}\rangle$.

$$|\psi''_{out}\rangle = +\frac{\sqrt{2}}{2}|1,0,0,1\rangle - \frac{\sqrt{2}}{2}|0,1,1,0\rangle \tag{3.27}$$

The outcome is correct because the final state is a Bell's state, as expected. So, output outcomes for probability distribution for the state in $|\psi''_{out}\rangle$ are $\frac{1}{2}$ for the state $|1,0,0,1\rangle$ and $\frac{1}{2}$ for the state $|0,1,1,0\rangle$. Circuit simulations with Perceval (Table 3.2, Fig.3.11, Fig.3.12 and Fig.3.13) show perfect agreement with the theoretically obtained results.

state	**probability**
$\|1,0,1,0,0,0\rangle$	0.111111
$\|2,0,0,0,0,0\rangle$	0.000000
$\|1,0,0,0,1,0\rangle$	0.111111
$\|0,2,0,0,0,0\rangle$	0.000000
$\|0,1,0,1,0,0\rangle$	0.000000
$\|0,0,0,1,0,1\rangle$	0.055556
$\|0,0,0,0,2,0\rangle$	0.111111
$\|0,1,0,0,0,1\rangle$	0.055556
$\|0,0,1,0,1,0\rangle$	0.000000
$\|0,0,0,2,0,0\rangle$	0.000000
$\|0,0,0,0,0,2\rangle$	0.000000
$\|1,1,0,0,0,0\rangle$	0.000000
$\|1,0,0,1,0,0\rangle$	0.000000
$\|1,0,0,0,0,1\rangle$	0.111111
$\|0,1,1,0,0,0\rangle$	0.055556
$\|0,1,0,0,1,0\rangle$	0.055556
$\|0,0,1,0,0,1\rangle$	0.055556
$\|0,0,0,1,1,0\rangle$	0.055556
$\|0,0,1,1,0,0\rangle$	0.055556
$\|0,0,0,0,1,1\rangle$	0.055556
$\|0,0,2,0,0,0\rangle$	0.111111

Table 3.2: Probability of each state composing final output state for the Dual Rail Entangler with input $|0,1,0,1,0,0\rangle$; results obtained with Perceval.



```
Initial State: |0,1,0,1,0,0>
------------------------------------------------------------
Ouput State:
|1,0,1,0,0,0> (-0.3333333333333333-1.263373392013101e-09j)
|1,0,0,0,1,0> (8.603854712019301e-10-0.33333333333333337j)
|1,0,0,0,0,1> (-3.7847472817382997e-10+0.3333333333333334j)
|0,1,1,0,0,0> (8.917603877556248e-10-0.23570226039551578j)
|0,1,0,0,1,0> (0.23570226039551584+6.068048979956943e-10j)
|0,1,0,0,0,1> (-0.23570226039551584-2.66042558938492e-10j)
|0,0,2,0,0,0> (7.459158944326773e-11-0.33333333333333315j)
|0,0,1,1,0,0> (-0.23570226039551584+7.985322125758645e-10j)
|0,0,1,0,0,1> (-0.23570226039551578+5.72973640755592e-10j)
|0,0,0,1,1,0> (-9.62582347163732e-10-0.23570226039551578j)
|0,0,0,1,0,1> (1.363797379137784e-09+0.23570226039551584j)
|0,0,0,0,2,0> (-6.458912113628097e-10-0.33333333333333326j)
|0,0,0,0,1,1> (7.974764182350216e-10+0.23570226039551584j)

-1/3*|1,0,1,0,0,0>
-I/3*|1,0,0,0,1,0>
+I/3*|1,0,0,0,0,1>
-sqrt(2)*I/6*|0,1,1,0,0,0>
+sqrt(2)/6*|0,1,0,0,1,0>
-sqrt(2)/6*|0,1,0,0,0,1>
-I/3*|0,0,2,0,0,0>
-sqrt(2)/6*|0,0,1,1,0,0>
-sqrt(2)/6*|0,0,1,0,0,1>
-sqrt(2)*I/6*|0,0,0,1,1,0>
+sqrt(2)*I/6*|0,0,0,1,0,1>
-I/3*|0,0,0,0,2,0>
+sqrt(2)*I/6*|0,0,0,0,1,1>
------------------------------------------------------------
```

Figure 3.11: Output state from simulation on Perceval of Dual-Rail Entangler Circuit

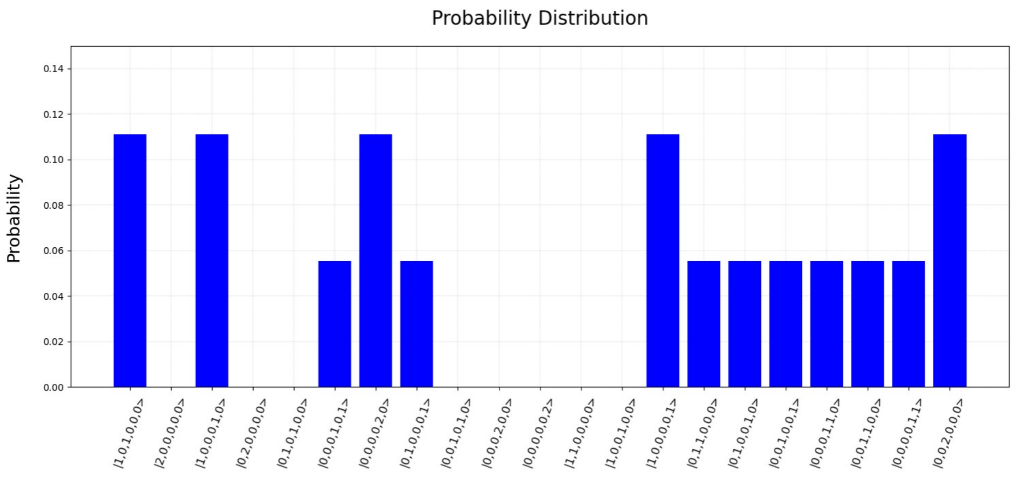


Figure 3.12: Probability distribution for state $|\psi_{out}\rangle$ on Perceval of Dual-Rail Entangler Circuit

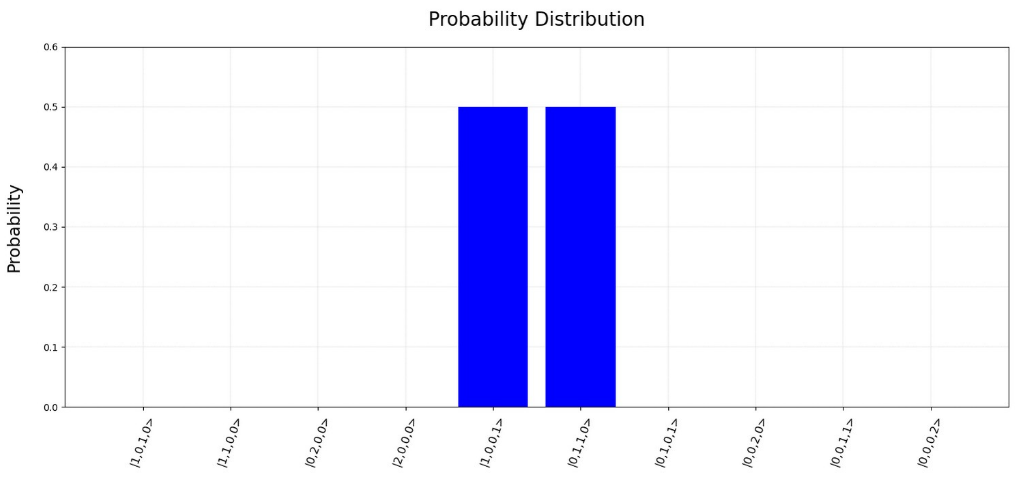


Figure 3.13: Probability distribution for state $|\psi''_{out}\rangle$ on Perceval of Dual-Rail Entangler Circuit

3.5 Conclusions

The field of quantum computing has made tremendous progress over the past few decades, with several promising platforms emerging for implementing quantum computers. Among these platforms, photons have emerged as one of the most promising candidates due to their inherent advantages in terms of coherence, manipulation, and communication. The use of photons in quantum computing is still relatively young, but it has already led to several impressive demonstrations of quantum algorithms. For example, Shor's algorithm for factoring large numbers, considered one of the most important quantum algorithms, has been demonstrated using photonic qubits. Grover's algorithm for database search, which can provide a quadratic speedup over classical algorithms, has also been presented as feasible using photonic qubits. These demonstrations have shown the potential of photonic qubits for solving problems intractable on classical computers. However, they both are still far from practical application. Yet, several challenges still exist to overcome in using photonic qubits for quantum computing. One of the main challenges is the difficulty in performing strong non-linear interactions between photons, which are necessary for implementing high-fidelity two-qubit gates. Several approaches have been proposed to address this challenge, including cavity quantum electrodynamics, quantum dots, and diamond nitrogen-vacancy centres, as enunciated in the previous chapter. Recent advances in materials science have led to the development of new types of non-linear optical materials, which are promising candidates for implementing strong non-linear interactions between photons. Another challenge in using photonic qubits for quantum computing is detecting single photons with high efficiency. Several technologies for photon detection exist, including single-photon avalanche photodiodes (SPADs) and transition-edge sensors (TESs), but each has its own strengths and weaknesses. It is essential to have reliable and efficient photon detectors to build scalable and fault-tolerant quantum computers based on photonics. Despite these challenges, there has been significant progress in photonic quantum computing in recent years. Several photonic quantum processors have been built, including the first photonic quantum computer with five qubits. These processors have been used to demonstrate a variety of quantum algorithms and have shown promising results for quantum simulation and optimization problems. One of the most exciting applications of photonic qubits is in quantum communication. Photons are already a well-established technology for quantum communication, with several commercial applications such as quantum key distribution and secure transmission. Photonic qubits have the advantage of being able to be transmitted over long distances with minimal loss of coherence, which makes them ideal for use in quantum communication networks. The development of photonic quantum repeaters, which can extend the range of quantum communication, is an active area of research. In addition to quantum computing and communication, photonic qubits are also being explored in other quantum technologies, such as quantum sensing and metrology. For example, photonic qubits can be used to perform precision measurements of electromagnetic fields, which have applications in materials science, biology, and medicine. Overall, the field of photonic quantum computing is still in its early stages, but it has already shown great promise for solving significant problems intractable on classical computers. With continued progress in materials science, photon detection technologies, and non-linear optics, photonic qubits will likely play an increasingly important role in developing scalable and fault-tolerant quantum computers.

Chapter 4

Quantum Computing - Platforms and Algorithms

4.1 Introduction

Quantum computing is a branch of computing based on the principles of quantum mechanics. While classical computers store information in binary bits that can only take on one of two values (0 or 1), quantum computers store information in quantum bits or qubits, which can take on an infinite number of values. That allows quantum computers to perform certain computations much faster than classical computers. Quantum computing has the potential to revolutionise the way to solve complex problems in various fields, such as cryptography, chemistry, physics, and machine learning. Quantum computers are designed to harness the laws of quantum mechanics, which allow them to perform certain types of computations exponentially faster than classical computers. Quantum computing platforms and quantum algorithms are the two main pillars of quantum computing.

The first part of this chapter provides an introduction to the leading quantum computing platforms and quantum algorithms. Quantum computing platforms are the hardware and software systems used to implement quantum computers. There are several different types of quantum computing platforms, each with advantages and disadvantages. The three main types of quantum computing technologies are:

- **Superconducting Quantum Computing**: Superconducting quantum computing is one of the most famous quantum computing platforms. It uses superconducting qubits made from tiny loops of superconducting wire. These qubits are extremely sensitive to external noise, so they need to be kept at very low temperatures (near absolute zero) to maintain their quantum coherence. Superconducting quantum computers typically use a dilution refrigerator to keep the qubits cold. One of the advantages of superconducting quantum computing is that it is highly scalable. Superconducting qubits can be fabricated using existing semiconductor fabrication techniques, which makes it relatively easy to manufacture large numbers of qubits. Another advantage of superconducting quantum computing is that it is relatively mature, with several companies (such as IBM and Google) already offering commercial quantum computing services based on this technology.

- **Ion Trap Quantum Computing**: Ion trap quantum computing uses trapped ions as qubits. In this approach, a laser is used to ionise atoms, which are then trapped in an electric field. The qubits are stored in the ion's electronic spin states. The advantage of ion trap quantum computing is that it is highly controllable, with individual ions manipulated using laser beams. This level of control makes performing quantum error correction possible, which is necessary to correct errors that arise due to external noise. One of the disadvantages of ion trap quantum computing is that it is difficult to scale up to large numbers of qubits. It is also challenging to fabricate ion trap qubits using existing semiconductor fabrication techniques, which makes it more expensive than superconducting qubits.

- **Photonic Quantum Computing**: Photonic quantum computing uses photons (particles of light) as qubits. In this approach, qubits are encoded in the polarisation of photons. Photonic quantum computing is attractive because photons can travel long distances without being affected by external noise, which makes it possible to perform quantum communication over long distances. However, photonic quantum computing is still in the experimental stage and is not yet mature enough for commercial applications.

Quantum algorithms are specific algorithms designed to run on quantum computers. These algorithms take advantage of the unique properties of quantum mechanics, such as superposition and entanglement, to solve problems faster than classical algorithms. One of the most famous quantum algorithms is Shor's algorithm, which can factor large numbers exponentially faster than any known classical algorithm. That has significant implications for cryptography because many encryption schemes rely on the difficulty of factoring large numbers. Shor's algorithm can break these encryption schemes, making quantum computers potentially threatening current encryption methods. Another important quantum algorithm is Grover's algorithm, which can perform an unstructured search of a database of N items in $O(\sqrt{N})$ time, compared to $O(N)$ time for classical search algorithms. This algorithm has implications for data mining, optimisation, and machine learning. Quantum algorithms can be classified into three categories: quantum simulation, optimisation, and error-correcting algorithms. Quantum simulation algorithms are used to simulate quantum systems, which are difficult to simulate using classical computers. These algorithms are used in fields such as quantum chemistry and materials science. For example, the *Variational Quantum Eigensolver* (VQE) algorithm can approximate the ground state energy of molecules, which is helpful in designing new drugs and materials. Quantum optimisation algorithms are used to solve optimisation problems, ubiquitous in various fields such as logistics, scheduling, and finance. One of the most famous quantum optimisation algorithms is the *Quantum Approximate Optimization Algorithm* (QAOA), which can be used to find the optimal solution to a combinatorial optimisation problem. Quantum error-correcting algorithms are used to correct errors in quantum computers due to noise and decoherence. These algorithms are crucial for building large-scale, fault-tolerant quantum computers. One of the most famous quantum error-correcting codes is the surface code, which encodes qubits in a two-dimensional grid of physical qubits and can correct errors using measurements on neighbouring qubits. Quantum algorithms are implemented using quantum circuits, which are sequences of quantum gates that perform operations on qubits. Qubits are the quantum equivalent of classical bits, and they can be in a superposition of states, which means they can be in multiple states simultaneously. This property allows quantum computers to perform certain computations faster than classical computers. Quantum algorithms also require a certain number of qubits to perform the computation. The number of qubits needed depends on the problem size and the algorithm's complexity. Currently, quantum computers with a few hundred qubits are available, and they can perform some simple quantum algorithms. However, building large-scale, fault-tolerant quantum computers remains a significant challenge.

4.2 Quantum Computing insights

After years of solid theoretical work to appropriately approach the domain of quantum computation, despite several breakthroughs in the fields of nanoelectronics and photonics, quantum computer that is reliable, scalable, stable, resilient to environmental noise, and general-purpose has not yet attained maturity. Indeed, as previously said, quantum computing is a very new field; we are thus in a historical period quite comparable to that of the first explorers of the New World. We are in a historical phase of transition, although studded with intense research, named by Preskill NISQ (Noisy Intermediate-Scale Quantum) era [75].

Von Neumann's architectural concept serves as the foundation for contemporary computing, which is not entirely adaptable to the new paradigm of quantum computation; translating the entire range of classical digital circuits is a possible (and desirable to overcome Moore's law) operation, but not without pain, due to the various quantum effects to be managed and the enormous number of qubits required even for simple combinational logic circuits.

There is the tendency to think of computers as machines capable of manipulating enormous amounts of data in a reasonable time, beyond the semantics of the data itself; but they remain fundamentally and intimately fast logic-arithmetic computing machines. Furthermore, their peculiar architecture gives them a great lot of flexibility in dealing with a wide range of challenges. This chapter will be centred on general-purpose quantum computers rather than adiabatic quantum computers such as D-Wave [76], which are tailored to address specific optimisation problems featuring a high volume of qubits. The current metric that allows to classify a quantum computer as general-purpose is based on Di Vincenzo's five criteria [77]; they state that a quantum computer should:

1. be feasible and physically realised through a scalable technology and in which the qubits are easy to be identified and not too difficult to be managed;
2. be able to return to a predetermined fiducial state; typically, the $\bigotimes_{j=1}^{n} |0\rangle$ state is chosen;
3. have long enough decoherence periods to allow the single quantum gates that make up the circuit to carry out the transformation operations; in other words, qubit decoherence periods have to be longer than gate action times;
4. possess an universal base set of gates, so that any sequence of unitary transformations may be performed;
5. be able to undertake a precise and defined qubit measurement process that permits it to be collapsed onto one of the computational base's components.

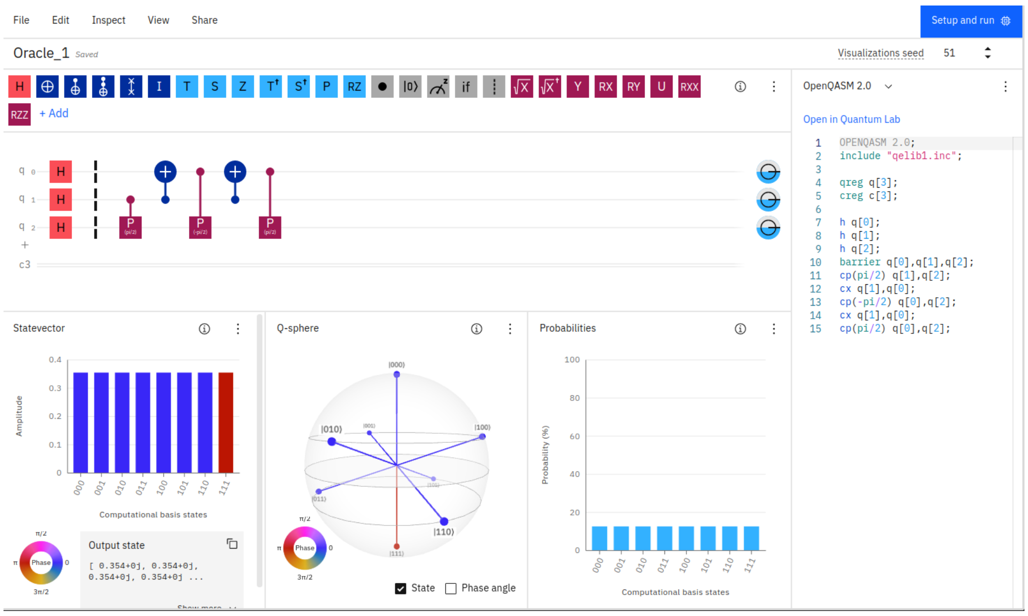
After understanding the current scientific advancements in this field, it can be beneficial to apply our theoretical knowledge by utilising the available means from IT giants, public and private institutions, and individual enthusiasts.

Many of them provide a variety of analysis and synthesis tools for developing a quantum algorithm; some are sophisticated simulators, while others are cloud services that allow immediate access to a real quantum computer after registration. Numerous options are available for developing quantum algorithms, ranging from analysis and synthesis tools to advanced simulators and cloud services that provide real quantum computers upon registration. Simulators can adapt to any scheme and optimise suggested code, but it is best to have an expert oversee the use of a real quantum computer for complex algorithms.

The main frameworks and/or programming environments for exploring quantum architectures are now presented.

IBM’s QExperience and Qiskit

It is an online platform that enables quantum computation on quantum architectures connected to the Cloud. However, before implementing the generated circuit, it is necessary to select which backend to use among those offered. The project’s execution will be queued with the other tasks requested by other users. It is also possible to instantiatly run multiple simulations



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];

h q[0];
h q[1];
h q[2];
barrier q[0],q[1],q[2];
cp(pi/2) q[1],q[2];
cx q[1],q[0];
cp(-pi/2) q[0],q[2];
cx q[1],q[0];
cp(pi/2) q[0],q[2];
```

Output state

```
[ 0.354+0j, 0.354+0j,
0.354+0j, 0.354+0j ...
```

Figure 4.1: IBM quantum composer example

by downloading a dedicated Python library, initially released by IBM but now Open-Source, called Qiskit[1]. Programming is very intuitive and takes place thanks to a tool called Composer, a fully graphical tool, consisting of an area of five lines, each reserved for a qubit, as shown in Figure 4.1. Obviously, quantum logical operators can be inserted within the composer, which can be selected from a *palette* and placed in our circuits through the action of Drag and Drop; at the end of each line there must be the operator that allows the measurement of the quantum state of the qubit corresponding to the line, storing the result on a classical bit representation. The results, once obtained, will be represented by means of a histogram.

[1] https://quantum-computing.ibm.com/, https://qiskit.org/

IBM Quantum Experience also provides a text-level programming mode that takes advantage of the QASM language, very similar to C.

IBM provides users with five quantum simulators with different characteristics. The first is the Clifford simulator (stabiliser), which makes it possible to simulate Clifford circuits with and without noise. The second is the Matrix Product State (MPS), which simulates tensor networks. The third is the Extended Clifford (extended_stabilizer), which extends the functionality of the stabiliser simulator. The Extended Stabilizer technique is divided into two main sections: the former is a method for breaking down quantum circuits into stabilizer circuits, a type of circuit that can be efficiently replicated classically, the latter is a method of integrating these circuits in order to conduct measurements. The fourth is the general purpose (Qasm_simulator), which allows general purpose simulations to be carried out and is the backend that is provided by default for theoretical simulation. The fifth is the Schrödinger wavefunction (statevector), which simulates a quantum circuit by calculating the wavefunction of the state vector

IBM also provides 22 real quantum computers, of which 8 are free, and 14 are chargeable. These computers can be grouped into various categories according to the technology used by the quantum processors: Falcon, Egret, Hummingbird, Eagle, Osprey. The Falcon family consists of tiny designs having 5 to 27 qubits. All qubits and readout resonators are on the same layer of the optimised 2D lattice. The Falcon family of devices provides a key platform for medium-scale circuits, as well as a good platform for proving performance and scalability enhancements prior to pushing them onto bigger devices. The Hummingbird is a new series of quantum processor that supports up to 65 qubits using a heavy-hexagonal qubit architecture. More powerful machines are the quantum chips Eagle (up to 127 qubits) and the Osprey (433 qubits).

Google's Cirq and Tensorflow Quantum

Cirq is a Python software library that allows us to write, manipulate, and optimise quantum circuits and to run them on quantum computers and simulators. Cirq[2] tries to highlight the specific characteristics of the hardware, since in the Noisy Intermediate-Scale Quantum regime (also known as NISQ) these peculiarities determine the possibility or not of running a simulation on a given circuit.

Research in quantum machine learning examines the interaction between quantum computing and machine learning. According to current thinking, quantum computers can be utilised and trained in the same way that neural networks are used. Physical control factors such as electromagnetic field strength or laser pulse frequency can be adjusted in an organised manner in order to address a problem. TensorFlow Quantum (TFQ)[3] is a quantum machine learning library enabling quick prototyping of hybrid quantum-classical ML models. Cirq and TensorFlow Quantum are both extremely versatile and feature-rich, and are in great demand. They are also intuitive and simple to use.

Xanadu's Strawberry Fields and Pennylane

The open-source software PennyLane[4] is based on the notion of quantum differentiable programming and is free to download and use. Classical machine-learning libraries are easily integrated with quantum simulators and hardware, allowing the user to train quantum circuits. PennyLane's primary task is to manage the assessment of parametrized quantum circuits (also known as variational circuits) on quantum devices and to make them accessible to machine learning libraries. PennyLane also gives access to quantum circuit gradients, which the machine learning library can utilise to execute back-propagation, including through quantum circuits an important process in optimisation and machine learning. One promising technology for physically realising a quantum computer is photonic processors. This technology uses the modes of photons (qumodes) to store information. Many companies are working on this new type of machine. Xanadu makes a programming environment called Strawberry fields available to users after registering on the website (Fig.4.2).

[2]Cirq official website: `https://quantumai.google/cirq`
[3]TensorFlow Quantum official website: `https://www.tensorflow.org/quantum`
[4]PennyLane official website: `https://pennylane.ai/`

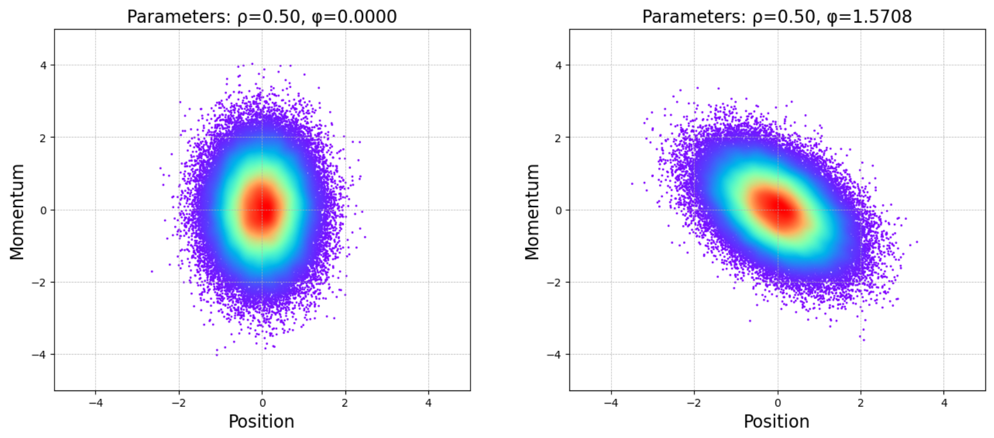


Figure 4.2: StrawberryFields Python's library example

Microsoft's Q# and Azure Quantum

Microsoft Azure[5] is a cloud-based service that developers, academics, and businesses may utilise to execute quantum computing applications or solve optimisation issues. Q#, Microsoft's open source programming language for building quantum algorithms, is included in the Microsoft Quantum Development Kit (QDK). It also includes Q# libraries, Python and .NET APIs, and a Python SDK for optimisation solvers and quantum simulators. The QDK is a full-featured development kit for Q# that can be used in conjunction with conventional tools and languages to create quantum applications that can be executed in a variety of settings. Q# applications can be executed as a console application, in Jupyter Notebooks, or through a Python or .NET host program. The Q# libraries allow you to do sophisticated quantum operations without the need to create low-level operation sequences.

The QDK provides numerous quantum machine classes, which are all specified under the namespace Microsoft.Quantum.Simulation.Simulators. The first is the *Full State Simulator*,which replicates a quantum machine on your local computer. The full state simulator may be used to execute and debug quantum algorithms developed in Q# and allow you to use up to 30 qubits. The second one is the *Simple Resources Estimator*, it can estimate

[5]Microsoft's Q# official website: https://azure.microsoft.com/en-us/services/quantum/

resources for Q# operations that utilise thousands of qubits, as long as the classical component of the code executes in an acceptable amount of time. The third one is the *Trace-based resource estimator* which executes a quantum program without really mimicking a quantum computer's state. As a result, the quantum trace simulator can run quantum algorithms with thousands of qubits. It help the developer to debug classical code that is a component of a quantum program, moreover calculating the resources needed to run a specific instance of a quantum program on a quantum computer, the trace simulator serves as the foundation for the Resources estimator, which gives a more limited set of indicators. The fourth one is the *Toffoli simulator*, which is a limited-scope special-purpose simulator that only supports X, CNOT, and multi-controlled X quantum operations. There is access to all classical logic and calculations; it is not so powerful and effective as the Full State Simulator.

Amazon's AWS Braket

According to the AWS[6] concept, Amazon Braket is a fully managed quantum computing service. Amazon Braket provides a development environment for experimenting and developing quantum algorithms, as well as testing them on quantum circuit simulators and running them on different quantum hardware technologies. To design quantum algorithms and manage experiments, it may be used his/her own development environment or fully managed Jupyter notebooks in Amazon Braket. There are four circuit simulators available with Amazon Braket to perform and evaluate quantum algorithms. The first one is a free local simulator that may be used on your laptop or on an Amazon Braket managed notebook. Depending on your hardware, the local simulator is well suited for conducting small and medium-scale simulations up to 25 qubits without noise or up to 12 qubits with noise. The second one is SV1, a full managed *State Vector* simulator, which calculates the outcome by taking the complete wave function of the quantum state and applying the circuit's operations. You can use SV1 to test your quantum method at scale after designing it in the Amazon Braket SDK using the local simulator, and then

[6]AWS bracker official website: https://aws.amazon.com/braket/

utilising SV1's parallel processing to perform several batches of simulations. SV1 can handle simulations of up to 34 qubits. The third one is DM1, is a fully managed and high-performance simulator that simulates the effects of noise on quantum circuits using *Density Matrix* computations. You may define realistic error models for your circuits and investigate the influence of noise on your algorithms using DM1. DM1 allows you to simulate circuits with up to 17 qubits. The last one is TN1 a *Tensor Network* simulator for structured quantum circuits is a fully managed, high-performance tensor network simulator. A tensor network simulator converts quantum circuits into a structured graph in order to determine the most efficient approach to calculate the circuit's result. You can use TN1 to simulate certain types of quantum circuits up to 50 qubits in size.

Rigetti's Forest

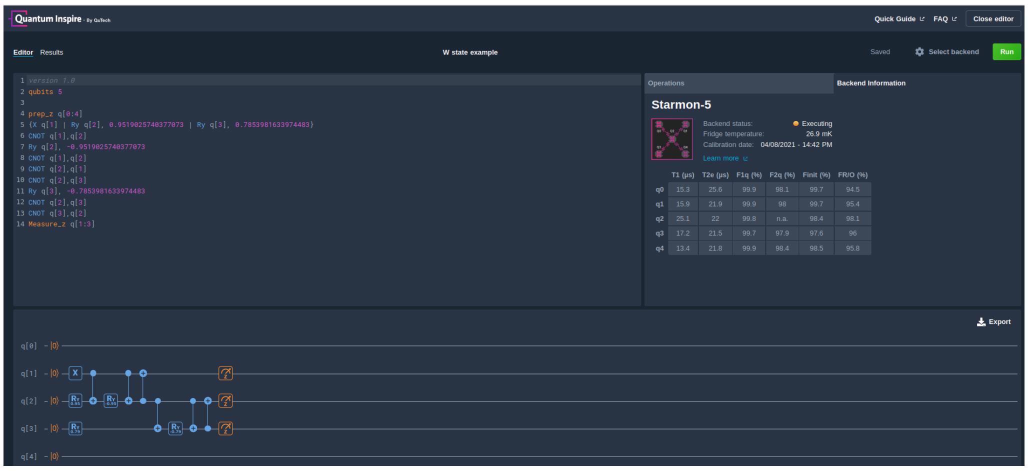
Forest[7] is a cloud computing platform that provides developers with access to quantum processors so they may test quantum algorithms and simulate them by using a quantum device with more than 32 qubits. It is built on a specialised instruction language known as Quil (Quantum Instruction Language), which can enable a form of hybrid computing (simultaneous use of quantum components and classical logic). Moreover, applications may be developed and run using free and open source Python tools.

Quantum Inspire

QuTech's Quantum Inspire (QI)[8] is a quantum computing platform. The objective of Quantum Inspire is to give users access to various tools for doing quantum calculations, as well as insights into quantum computing concepts and access to the community. QuTech, the advanced research center for quantum computing and quantum internet created by TU Delft and TNO, started Quantum Inspire (QI). Quantum Inspire gives users a range of options for programming quantum algorithms, running them, and analyzing the results (Fig.4.3). It gives you a graphical interface for programming

[7]Righetti forest official website: https://www.rigetti.com/

[8]QuTech. (2020). Quantum Inspire Home. Retrieved from Quantum Inspire: https://www.quantum-inspire.com/



```
version 1.0
qubits 5

prep_z q[0:4]
{X q[1] | Ry q[2], 0.9519025740377073 | Ry q[3], 0.7853981633974483}
CNOT q[1],q[2]
Ry q[2], -0.9519025740377073
CNOT q[1],q[2]
CNOT q[2],q[1]
CNOT q[2],q[3]
Ry q[3], -0.7853981633974483
CNOT q[2],q[3]
CNOT q[3],q[2]
Measure_z q[1:3]
```

Backend status: Executing
Fridge temperature: 26.9 mK
Calibration date: 04/08/2021 - 14:42 PM

	T1 (μs)	T2e (μs)	F1q (%)	F2q (%)	Finit (%)	FR/O (%)
q0	15.3	25.6	99.9	98.1	99.7	94.5
q1	15.9	21.9	99.9	98	99.7	95.4
q2	25.1	22	99.8	n.a.	98.4	98.1
q3	17.2	21.5	99.7	97.9	97.6	96
q4	13.4	21.8	99.9	98.4	98.5	95.8

Figure 4.3: Quantum Inspire composer example

in QASM (Quantum Assembly Language) and visualizing operations in circuit diagrams. Non-quantum specialists may learn to write quantum algorithms with the QI Editor, which includes automated bug detection and autocomplete. Quantum Inspire has three account types: Anonymous, Basic, and Advanced. You can create some simple algorithms and utilize up to 5 qubits in the QX simulator as an anonymous user. A basic account allows you to simulate quantum algorithms with up to 26 qubits and run them on our hardware back-ends. With an advanced account, you may simulate up to 31 qubits on Cartesius, one of the Dutch supercomputers at SURFsara, utilizing more powerful computing resources.

Quirk

Quirk[9] is a quantum circuit simulator that allows you to manipulate and explore tiny quantum circuits by dragging and dropping components (Fig.4.4). The visual design of Quirk offers a very intuitive sense of what is going on, state displays update in real time as you modify the circuit, and the whole experience is quick and engaging. Using Quirk consists primarily of dragging gates from the toolboxes, putting them into the circuit, and inspecting the

[9]A live version of Quirk may be found at https://algassert.com/quirk, but you can also obtain the source code from https://www.github.com/Strilanc/Quirk and create your own.

state displays within and to the right of the circuit. As you change the circuit, Quirk actively rewrites the URL in the address bar to refer to the current circuit. You may also save the circuit by clicking the Export button above the circuit editing box. You can export an escaping link, an offline copy of Quirk defaulting to the current circuit, a JSON representation of the current circuit, or a JSON representation of the full simulator output. Quirk is a free and open source software application. The source code is accessible under a permissive Apache license, allowing anybody to create and distribute updated versions.

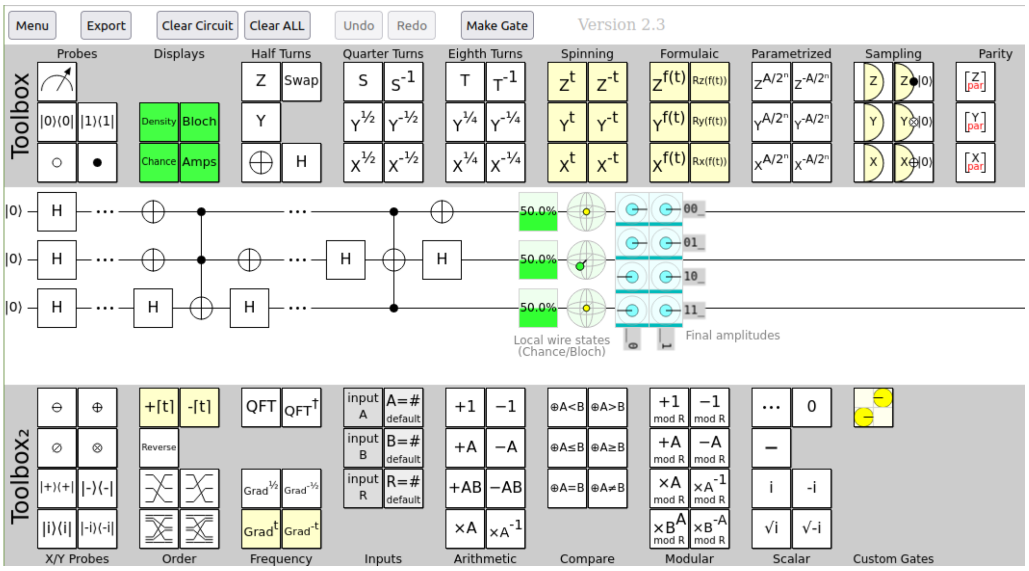


Figure 4.4: Quirk quantum composer example

Quandela's Perceval and Cloud

An exciting product for modelling photonic circuits, according to the discrete variable paradigm (DV model), is Perceval[10]. Perceval offers tools for creating photonic circuits using linear optical components like beamsplitters and phase shifters, defining single-photon sources, working with Fock states, and performing simulations using a straightforward object-oriented Python API. Perceval can be used to replicate previously published experimental results or conduct direct experiments using a fresh wave of quantum algorithms. It

[10]Perceval by Quandela: https://perceval.quandela.net/

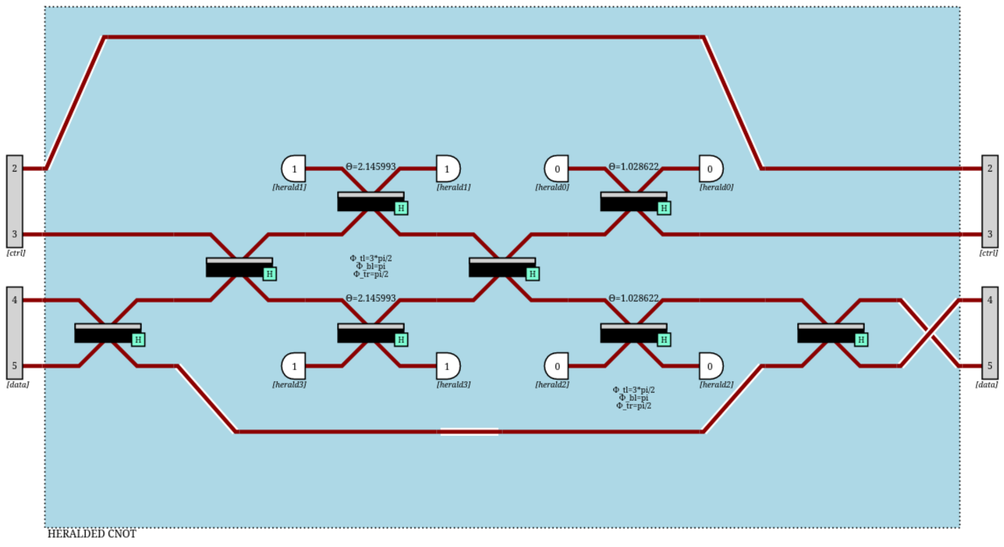


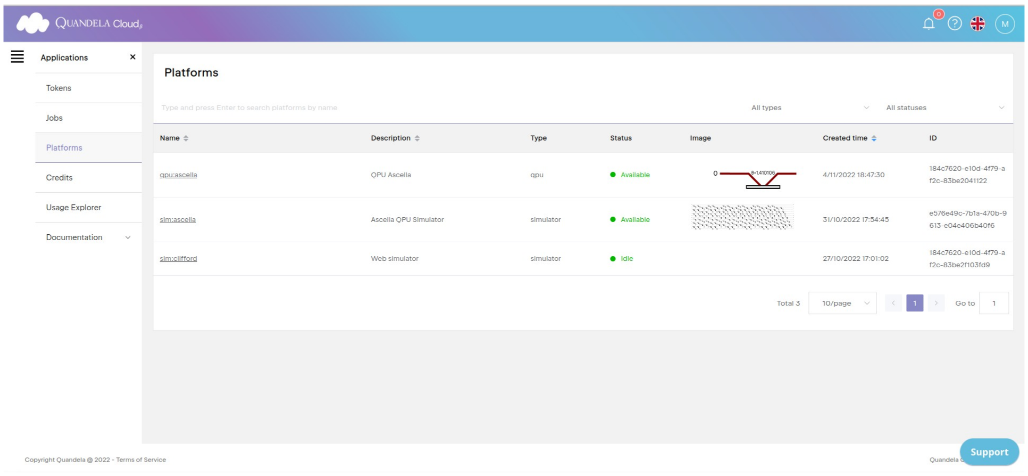
Figure 4.5: Perceval Python's Framework example

seeks to serve as a support tool for creating photonic circuits by modelling and improving their design, simulating both the ideal and practical behaviours, and offering a standardised interface to operate them using the backends. Fock states, state vectors, and state vector distributions, as well as unitary matrices and parameters, are just a few of the technical tools that may be used to manage Perceval. It can be regarded as a good tool for students, researchers and scholars in the field of photon quantum computing. Additionally, it has cross-platform tools for adaptable viewing of the circuits and outcomes that are compatible with notebooks and local scripting usage (Fig.4.5).

Alongside Perceval, the French company Quandela[11] also provides the user and researcher with an online platform with a real photonic quantum computer (a *12-mode* photonic quantum computer[12], called "Ascella"), which allows for assessing the consistency of theoretical results with those obtained experimentally. Among the platform's most significant merits are the continuous-time availability of the QPU (Quantum Processing Unit) throughout the day, the well-maintained and intuitive user interface, and the user-friendliness of the various functions offered (Fig.4.6 and Fig.4.7).

[11]https://www.quandela.com/

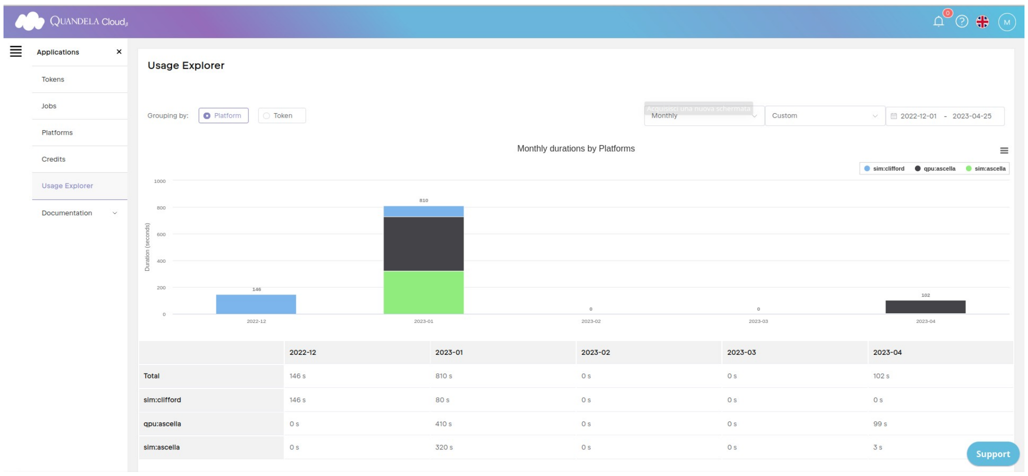
[12]https://cloud.quandela.com/webide/login



Platforms

Name	Description	Type	Status	Image	Created time	ID
qpu:ascella	QPU Ascella	qpu	Available		4/11/2022 18:47:30	184c7620-e10d-4f79-af2c-83be2041122
sim:ascella	Ascella QPU Simulator	simulator	Available		31/10/2022 17:54:45	e576e49c-7b1a-470b-9613-e04e406b40f6
sim:clifford	Web simulator	simulator	Idle		27/10/2022 17:01:02	184c7620-e10d-4f79-af2c-83be2f103fd9

Figure 4.6: Quandela Cloud: various available platforms to run optical quantum algorithms



Usage Explorer

Grouping by: [✓ Platform] Token

Monthly — Custom — 2022-12-01 - 2023-04-25

	2022-12	2023-01	2023-02	2023-03	2023-04
Total	146 s	810 s	0 s	0 s	102 s
sim:clifford	146 s	80 s	0 s	0 s	0 s
qpu:ascella	0 s	410 s	0 s	0 s	99 s
sim:ascella	0 s	320 s	0 s	0 s	3 s

Figure 4.7: Quandela Cloud: usage of single platforms

4.3 Relevant Quantum Algorithms

This section contains some important algorithms for quantum computing that will have a revolutionary impact on the world of computer applications in the next few years.

4.3.1 The Deutsch–Jozsa algorithm

In the Deutsch–Jozsa problem [78], the goal is to establish if an input unknown function has certain characteristics; this function, $f : \{0,1\}^n \rightarrow \{0,1\}$, with $n \in \mathbb{N}$, which works as a black block is called *oracle*. This function can be either *constant*, when all inputs are 0 or 1, or *balanced*, if it has the value 1 for half of the input domain and 0 for the other half. The goal is to use the oracle in order to see whether f is constant or balanced.

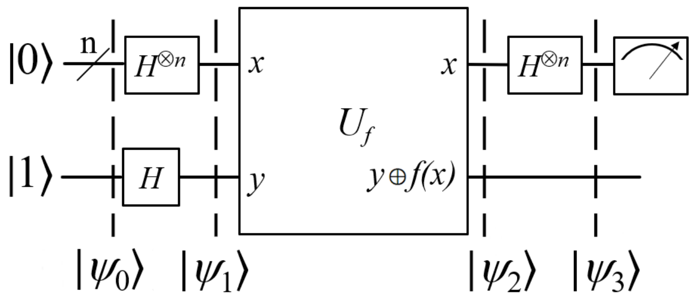


Figure 4.8: Deutsch-Jozsa quantum circuit to implement the algorithm

In the worst case, $2^{n-1} + 1$ evaluations of f would be needed for a standard deterministic algorithm, where n is the number of bits, whereas the Deutsch-Jozsa quantum algorithm can always generate a right outcome, taking a single assessment step.
In Figure 4.8 it is shown the circuit to implement the Deutsch–Josza algorithm.

In Algorithm 2, the procedure is shown in more details.

Therefore, if *f(x)* is *constant* (*constructive interference*), the probability is 1; if *f(x)* is *balanced*, it evaluates to 0 (*destructive interference*). In other words, if measurement result is the state $|00...00\rangle$ then *f(x)* is *constant*; any other state means *f(x)* is *balanced* and all this in *only one shot*.

4.3.2 The Grover's algorithm

Grover's algorithm [79], also known as the quantum search algorithm, is a quantum algorithm for unsorted search on a dataset that uses only $O(\sqrt{N})$

Algorithm 2 Deutsch-Jozsa algorithm. Part 1

```
1: In the first stage, the input state is prepared
   $$|\psi_0\rangle = \bigotimes_{i=0}^{n-1} |0\rangle \otimes |1\rangle = |0\rangle^{\otimes n}|1\rangle = |00...01\rangle$$
2: The Hadamard transform is applied to each qubit of $|\psi_0\rangle$
   $$\begin{aligned}|\psi_1\rangle &= \bigotimes_{i=0}^{n-1} H|0\rangle \otimes H|1\rangle = \\ &= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)^{\otimes n} \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \left(|0\rangle - |1\rangle\right)\end{aligned}$$
3: The $f$ function has been introduced as a quantum oracle. The oracle converts the state $|x\rangle|y\rangle$ to $|x\rangle|y \oplus f(x)\rangle$ ($\oplus$ is the addition operation in modulo 2 arithmetic). When the quantum oracle is used, the result is:
   $$\begin{aligned}|\psi_2\rangle &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \left(|f(x)\rangle - |1 \oplus f(x)\rangle\right) = \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}|x\rangle \left(|0\rangle - |1\rangle\right)\end{aligned}$$
   as $f : \{0,1\}^n \rightarrow \{0,1\}$ can only get values either 0 or 1.
```

Algorithm 2 Deutsch-Jozsa algorithm. Part 2

```
4: Let $|\widetilde{\psi_2}\rangle$ be such that $|\psi_2\rangle = |\widetilde{\psi_2}\rangle \otimes |y \oplus f(x)\rangle$ and $|\widetilde{\psi_3}\rangle$ such that $|\psi_3\rangle = |\widetilde{\psi_3}\rangle \otimes |y \oplus f(x)\rangle$:
   $$|\widetilde{\psi_2}\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \ \wedge \ |\widetilde{\psi_3}\rangle = H^{\otimes n} |\widetilde{\psi_2}\rangle \ \Rightarrow$$
   $$\Rightarrow |\widetilde{\psi_3}\rangle = \frac{1}{2^n} \sum_{x,y=0}^{2^n-1} (-1)^{(f(x)-x\cdot y)} |y\rangle =$$
   $$= \frac{1}{2^n} \sum_{y=0}^{2^n-1} \left( \sum_{x=0}^{2^n-1} (-1)^{f(x)} \cdot (-1)^{x\cdot y} \right) |y\rangle$$
   with $x \cdot y = \bigoplus_{i=0}^{n-1} x_i \cdot y_i$
   This means that
   $$\langle \widetilde{\psi_3} \, |\Psi\rangle\langle\Psi| \, \widetilde{\psi_3}\rangle = \left| \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2, \qquad with \ \ |\Psi\rangle = \bigotimes_{i=0}^{n-1} |0\rangle$$
   which is the probability for the state $|\, \widetilde{\psi_3}\rangle$ on the eigenstate $|\Psi\rangle$.
```

function evaluations to find the unique input to a black box function that generates the correct output value with a high probability; $N = 2^n$ is the number of item in our dataset, and n is the amount of needed information to represent the whole set. The criterion for the search is implemented in a previously defined function, called *oracle*; this function returns whether or not the input number meets the search criteria.
A common form for this function is

$$f(x) = \begin{cases} 1 & if \ \ x = \omega \\ 0 & if \ \ x \neq \omega \end{cases}$$

where ω is the solution for our problem.

In Algorithm 3, the procedure is shown in more details (Figure 4.9).

The logic loop can be implemented circuitly by constructing a sequence of Grover operators (oracle and diffusion operators) and repeating them $\sim \sqrt{N}$ times. The obtained speedup with this algorithm is quadratic.

Algorithm 3 The Grover algorithm.

1. In the first stage, the input state is prepared

$$|\psi_0\rangle = \bigotimes_{i=0}^{n-1} |0\rangle \otimes |1\rangle = |0\rangle^{\otimes n}|1\rangle = |00...01\rangle$$

and then the Hadamard transformation is applied to each qubit of $|\psi_0\rangle$

$$|\psi_1\rangle = \bigotimes_{i=0}^{n-1} H|0\rangle \otimes H|1\rangle = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)^{\otimes n} \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) =$$
$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \left(|0\rangle - |1\rangle\right)$$

2. The next block, consisting of the oracle and the diffusion operator, must be repeated a number of times equal to $\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$; that means algorithm complexity is $O(\sqrt{N})$.
The oracle will have the task of marking the desired item with a negative phase, leaving the phases of the other elements unchanged.

$$U_\omega|x\rangle = \begin{cases} |x\rangle & if \;\; x = \omega \\ -|x\rangle & if \;\; x \neq \omega \end{cases} \;\Leftrightarrow\; U_\omega|x\rangle = (-1)^{f(x)}|x\rangle$$

Using a general oracle, it will be possible to trace it back to our particular problem by adding an ancillary qubit with state $H|1\rangle$

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \;\wedge\; |y\rangle = H|1\rangle = |-\rangle =$$
$$= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \Rightarrow$$
$$\Rightarrow U_\omega|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle = U_f|x\rangle|-\rangle$$

In the next stage, the diffusion operator will have the task of overturning the counter-phase component back, amplifying its probability amplitude; for the condition of normalization of the state, all the other components will suffer a consequent attenuation. The operator presents the following form:

$$U_s = H^{\otimes n} \cdot \left(2|0\rangle^{\otimes n}\langle 0|^{\otimes n} - \mathbb{I}_n\right) \cdot H^{\otimes n}$$

Repeating the operation a number of times proportional to $\sqrt{N}$, the probability amplitude for the w component will tend to 1 in absolute value, while all the other components will tend to zero.

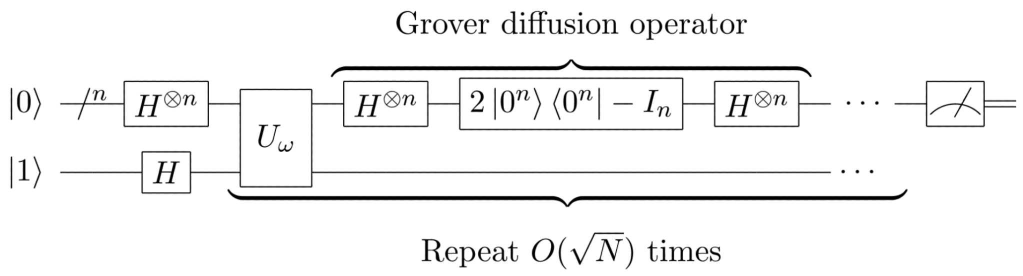

Grover diffusion operator
$|0\rangle$
n
$H^{\otimes n}$
$|1\rangle$
H
U_ω
$H^{\otimes n}$
$2|0^n\rangle\langle 0^n| - I_n$
$H^{\otimes n}$
Repeat $O(\sqrt{N})$ times

Figure 4.9: Quantum circuit to implement the Grover's algorithm

4.3.3 The Shor's algorithm

The *general number field sieve* is the most effective classical factorization algorithm known at this time, and it takes exponential times of the order

$$O\left(e^{c(logN)^{\frac{1}{3}}(loglogN)^{\frac{2}{3}}}\right)$$

to factor a big enough number N[80, 81].

Shor's quantum algorithm[82], on the other hand, could factor an integer in polynomial time; in particular, the complexity of this algorithm is:

$$O\left((logN)^2(loglogN)(logloglogN)\right)$$

This algorithm did not take long to pique many people's interest due to its potential to break classical cryptography; in fact, the reliability of many cryptographic systems is strongly dependent on the complexity of factoring large numbers; therefore, if a fast and safe method for factoring large numbers was developed, many current cryptographic systems might become vulnerable.

Shor's algorithm solves the following problem: determine the values of p and q given a composite number $N = pq$ and prime numbers p and q. The solution to the problem is to return to evaluating the order, or period, of a given function.

In Algorithm 4, the procedure is explored in more details.

Steps 1 and 3 can also be performed efficiently by a classical computer, while a quantum computer can more easily find the period of the function f in step 2, thus obtaining the required speed-up compared to the classic

Algorithm 4 The Shor algorithm.

```
1. Choose a random integer number in $1 < a < N$ and use Euclid's algorithm
   to find the greatest common divisor $gcd(N, a)$.
   If $gcd(N, a) \neq 1$, a factor of N has been discovered. If, on the other hand,
   $gcd(N, a) = 1$, continue to step 2.

2. Determine the order of $a \ mod \ N$, that is, the smallest integer value of $r$
   such that $a^r \ mod \ N = 1$.

3. Evaluate $r$

    3.1. If $r$ is odd, go back to step 1
    3.2. Else calculate $a^{r/2} \ mod \ N$

4. Evaluate $a^{r/2} \ mod \ N$

    4.1. If $a^{r/2} \ mod \ N = -1$ go back to step 1
    4.2. Else continue to step 5

5. Calculate, using Euclid's algorithm, the greatest common divisor

       $gcd(a^{r/2} + 1, N) \ , \ gcd(a^{r/2} - 1, N)$

   Both of them are non-trivial factors of $N$.
```

computer. Shor's algorithm could be used to crack public-key cryptographic schemes like the commonly used RSA scheme if a quantum computer with enough qubits could operate without succumbing to quantum noise and other quantum decoherence phenomena. The commonly used RSA asymmetric encryption algorithm, developed in 1977 by Ronald Rivest, Adi Shamir, and Leonard Adleman, is based on the principle that factoring large integers is computationally intractable. This theorem holds true for classical computers: no classical algorithm that can factor integers in polynomial time is known. However, Shor's algorithm demonstrates that the factorization of integers is effective on an ideal quantum computer; hence, it could potentially be able to breach RSA if one had quantum computers with a large number of qubits, which do not yet exist in the current state of technology.

This does not rule out the possibility of implementing this algorithm on a real quantum computer in the future: the only way to protect against quantum computer threats is to develop modern cryptographic schemes.

4.4 Grover's Algorithm

In this section, two practical implementations of Grover's algorithm are proposed using the following tools:

- **computation with fermionic particles**: utilised the open-source Python3 library, Qiskit, for simulating the circuit under ideal conditions, and the EBM Quantum Experience cloud platform for validating the results with a real fermionic quantum computer,

- **computation with photons**: utilised the open-source Python3 library, Perceval, for simulation under ideal circuit conditions, and the Quandela cloud platform for validation of results with a real photonic quantum computer.

Both realisations employ 3 qubits and thus a search space of dimension $2^3 = 8$. The value to be searched will be the binary string 101 only.

Grover's algorithm is a quantum algorithm developed by Lov K. Grover in 1996 that can be used to search an unsorted database with N elements in

$O\left(\sqrt{N}\right)$ time, which is exponentially faster than classical algorithms that require $O(N)$ time. The algorithm works by using a quantum oracle to mark the solution(s) to a search problem and then applying a series of Grover iterations to amplify the amplitude of the marked state(s), making them more likely to be measured. The algorithm terminates when a marked state is measured. The quantum oracle performs a simple task: given an input, it returns 1 if the input solves the search problem and 0 otherwise. The oracle is implemented by a black box that applies a phase flip to the marked state(s), i.e., it changes the sign of the amplitude of the marked state(s). The oracle can be constructed efficiently for various problems, including unstructured search, element distinctness, and graph connectivity. The Grover iterations apply a sequence of two quantum gates: the Hadamard gate and the Grover diffusion operator. The Hadamard gate creates a superposition of all possible states, while the Grover diffusion operator reflects the amplitudes about the average amplitude. Applying these gates repeatedly amplifies the amplitude of the marked state(s) while suppressing the amplitude of the unmarked states. The number of Grover iterations required to find a marked state with high probability depends on the number of solutions to the search problem and the database size. Grover's algorithm has important implications for quantum computing, demonstrating a significant speedup over classical algorithms for a wide range of problems. However, the algorithm is limited by the requirement for a quantum oracle, which can be difficult to construct for some situations, and by the need for precise control of the quantum state during the iterations.

4.4.1 Ideal Grover's Circuit implementation

A possible simple translation of Algorithm 3, with three qubits, with the only value to be searched equal to the binary string 101, is shown in Fig.4.10. The *Oracle* is a black box function that takes a quantum state as input and returns a phase flip on the state if it contains the marked item and no operation otherwise. The phase flip changes the sign of the amplitude of the marked item, making them more likely to be measured by the quantum computer. The quantum implementation of the oracle can be achieved using

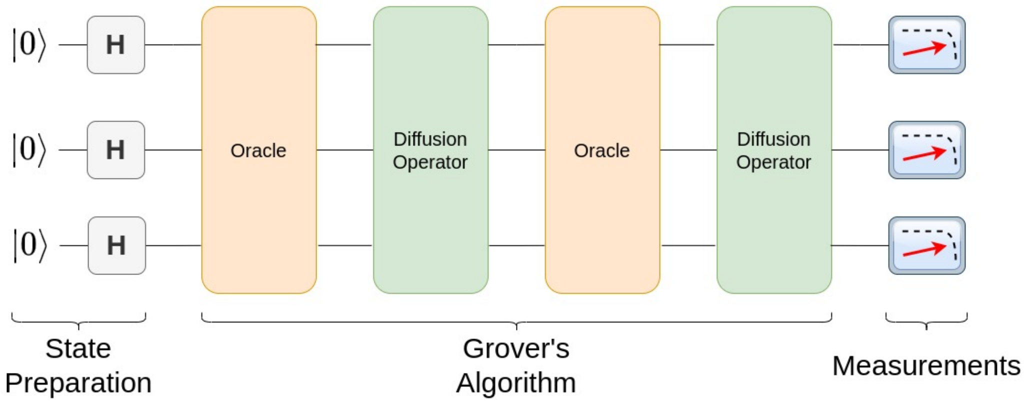


Figure 4.10: Quantum circuit scheme to implement the Grover's algorithm

a unitary operator that maps the input quantum state to an output quantum state with a phase flip on the marked item. This unitary operator can be constructed using reversible classical logic gates and additional qubits. For example, if the search problem is to find a marked item in a database of N items, the quantum input state can be a uniform superposition of all possible basis states $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |x_i\rangle$, where $|x_i\rangle$ represents a basis state of the system. The oracle operator, O, can then be defined as: $O|\psi\rangle = (-1)^{f(x)} |\psi\rangle$ where $f(x)$ is the function that determines whether x is the marked item, and $(-1)^{f(x)}$ is the phase flip that is applied to the state if x is marked.

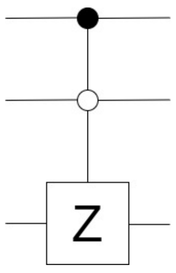


Figure 4.11: Oracle

$$O = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

The elements characterising the Oracle block are illustrated: Equation 4.1 displays the matrix that enables a sign flip transformation on state $|101\rangle$; it can be implemented through a Controlled-Controlled Z gate in a physical setting (Fig.4.11).

The Controlled-Controlled Z gate (CCZ gate - Fig.4.12) is a three-qubit gate that performs a controlled phase flip on the target qubit when both control qubits are in the $|1\rangle$ state. Otherwise, the target qubit remains unchanged. In this particular case, its defining matrix is in Eq.4.2.

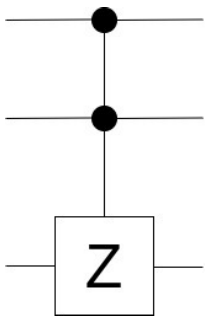


Figure 4.12: CCZ gate

$$CCZ = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \tag{4.2}$$

The *Diffusion Operator*, also known as the *Grover Operator*, is a unitary operator; it can repeatedly be applied to the quantum state to amplify the marked state's amplitude and suppress the unmarked states' amplitude. The number of iterations required depends on the number of marked states and the dataset size. In this specific case, it takes about $\frac{\pi}{4}\sqrt{N}$ iterations to achieve a high probability of measuring a marked state in a dataset of size N.

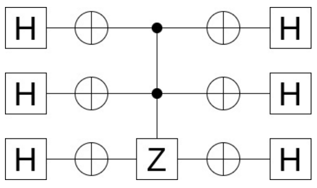


Figure 4.13: Diffusion Operator

$$S = \begin{pmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{pmatrix} \tag{4.3}$$

The elements characterising the Diffusion Operator block are illustrated: Equation 4.3 displays the matrix that enables the inversion of states around their average value and their subsequent reflection; it can be implemented through the circuit in Fig.4.13.

Thus, the unitary operator U performing the overall transformation is given by Eq.4.4.

$$
\begin{gathered}
U = S \cdot O \cdot S \cdot O \cdot (H \otimes H \otimes H) = \\
= \begin{pmatrix}
-\frac{\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} \\
-\frac{\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} \\
-\frac{\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} \\
-\frac{\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} \\
-\frac{\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} \\
\frac{11\sqrt{2}}{16} & \frac{\sqrt{2}}{16} & -\frac{\sqrt{2}}{16} & \frac{\sqrt{2}}{16} & \frac{\sqrt{2}}{16} & -\frac{\sqrt{2}}{16} & \frac{\sqrt{2}}{16} & -\frac{\sqrt{2}}{16} \\
-\frac{\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} \\
-\frac{\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{3\sqrt{2}}{16} & \frac{3\sqrt{2}}{16} & \frac{5\sqrt{2}}{16} & -\frac{5\sqrt{2}}{16}
\end{pmatrix}
\end{gathered}
\tag{4.4}
$$

In Eq.4.5, it is shown the final state $|\psi_{out}\rangle$ resulting from Grover's circuit. In Table 4.1 is displayed the probability of getting every single state after a measurement process (in the second row, there is the exact probability value expressed as a fraction, while in the third row as a decimal number); it indicates that if multiple investigations were conducted and measured the outcome, the state $|101\rangle$ would be gotten around 94% of the time.

$$
\begin{aligned}
|\psi_{out}\rangle &= U \cdot |000\rangle = \\
&= -\frac{\sqrt{2}}{16}|000\rangle - \frac{\sqrt{2}}{16}|001\rangle - \frac{\sqrt{2}}{16}|010\rangle - \frac{\sqrt{2}}{16}|011\rangle + \\
&\quad - \frac{\sqrt{2}}{16}|100\rangle + \frac{11\sqrt{2}}{16}|101\rangle - \frac{\sqrt{2}}{16}|110\rangle - \frac{\sqrt{2}}{16}|111\rangle
\end{aligned}
\tag{4.5}
$$

$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{121}{128}$	$\frac{1}{128}$	$\frac{1}{128}$
0.0078125	0.0078125	0.0078125	0.0078125	0.0078125	0.9453125	0.0078125	0.0078125

Table 4.1: Probability Distribution for state $|\psi_{out}\rangle$

4.4.2 Grover's Circuit with IBM Quantum Experience

Of particular interest is studying the algorithm, first simulating it with Qiskit and subsequently testing it on a real IBM quantum computer (*ibm_manila* platform - cf. Section 2.4.5).

Qiskit

The Qiskit circuit can be seen in Figure 4.14, and the corresponding Python code to create it is presented in Listing 4.1.

```
import qiskit

n_qubits = 3
qr = qiskit.QuantumRegister(n_qubits, name="q")
circuit = qiskit.QuantumCircuit(qr)
# Preparation
circuit.reset(qr[0])
circuit.reset(qr[1])
circuit.reset(qr[2])
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])
circuit.barrier(qr[0], qr[1], qr[2])
# Oracle
circuit.x(qr[1])
circuit.ccz(qr[0], qr[1], qr[2])
circuit.x(qr[1])
circuit.barrier(qr[2], qr[1], qr[0])
# Diffusion Operator
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])
circuit.x(qr[0])
circuit.x(qr[1])
circuit.x(qr[2])
circuit.ccz(qr[0], qr[1], qr[2])
circuit.x(qr[0])
circuit.x(qr[1])
circuit.x(qr[2])
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])
circuit.barrier(qr[0], qr[1], qr[2])
# Oracle
circuit.x(qr[1])
circuit.ccz(qr[0], qr[1], qr[2])
circuit.x(qr[1])
circuit.barrier(qr[2], qr[1], qr[0])
# Diffusion Operator
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])
circuit.x(qr[0])
circuit.x(qr[1])
circuit.x(qr[2])
circuit.ccz(qr[0], qr[1], qr[2])
circuit.x(qr[0])
circuit.x(qr[1])
circuit.x(qr[2])
circuit.h(qr[0])
circuit.h(qr[1])
circuit.h(qr[2])
```

Listing 4.1: Qiskit code to build the circuit

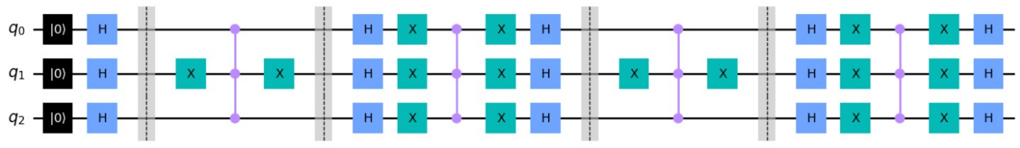


Figure 4.14: Quantum circuit scheme to implement the Grover's algorithm

Figures 4.15 and 4.16 illustrate the probability amplitudes (the height of the bars in the histogram represents the value of the modulus of the probability amplitude, while the colour of the bar represents the phase, according to the colour conventions in Fig.2.18c - blue for a phase of 0 radians and orange for a phase equal to π radians) and the probability distribution of the output state components, respectively.

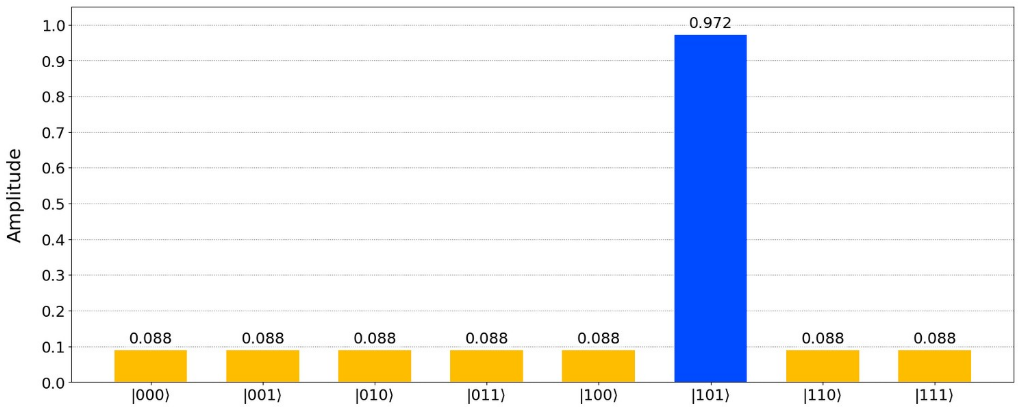


Figure 4.15: Probability Amplitudes for the output state

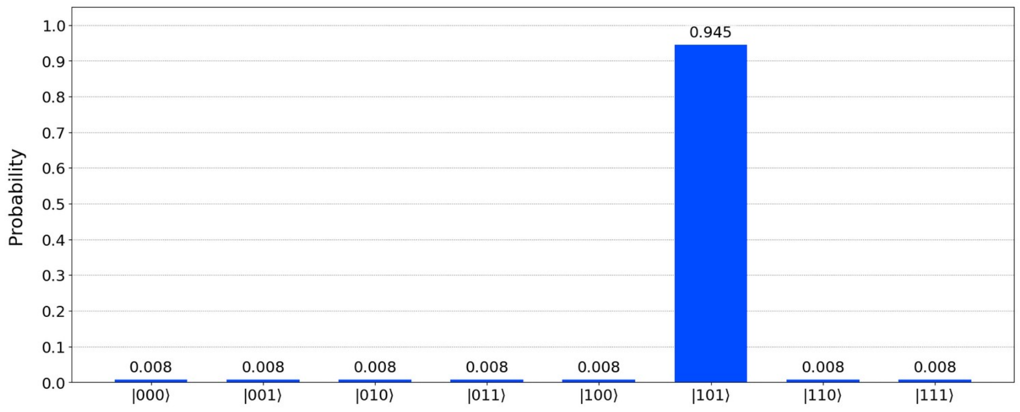


Figure 4.16: Probability Distribution for the output state

The obtained results perfectly align with the expected ones.

IBM QExperience

Qiskit contains dedicated APIs that permit perfect compatibility with the QASM language of IBM's real quantum platforms, such as *ibm_manila*. That allows algorithms designed and implemented on Qiskit to be executed directly on the real quantum computer. The circuit is decomposed and transpilled according to a basic set of gates used by the platform itself; *ibm_manila* uses the set formed by gates **CX** (CNOT), **ID** (Identity), **RZ** (rotation around the Z axis of the Bloch's sphere for a predefined angle), **SX** ($\sqrt{X}$), **X**.

Fig.4.17 shows the equivalent circuit in Fig.4.14, except for a global phase of $\frac{\pi}{4}$ radians, decomposed according to *ibm_manila*'s basic gates.

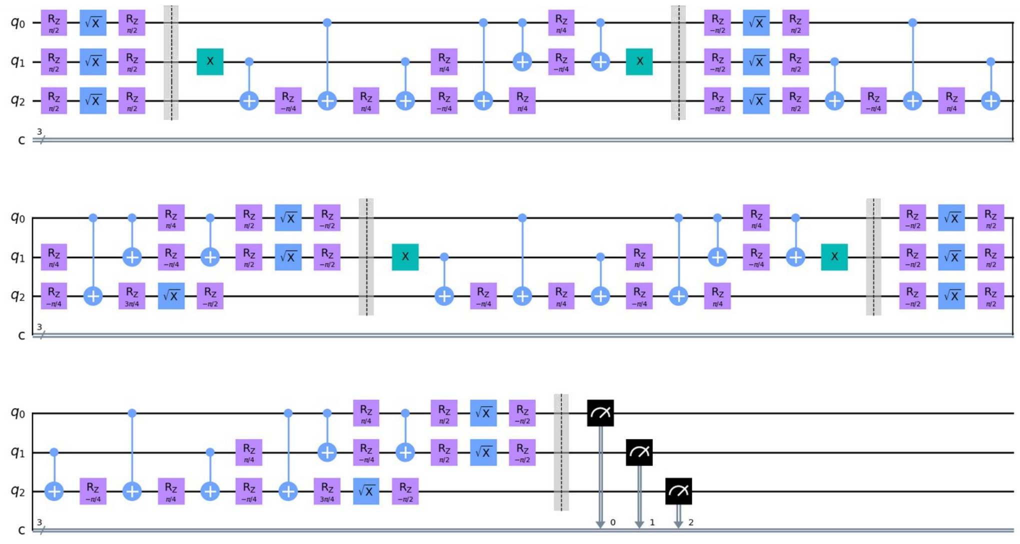
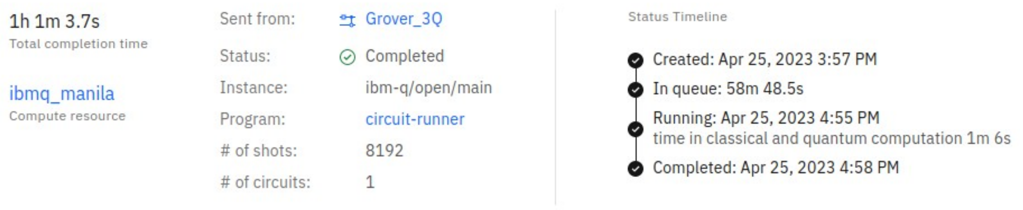


Figure 4.17: Decomposition of the circuit using the base gates for *IBM_manila* platform.

The job was completed in just over a minute, but the wait time in the queue was nearly an hour, as shown in Fig.4.18. Unfortunately, as depicted in Fig.4.19, the results indicate the presence of noise, which negatively impacts performance. It is primarily due to individual gate decoherence and circuit depth. The probability for state $|101\rangle$ drops significantly, from a predicted 94% to an observed 44%.

However, like the optical Entangler circuit, it is possible to implement error mitigation and outcome filtering techniques to enhance system performance. When using real quantum computers to reduce errors, it is important to consider the individual performance of the logic qubits (decoherence times of individual qubits and entanglement-coupled qubits), the depth of the circuit to be implemented and the Quantum Volume and the QPU (Quantum



1h 1m 3.7s
Total completion time

ibmq_manila
Compute resource

Sent from: Grover_3Q
Status: Completed
Instance: ibm-q/open/main
Program: circuit-runner
# of shots: 8192
# of circuits: 1

Status Timeline

Created: Apr 25, 2023 3:57 PM
In queue: 58m 48.5s
Running: Apr 25, 2023 4:55 PM
time in classical and quantum computation 1m 6s
Completed: Apr 25, 2023 4:58 PM

Figure 4.18: Job info

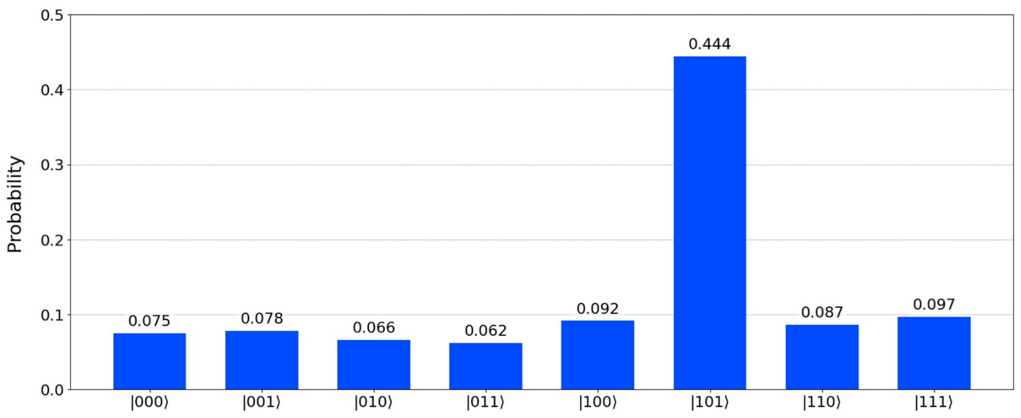


Figure 4.19: Probability Distribution for the output state from *IBM_manila* platform.

Processing Unit) of the specific quantum computer. By taking these aspects into account globally, it is possible to significantly increase the probability of the correct quantum output state and reduce execution times [83].

4.4.3 Grover's Circuit with Quandela Cloud

The circuit that performs Grover's algorithm can also be implemented using beam splitters and phase shifters, as described in the previous chapter. Considering the limited number of states for the specific case, equal to 8, the most straightforward implementation is the one according to the Single Photon Mode. This mode generally presents serious scalability problems as the relationship between the system states and its modes is bijective.

Figure 4.20 displays the circuit diagram generated by Perceval. This primary circuit is built using the same base circuit as demonstrated in the previous chapter, which can be found in Figure 3.3. The circuit diagram, too long to be displayed in its original horizontal extension, has been cut and reported in three vertical sub-images solely for readability reasons.

Upon analysing the scattering matrix of the transformation, it has become clear that it perfectly corresponds to matrix in Eq.4.4. The optical circuit system follows the same evolution as the theoretical circuit. As expected, this results in identical graphs for the probability amplitudes of the final state and the probability distributions of the measured states. To illustrate this, Fig.4.21 and Fig.4.22 show the graphs obtained by simulating the circuit using Perceval; the graphic conventions utilised in this figure are identical to those used in Figures 4.15 and 4.16.

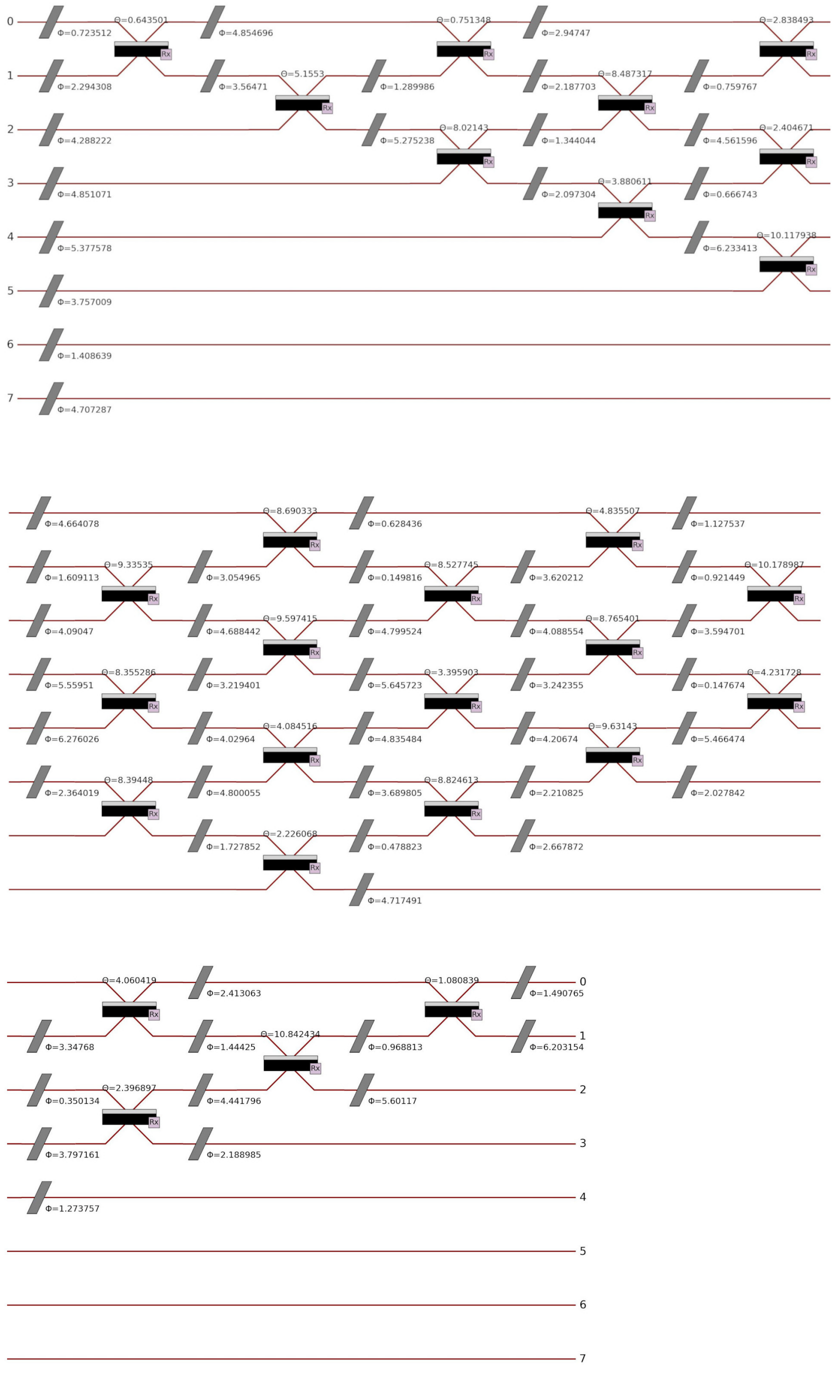


Figure 4.20: Grover's algorithm implemented using an Optical Quantum Processor

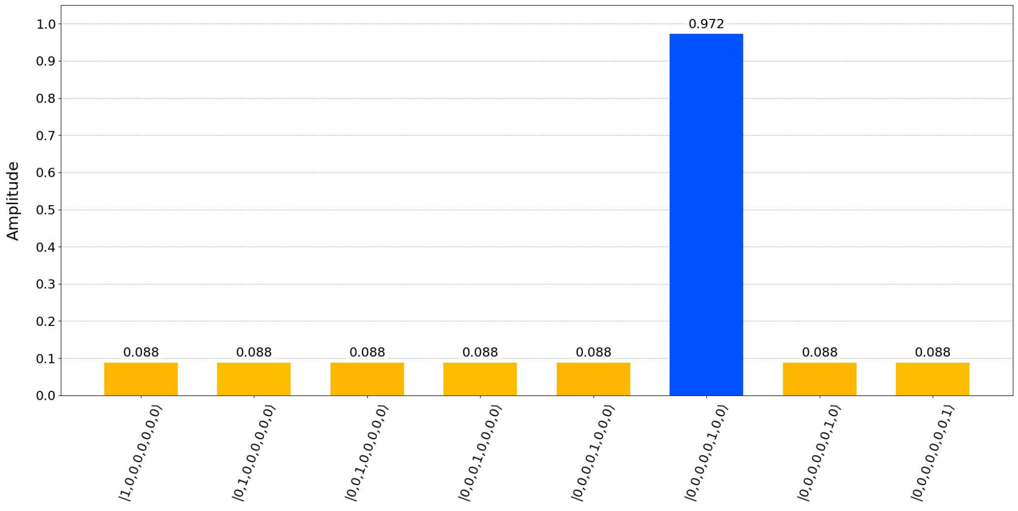


Figure 4.21: Probability Amplitudes for the output state in Grover's algorithm, implemented with Perceval

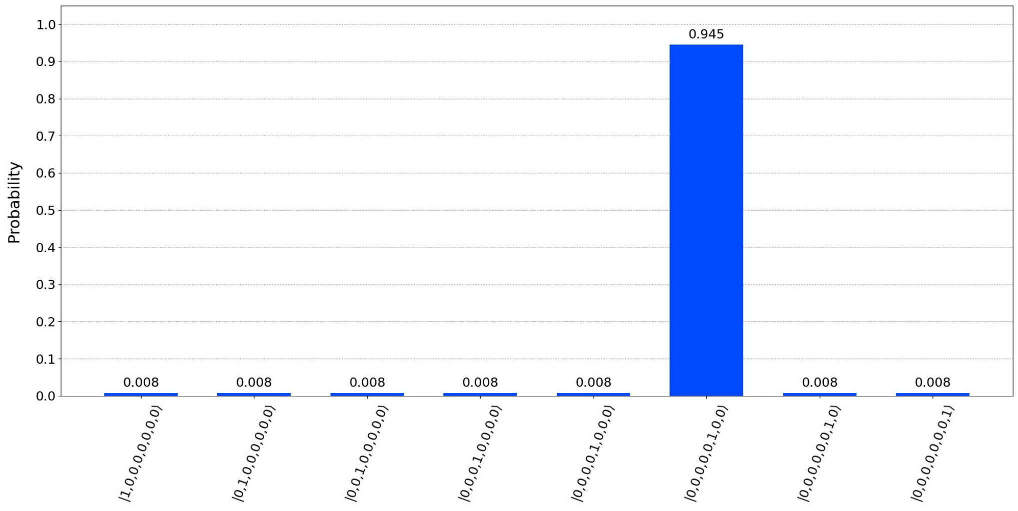
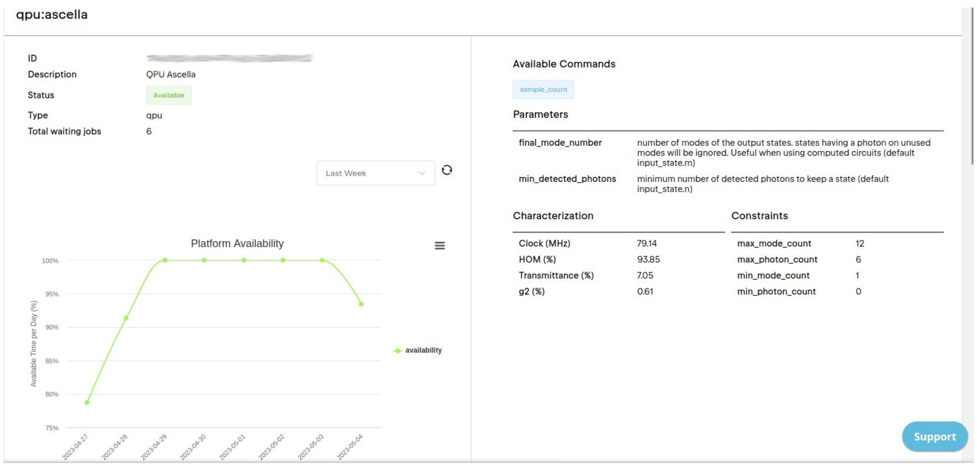


Figure 4.22: Probability Distribution for the output state in Grover's algorithm, implemented with Perceval

Like Qiskit for IBM's quantum platforms, Perceval has specific APIs that guarantee full compatibility with Quandela's actual quantum processors, such as *QPU:Ascella* (Fig.4.23). This function allows algorithms developed and implemented on Perceval to run directly on the optical quantum computer in real-time.



qpu:ascella

ID	[redacted]
Description	QPU Ascella
Status	Available
Type	qpu
Total waiting jobs	6

Available Commands

sample_count

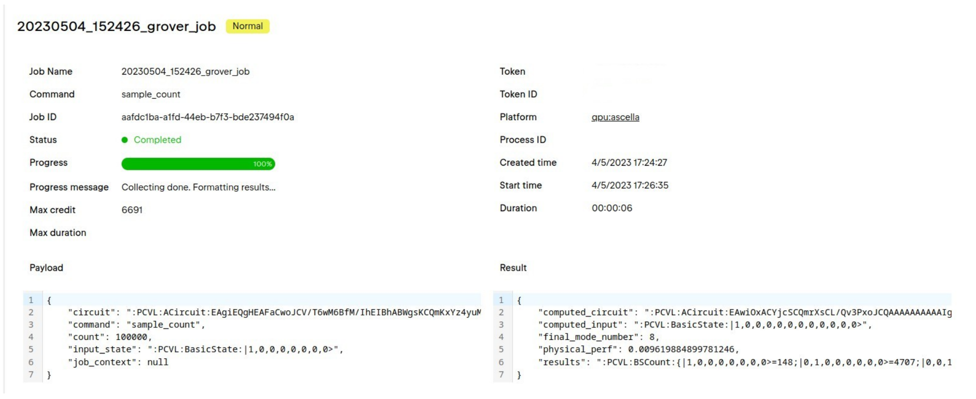
Parameters

final_mode_number	number of modes of the output states. states having a photon on unused modes will be ignored. Useful when using computed circuits (default input_state.m)
min_detected_photons	minimum number of detected photons to keep a state (default input_state.n)

Characterization		Constraints	
Clock (MHz)	79.14	max_mode_count	12
HOM (%)	93.85	max_photon_count	6
Transmittance (%)	7.05	min_mode_count	1
g2 (%)	0.61	min_photon_count	0

Figure 4.23: QPU:Ascella characteristics

The task was completed in just two minutes, with a queue wait time of only 6 seconds (Fig.4.24). However, the results indicate the presence of noise, which negatively impacts performance due to various physical effects on individual gates, such as the HOM effect[13] [84, 85] and non-ideal transmittance of beam splitters. Additionally, circuit depth also plays a role. The probability of state $|0,0,0,0,0,1,0,0\rangle$ (that is $|101\rangle$) is similar to the predicted theoretical results, with an observed value of 92% compared to the predicted 94% (Fig.4.25).


20230504_152426_grover_job Normal

Job Name	20230504_152426_grover_job	Token	[redacted]
Command	sample_count	Token ID	[redacted]
Job ID	aafdc1ba-a1fd-44eb-b7f3-bde237494f0a	Platform	qpu:ascella
Status	● Completed	Process ID	
Progress	100%	Created time	4/5/2023 17:24:27
Progress message	Collecting done. Formatting results...	Start time	4/5/2023 17:26:35
Max credit	6691	Duration	00:00:06
Max duration			

Payload

```
{
    "circuit": ":PCVL:ACircuit:EAgiEQgHEAFaCwoJCV/T6wM6BfM/IhEIBhABWgsKCQmKxYz4yuM
    "command": "sample_count",
    "count": 100000,
    "input_state": ":PCVL:BasicState:|1,0,0,0,0,0,0,0>",
    "job_context": null
}
```

Result

```
{
    "computed_circuit": ":PCVL:ACircuit:EAwiOxACYjcSCQmrXsCL/Qv3PxoJCQAAAAAAAAAIg
    "computed_input": ":PCVL:BasicState:|1,0,0,0,0,0,0,0,0,0,0,0>",
    "final_mode_number": 8,
    "physical_perf": 0.009619884899781246,
    "results": ":PCVL:BSCount:{|1,0,0,0,0,0,0,0>=148;|0,1,0,0,0,0,0,0>=4707;|0,0,1
}
```

Figure 4.24: Job executed on QPU:Ascella

13The HOM effect, also known as the Hong-Ou-Mandel effect, is a phenomenon in quantum optics where two identical photons, when injected into opposite inputs of a beam splitter, always emerge together in one output port or the other. This effect is due to the wave nature of photons and the quantum mechanical principle of indistinguishability.

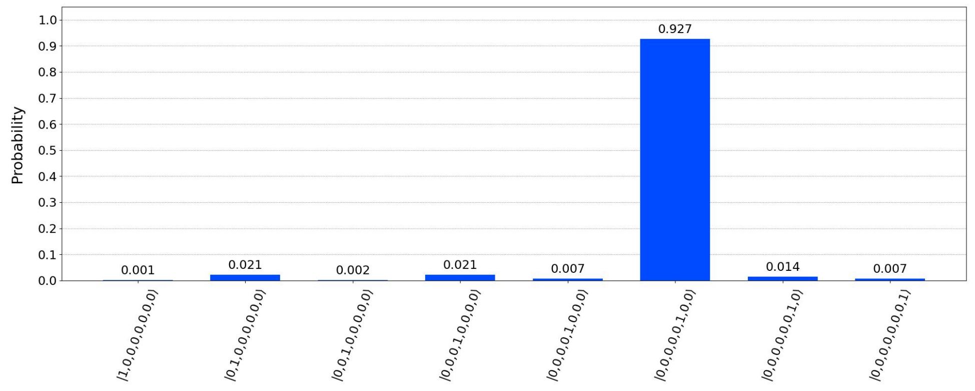


Figure 4.25: Probability Distribution on QPU:Ascella

The impressive outcomes of this technology make it a compelling and promising option, despite the current uncertainties and challenges. Numerous researchers are dedicated to advancing quantum technology [86, 87, 88], particularly in the field of photonics [89, 90, 91, 92]. This work has made a significant contribution by surpassing the limitations of the results achieved using superconducting computers on Grover's algorithm. With the use of a photonic quantum computer, it was possible to partially overcome the problem of particle interaction, resulting in a substantial increase in the probability of obtaining the corrected state. This aspect is a critical point for fermionic computers, which seem to achieve only up to 80% probability [83]. However, the biggest issue with this solution is scalability. Each qumode corresponds to only one quantum state, requiring eight qumodes for a three-qubit Grover algorithm. This fact becomes a real concern that is not easy to solve when attempting to implement the algorithm on a large data set.

4.5 Conclusions

Quantum computing is a rapidly growing area that has the potential to revolutionise the way some complex problems can be solved. The development and usage of quantum computing technology have made significant progress in recent years. Researchers are continuously exploring new ways to leverage this technology to solve complex or impossible problems for classical computers. Quantum computing's unique properties, such as superposition and entanglement, enable it to perform specific calculations exponentially faster than classical computers. This technology can potentially transform fields such as cryptography, materials science, and drug discovery. Significant investments from both private companies and governments have driven the development of quantum computers. That has allowed the maturation of quantum computing hardware, software, and algorithms that have demonstrated several quantum advantage applications. However, there are still significant challenges to be addressed in the field of quantum computing. One of the primary challenges is the development of reliable and scalable quantum hardware. Though substantial progress has been made in developing quantum computing systems, they are still prone to errors. Considerable improvements are needed to make them practical for widespread usage. Another challenge is the evolution of quantum software and algorithms that can exploit quantum computing's unique properties. The field of quantum software is still in its early stages, and researchers are continuously exploring new algorithms that can be used to solve complex problems. Despite these challenges, quantum computing has already demonstrated significant potential for solving complex problems. In cryptography, quantum computers can break many classical encryption algorithms, making it necessary to develop new quantum-resistant encryption methods. In materials science, quantum computing can be used to simulate complex chemical reactions, enabling the design of new materials with specific properties. In drug discovery, quantum computing can be used to simulate the behaviour of molecules, enabling the discovery of new drugs. In addition to these applications, quantum computing has the potential to impact several other fields, including finance, logistics, and optimisation. The development of quantum computing technology will likely lead to entirely new industries and business models with significant economic and social implications. In conclusion, the development and usage of quantum computing technology have the potential to transform the way we solve complex problems. While considerable challenges remain to be addressed, the progress made in recent years is encouraging. Researchers are continuously exploring new ways to leverage this technology to solve real-world problems. As quantum computing technology continues to develop, it can be expected to significantly impact several fields, leading to the creation of new industries, business models, and economic opportunities.

Chapter 5

Quantum Machine Learning

5.1 Introduction

The stages of knowledge in the history of humankind have always alternated between amazement at the immensity of the phenomenon before us and the joyful conquest for the objectives set. The development and introduction into the science of a powerful mathematical arsenal have slowly enabled many obstacles to be overcome, confirming Galileo's intuition that mathematics is the straightforward language that enables us to dialogue with nature. At the beginning of the 20th century, our optimism in positivist determinism was strongly shaken by new phenomena that led to the birth of quantum mechanics. Nevertheless, the observation of the generation of deterministic chaos from models with a seemingly simple apparatus of differential equations and the evident need to describe basic molecular structures by simulating them with automatic calculation tools that require ever-increasing computational capabilities are suggesting that the time has come for a profound reflection on our tools of scientific investigation. Feynman stated that describing the reality that surrounds us, which has an intrinsically quantum nature, through a type of computation based on quantum mechanics would be the key to exponentially lowering the computational complexity of the system in question and correctly managing the predictive capacity for the model. It might be also added that the introduction of quantum computing machines would also solve the problems related to the imminent reaching of the construction limit of current computers (Amdahl's law), to the possibility of a drastic reduction in the energy required by today's computing machines thanks to computational reversibility, and to the development of nanotechnology.

The possibility of such opportunities has resulted in a growing international interest in such devices, with considerable funding in the relevant research areas. Quantum Computing machines can therefore respond to questions that would be unimaginable for actual old-style apparatuses [78], as they would need the whole Universe's existence time to complete the same task. That is allowed by some quantum peculiarities that show up at the littlest of scales, like Superposition, Entanglement and Interference [79]. Although the theoretical study of quantum algorithms suggests the real possibility of solving computational problems that are currently intractable [82], the current technology has not reached maturity, such as to obtain truly appreciable advantages. We are still in the so-called era of Noisy Intermediate-Scale Quantum (NISQ) [75]. NISQ-devices are noisy and have limited quantum resources; this presents various obstacles when implementing a gate-based quantum algorithm. In order to determine if an implementation of a gate-based algorithm would run effectively on a particular NISQ-device, numerous aspects of circuit implementation, such as depth, width, and noise, should be considered [93, 94].

Variational Methods [95, 96] are widely used in physics, and most of all in quantum mechanics [97]. Their direct successors, Variational Quantum Algorithms (VQAs), have appeared to be the most effective technique for gaining a quantum advantage on NISQ devices. VQAs are undoubtedly the quantum equivalent of very effective machine-learning techniques like neural networks. Furthermore, VQAs use the classical optimisation toolbox since they employ parametrised quantum circuits to run on the quantum computer and then outsource parameter optimisation to a classical optimiser. In contrast to quantum algorithms built for the fault-tolerant time period, this technique provides the extra benefits of keeping the quantum circuit depth small and minimising noise [98].

The aim is to find the exact, or a well-approximated, parameter values' set that minimises a given cost (loss) function, which depends on the parameters themselves and, naturally, on the input values, x; these are the non-trainable part of the schema. An output measurement apparatus and a parameter modulation circuit serve as regulators (Fig.5.1). In this specific case, one is

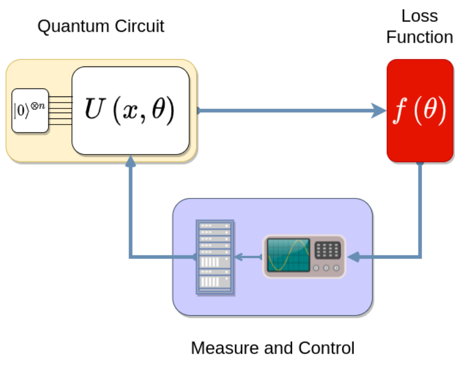


Figure 5.1: Scheme

dealing with a *n-qubits* quantum ansatz, whose transformation is represented by the $n \times n$ unitary matrix U that depends on the vector of parameters θ, with dimensions $m \,\epsilon\, \mathbb{N}$ (Eq.5.1); the loss function is the *expectation value* measured on every single qubit channel (Eq.5.3); the observable is the operator O_i, where Z_i is the Z-Pauli Operator, acting on i-th qubit (Eq.5.2); the quantum state on which to operate the measurements will be the state resulting from the ansatz (ψ in Eq.5.1).

$$\left|\psi\left(\bar{\theta}\right)\right\rangle = U\left(\bar{\theta}\right) \cdot |0\rangle^{\otimes n} \tag{5.1}$$

$$O_i = Z_i\,, \quad i \,\epsilon\, \{1..n\} \subset \mathbb{N} \tag{5.2}$$

$$\left\langle E_i\left(\bar{\theta}\right)\right\rangle = \left\langle \psi^{\dagger}\left(\bar{\theta}\right) \middle| O_i \middle| \psi\left(\bar{\theta}\right)\right\rangle \tag{5.3}$$

It is well-known that Quantum Computers can take care of specific issues quicker than traditional ones. As it may, stacking information into a quantum computer is not paltry. It should be encoded in quantum bits (qubits) to stack the information. There are multiple ways qubits can address the data, and, in this manner, various information encodings are conceivable [99]. Several methods can embed data: Basis Encoding [100, 101], Amplitude Encoding [102, 103, 104], and Angle Encoding [105, 106] are the most common to implant information into a firstly prepared quantum state.

In Section 5.2, aspects of the usability of quantum computation in machine learning have been explored. Specifically, the aim was to compare the performance of a basic FFNN with dense layers and an analogous network composed of quantum components in learning a simple classification problem.

In Section 5.3, this investigation focused on developing new variational methods based on quantum optics and applied to Machine Learning models. Three different optical quantum models are compared to try to understand the relationship between response accuracy and system hyperparameters.

Section 5.2 is based on a paper published by the author in 2022 [107], while section 5.3 is part of a paper submitted by the author to IEEE Access in August 2023.

5.2 Quantum models for Machine Learning

The possibility of using machine learning techniques in quantum computing has been gaining ground since 2010 [108, 109, 110]. The incorporation of quantum algorithms into machine learning programmes is known as quantum machine learning [13, 14, 41, 111]. The expression is most commonly used to refer to machine learning methods for evaluating classical data run on a quantum computer. While machine learning techniques are used to calculate vast quantities of data, quantum machine learning employs qubits and quantum operations or specialised quantum systems to speed up computing [112] and data storage [113]. Quantum machine learning also refers to an area of study investigating the theoretical and functional analogies between specific physical systems and learning systems, namely neural networks [114, 115].

Unfortunately, the current technological development does not yet allow the full potential of quantum computers to be expressed, which will only reach maturity in the next few decades. At present, however, it is possible to use quantum computers in feedback circuits that mitigate the effect of various noise components [116, 117]. VQAs, which utilise a conventional optimiser to train a parameterised quantum circuit, have been considered an effective technique for addressing these restrictions.

The core part of the computational stage consists of a sequence of gates applied to specified wires in variational circuits. Like the design of a neural network that merely describes the basic structure, the variational approach can optimise the types of gates and their free parameters. In time, the quantum computing community has suggested several variational circuit types that can distinguish three main base structures, depending on the shape of the ansatz: layered gate ansatz [118], alternating operator ansatz [119, 120], and tensor network ansatz [121].

It has been implemented in a network inspired by a simple type of architecture, using a minimum number of quantum gates and trainable parameters.

Compared to the solutions mentioned above, the proposed network has a significant strength: its structural size is related to the number of parameters to be used (and thus the number of serial circuit stages) and not to the size of the individual element of the input dataset (Fig.5.2). Since, for the current fermionic circuits, the major problem is the decoherence noise of the quantum states, a solution with a high circuit depth would have considerably deteriorated the obtainable results; the second strength of the proposed solution is, therefore, the attempt to minimise the number of series circuit stages. That made it possible to adequately modulate the hyper-parameters of the implemented network, resulting in a classifier with an exceptionally high degree of accuracy and a low number of quantum gates.

5.2.1 The architecture of the system

The network has a general structure of this type:

- A first classical dense layer, to accept features as input
- A quantum circuit formed by a succession of rotational and entanglement gates
- A final classical dense layer for classification

The quantum structure proposed differs from some other proposals, which encodes input features to transform them into quantum states [122, 123]; the initial state, the "prepared state", is simply $|0\rangle^{\otimes n}$. The proposed architecture is based on the typical "Prepared State fixed; Ansatz parameterised" model. Several variants exist in the literature, especially the most known *Instantaneous Quantum Polynomial* (IQP) circuits [124]. The initial dense network, which named *input-layer*, acts as an interface between the input data and the quantum layer; its output directly drives the internal rotations of the quantum state, attributing new values to the rotation angles at each epoch (Fig. 5.2). Almost all the simulations were conducted using quantum

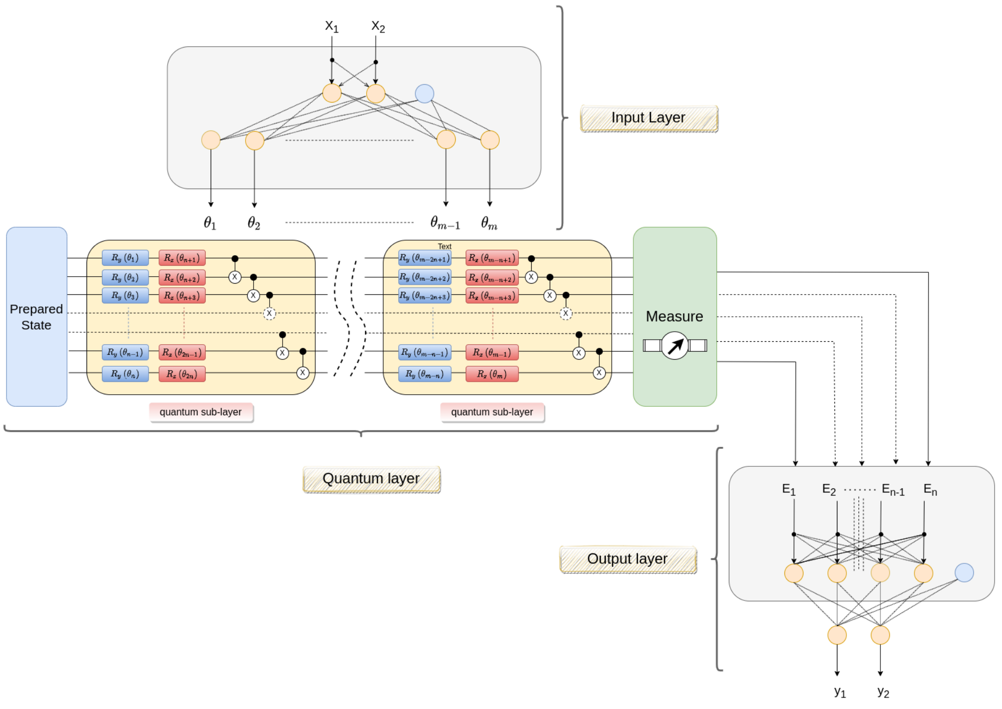


Figure 5.2: Architecture of the quantum neural network

sub-layers consisting of a first stack of rotation gates around the Y-axis of the Bloch's sphere, a second stack of rotation gate around the Z-axis and a chain of cX gates (CNOT) to maximise the entanglement effect. Rotation operators help primarily to ensure the *expressibility* of a parameterized quantum circuit. That is essentially the total coverage of Hilbert space by the hypothesis space of the ansatz itself [125]. In addition, the chain of CNOT operators is tasked with maximising the entanglement effect of individual qubits, as the entanglement phenomenon plays a crucial role in quantum computation. This feature is called *Entangling Capability* [126].

An authentic advantage is that the network's response remains independent of the number of qubits. The only real quantum noise remains due to the construction and constitution of the ansatz and the device, or simulator, used for experimentation.

5.2.2 Data extraction and processing

The database chosen for the classification problem is called *two-moons*, which generates two interleaving semicircles of 2D-points, and is characteristic for the study of clustering and classification algorithms.

The dataset was generated thanks to the dedicated function from the Python Scikit-Learn library[1].

The input domain is $[-2.0, 3.0] \times [-2.0, 2.5] \subset \mathbb{R}^2$. One thousand samples were randomly generated, equally distributed over the two classes. The classes were initially chosen with a Gaussian Error Coefficient of 0.05: this coefficient quantifies the capacity of separation of the two categories: the higher its value, the greater the degree of overlap of the two classes (Fig. 5.3). The number of samples in the dataset was divided in such a way as to ensure the following quotas: 70% for the training set, 20% for the validation set and the remaining 10% for the test set.

5.2.3 Model construction and validation

Two twin models have been realised in parallel, using for one of them the Python library for quantum simulation Cirq[2] with TensorFlowQuantum[3], by Google, and for the other an open-source Python library, PennyLane[4], by Xanadu, which can be perfectly interfaced with the most famous Deep Learning frameworks, such as Keras, and of course TensorFlow, and PyTorch.

All tests and simulations were performed using the specialised open-source library for Deep Learning, Keras[5].

The TFQ (TensorFlowQuantum) model is composed as follows (Fig. 5.4a):

[1]https://scikit-learn.org/stable/index.html
[2]https://quantumai.google/cirq
[3]https://www.tensorflow.org/quantum
[4]https://pennylane.ai/
[5]https://keras.io/

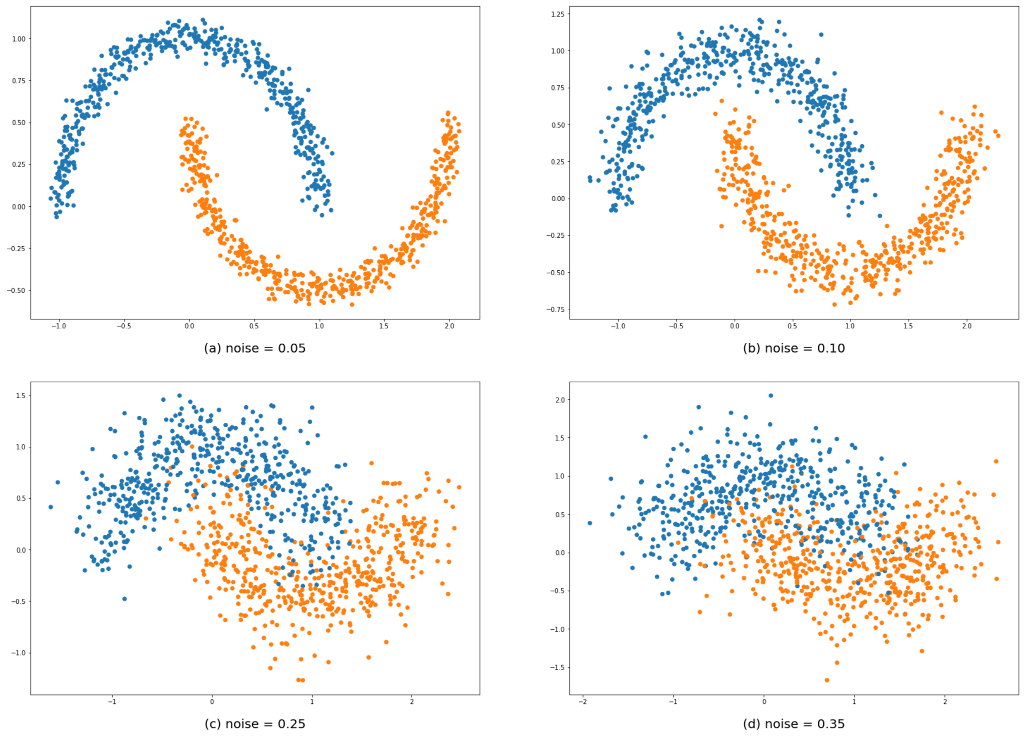
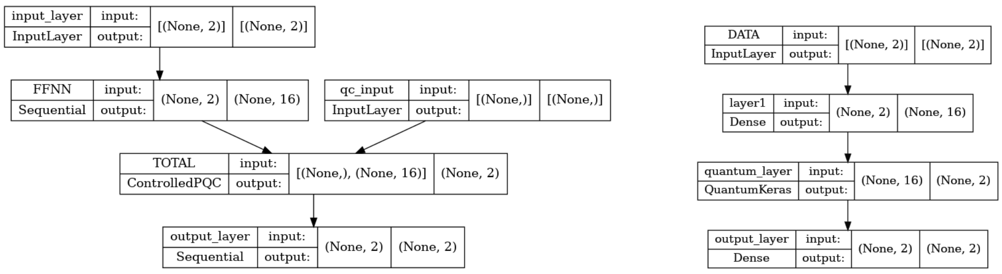


Figure 5.3: Distribution of pattern points on the plane in relation to Gaussian Noise

- A layer for data input, called *input_layer*.
- A Dense layer, which adapts the input to the number of parameters, in this specific case 16, called *FFNN*.
- A layer for the ansatz input in the form of a tensor, called *qc_layer*.
- A control layer (called *TOTAL*), which receives the two inputs, modulates the values of the parameters and returns, for each wire in the circuit, the expected value.
- A final Dense layer (called *output_layer*), with a number of outputs equal to the number of classes in the dataset, which returns the probability of belonging to them (*softmax* activation).

The PQML (PennyLane Quantum Machine Learning) model is composed as follows (Fig. 5.4b):

- A layer for data input, called *DATA*.
- A Dense layer, which adapts the input to the number of parameters, in this specific case 16, called *layer1*.
- A custom layer, called *quantum_layer*, designed as a subclass of the Layer class from the Keras library, which receives in input the values of the parameters of the ansatz and returns, for each wire of the circuit, the expected value.
- A final Dense layer (called *output_layer*), with a number of outputs equal to the number of classes in the dataset, which returns the probability of belonging to them (*softmax* activation).


input_layer | InputLayer | input: | output: | [(None, 2)] | [(None, 2)]
FFNN | Sequential | input: | output: | (None, 2) | (None, 16)
qc_input | InputLayer | input: | output: | [(None,)] | [(None,)]
TOTAL | ControlledPQC | input: | output: | [(None,), (None, 16)] | (None, 2)
output_layer | Sequential | input: | output: | (None, 2) | (None, 2)
DATA | InputLayer | input: | output: | [(None, 2)] | [(None, 2)]
layer1 | Dense | input: | output: | (None, 2) | (None, 16)
quantum_layer | QuantumKeras | input: | output: | (None, 16) | (None, 2)
output_layer | Dense | input: | output: | (None, 2) | (None, 2)

a TQF Model b PQML Model

Figure 5.4: Quantum Neural Network Models utilised

The number of ansatz parameters is given by the product of the number of qubits, the number of rotation operators acting on each line of the circuit within each sublayer, and the number of sublayers. For the tests and simulations, it has been opted to use an ansatz consisting of 4 sublayers to avoid creating an excessive computational load on the quantum layer. By initially constraining the number of qubits to 2 and not introducing any form of noise, it can be said that both models proved to be extremely light in terms of the number of trainable parameters: 54 against the 354 of a similar FFNN Fully-Connected (Feed-Forward Neural Network), with three layers. The disadvantage is that the simulation run times are about one order of magnitude longer than the latter: about 1.5 minutes per epoch, on a HP ProBook 450 G6, with 8 × Intel® Core™ i7-8565U CPU @ 1.80GHz, 32

GiB of RAM, GeForce MX130. In fact, to simulate an ansatz is necessary to operate many matrix products against only one matrix product for a standard dense layer.

The entire model was compiled using a classical optimiser, *SGD* (Stochastic Gradient Descent), with *Mean Absolute Error* as the Loss Function and *Accuracy* as the metric.

5.2.4 Analysis of results

The results obtained are excellent. Figure 5.5 shows the metrics (LOSS: loss function on training set, ACCURACY: accuracy on training set, VAL_LOSS: loss function on validation set, VAL_ACCURACY: accuracy on validation set,) for the TFQ model, trained for 20 epochs, with different values of Gaussian Noise on the dataset generation (legend).

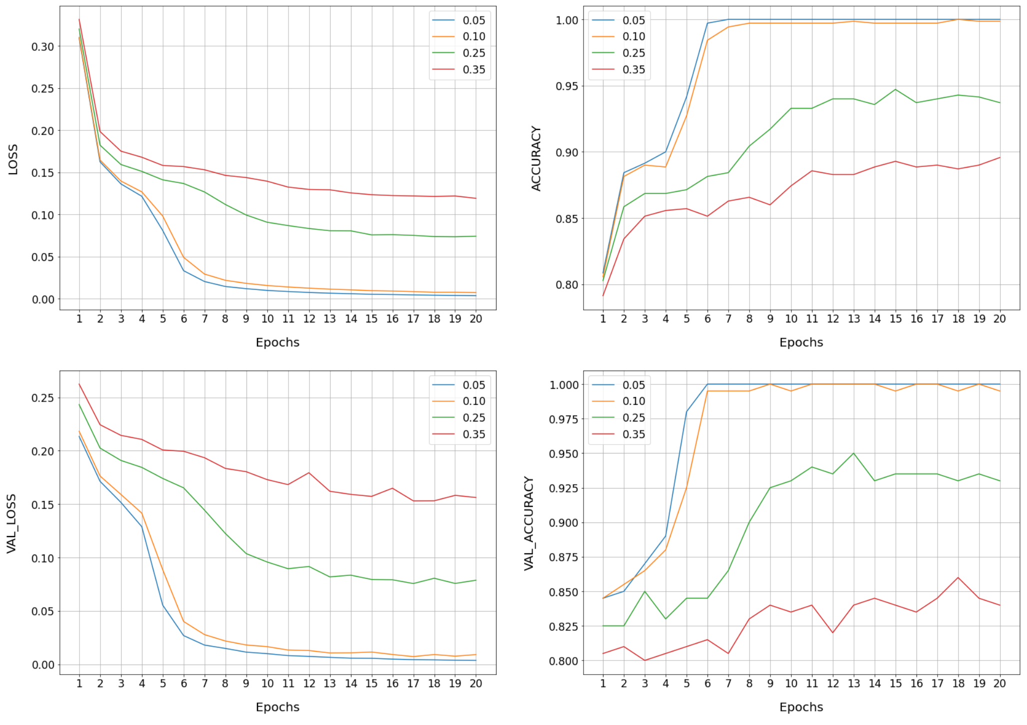


Figure 5.5: Metrics for TFQ model

The model achieves 100% accuracy for low-noise datasets and more than 80% accuracy for high-noise datasets.

	Samples: 20 / 200			Samples: 100 / 1,000			Samples: 500 / 5,000		
	n qubits			n qubits			n qubits		
NOISE	**2**	**3**	**4**	**2**	**3**	**4**	**2**	**3**	**4**
0.05	100.0 %	95.0 %	95.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
0.10	100.0 %	95.0 %	95.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
0.25	95.0 %	80.0 %	90.0 %	93.0%	94.0 %	94.0 %	93.8 %	93.8 %	90.0 %
0.35	80.0 %	75.0 %	85.0 %	88.0%	86.0 %	89.0 %	88.4 %	88.6 %	88.4 %

Table 5.1: Percentage of correct evaluations on the test set for TFQ model.

Table 5.1 shows the percentages of correct evaluations on the test set as a function of the Gaussian Noise on the dataset, the number of qubits used, datasets with different cardinalities (the header of the table shows the number of samples of the test set on the number of total samples). It is referred to TFQ model.

Figure 5.6 shows the usual metrics for the PQML model, trained for 20 epochs, with different values of Gaussian Noise on the dataset generation (legend).

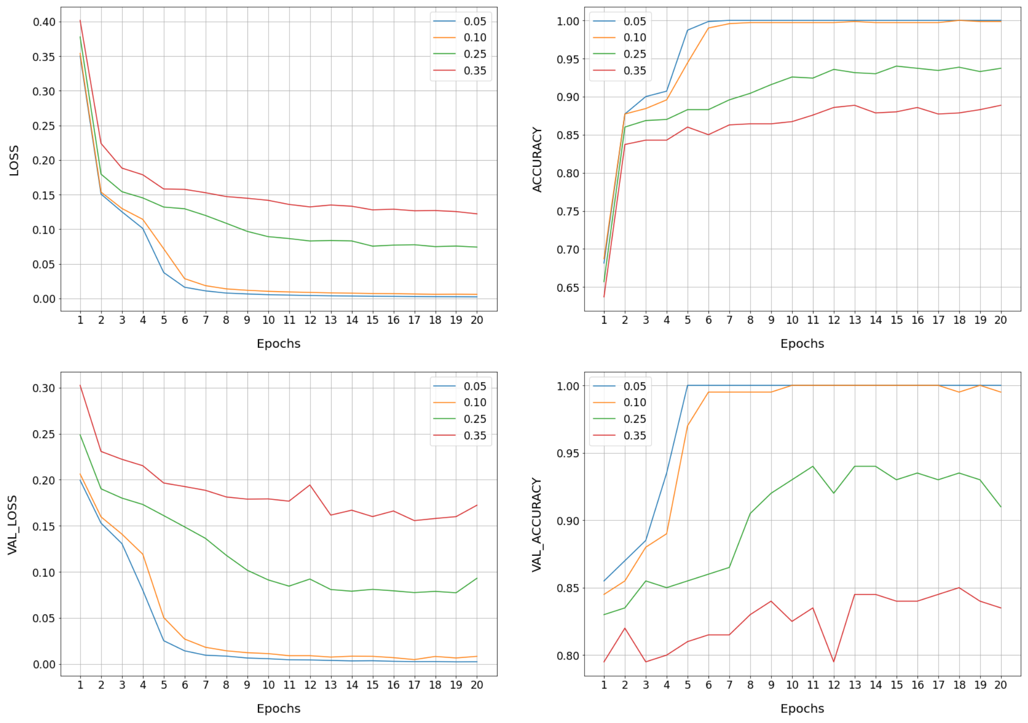


Figure 5.6: Metrics for PQML model

The model achieves 100% accuracy for low-noise datasets and beyond

80% accuracy for high-noise datasets.

NOISE	Samples: 20 / 200			Samples: 100 / 1,000			Samples: 500 / 5,000		
	n qubits			n qubits			n qubits		
	2	3	4	2	3	4	2	3	4
0.05	100.0 %	100.0 %	95.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
0.10	95.0 %	100.0 %	95.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
0.25	95.0 %	100.0 %	80.0 %	94.0%	93.0 %	94.0 %	94.8 %	94.0 %	93.8 %
0.35	75.0 %	90.0 %	75.0 %	87.0%	88.0 %	86.0 %	88.2 %	88.4 %	87.2 %

Table 5.2: Percentage of correct evaluations on the test set for PQML model.

Table 5.2 shows the percentages of correct evaluations on the test set as a function of the Gaussian Noise on the dataset, the number of qubits used, datasets with different cardinalities (the header of the table shows the number of samples of the test set on the number of total samples). It is referred to PQML model.

Figure 5.7 shows the usual metrics for a simple FFNN Fully-Connected, three dense layers, with the same inputs and output as quantum networks, trained for 20 epochs and with 354 trainable weights. Without the use of regularisers or drop-out layers, an early overfitting of the model can be observed. The two models exhibit remarkably similar tendencies, demonstrating their quality and validity.

Interestingly, unlike an analogous FFNN, a quantum network, since it contains only unitary transformations, can reduce the overfitting phenomenon without the necessity of introducing any particular regularizers or drop-out layers (at least for certain values of ansatz depth). The scale of the signal feed-forward (in time) and the mistakes conveyed backwards are maintained by a unitary matrix since it preserves the length of the vector to which it is applied [127].

5.3 Photonic Quantum models for Machine Learning

Our approach is based on using a *feedforward neural network* (FFNN), usually referred to as a *multilayer perceptron*, whose layers have been built utilising quantum photonic circuits in this particular instance. Eq.5.4 is the

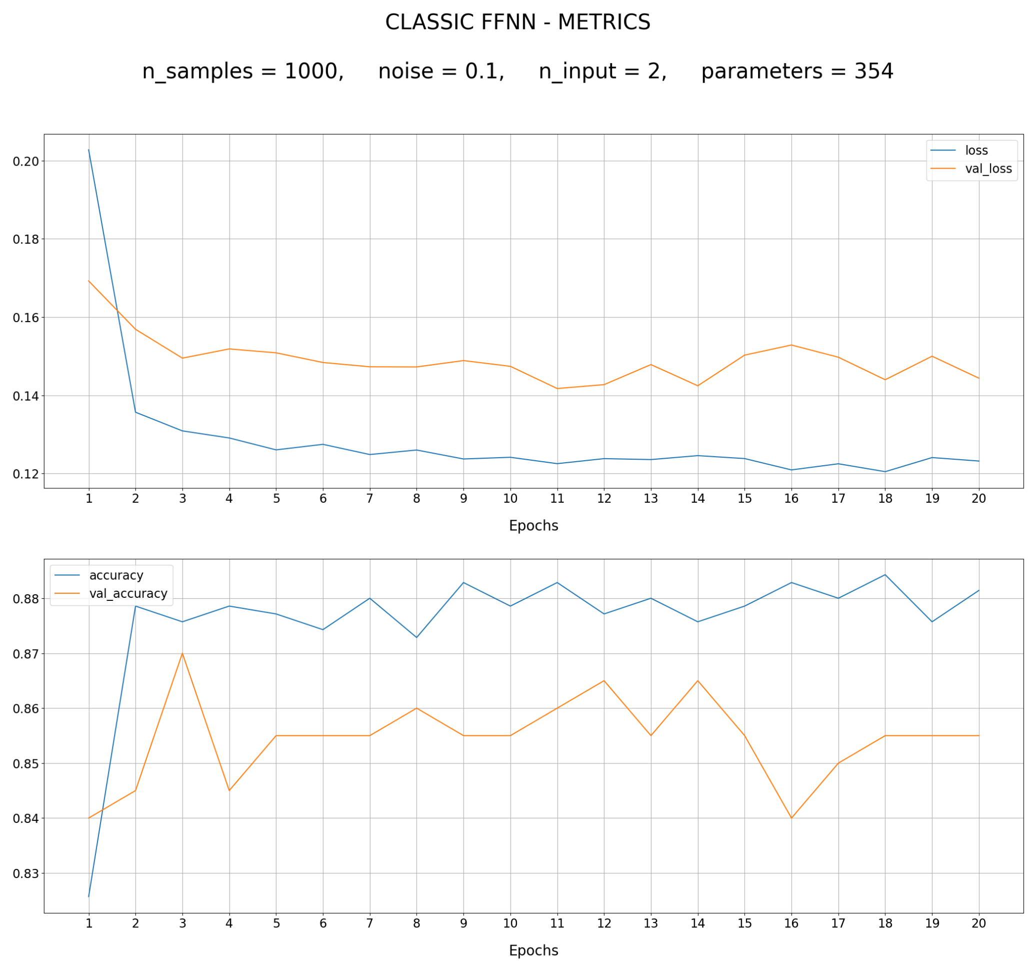


Figure 5.7: Metrics for an analogue FFNN

mathematical function that accurately depicts the transformation provided by an FFNN. The fundamental structure, however, is essentially the same: a horizontal stacked multi-layer arrangement with each layer being composed of an initial linear transformation (an *affine transformation*) and a nonlinear function called "activation" (Eq.5.5)

$$\bar{\mathbf{y}} = f\left(\bar{\mathbf{x}},\ \overline{\theta_W},\ \overline{\theta_b}\right) = \mathscr{L}_k \circ \mathscr{L}_{k-1} \circ\ \dots \mathscr{L}_2 \circ \mathscr{L}_1(\bar{\mathbf{x}}) \tag{5.4}$$

with $\overline{\mathbf{x}} \in \mathbb{R}^{n_1}$ is the input vector, $\overline{\theta_W} \in \mathbb{R}^{p_1}$, $\overline{\theta_b} \in \mathbb{R}^{p_2}$ are the parameter values' vectors and $(k, n_1, p_1, p_2) \in \mathbb{N}^4$.

$$\begin{gathered}\mathscr{L}_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}} \\ \mathscr{L}_i(\overline{\mathbf{x_i}}) = \varphi_i\left(W_i\overline{\mathbf{x_i}} + \overline{\mathbf{b_{i+1}}}\right)\end{gathered} \tag{5.5}$$

with $i = 1..k$ and $(n_1, n_2, ..., n_{k-1}, n_k, n_{k+1}) \in \mathbb{N}^{k+1}$. Moreover

- $W_i \in \mathbb{R}^{n_{i+1} \times n_i}$ is the i-th matrix for the i-th layer, called the *weight matrix* $\left(\text{containing } \overline{\theta_W}\right)$,
- $\overline{\mathbf{x_i}}$ is the i-th input vector,
- $\overline{\mathbf{b_i}} \in \mathbb{R}^{n_{i+1}}$ is the i-th vector, called the *bias vector* $\left(\text{containing } \overline{\theta_b}\right)$,
- φ_i is the non-linear activation function for the *i-th layer*

Usually, one is dealing with a *n-qubits* quantum ansatz, as said in previous subsection (Eq.5.1, 5.2, 5.3).

However, an alternative approach concerning one in Section 5.2 may be used to achieve the same results: an optical-quantum layer-based FFNN [128]. Thus, the followings (Eq.5.7) would be the transformation's attempt to ensure:

$$f : |\,\overline{\mathbf{x}}\,\rangle \to \left|\,\varphi\left(W\,\overline{\mathbf{x}} + \overline{\mathbf{b}}\right)\right\rangle \tag{5.6}$$

with $|\,\overline{\mathbf{x}}\,\rangle = \bigotimes_{i=1}^{n} |x_i\rangle$, n is the dimension of the feature space and $\overline{\mathbf{x}}$ the generic feature entering the system. Data can be encoded in position eigenstates.

The Singular Value Decomposition (SVD) theorem, which guarantees the factorisation of W into three matrices, two orthogonal and one as a positive diagonal matrix, may be used to decompose the matrix itself. That ensures the utilising of specific quantum gates that can mimic the behaviour determined by the corresponding matrices.

$$W = U \cdot \Sigma \cdot V^T \tag{5.7}$$

with

- $U^{n\times n}(\mathbb{R}) \mid U^{-1} = U^T$
- $\Sigma^{n\times m}(\mathbb{R}) \mid \Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_p)$
- $V^{m\times m}(\mathbb{R}) \mid V^{-1} = V^T$
- $(m, n) \in \mathbb{N}^2$

Regarding the physical implementation, squeezer gates may be utilised to obtain the diagonal transformation, while interferometers (a combination of beam splitters and rotation gates) can be employed to achieve the orthogonal ones. The addition operation using the bias vector, $\overline{\mathbf{b}}$, is then obtained by appending position displacement gates. Hanging a Kerr gate as the final circuital block might be a potential option to produce a non-linear transformation; this is the most common choice. Figure 5.8 shows the circuit diagram of a single layer relating to the input use of a single q-mode, while Figures 5.9 and 5.10 represent the situations with two q-modes and four q-modes, respectively. It can be said that this is another example

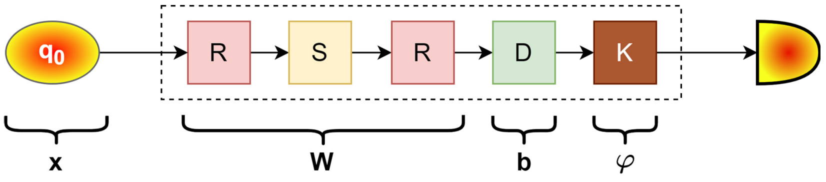


Figure 5.8: Circuital block diagram for a single layer, single q-mode, related to an optical-quantum layer-based FFNN

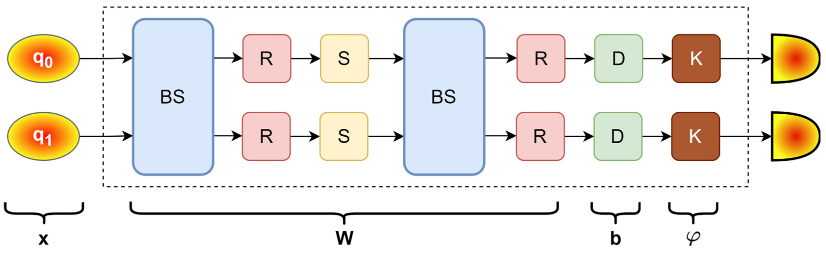


Figure 5.9: Circuital block diagram for a single layer, two q-modes, related to an optical-quantum layer-based FFNN

of quantum computation's applicability to machine learning. Concerning

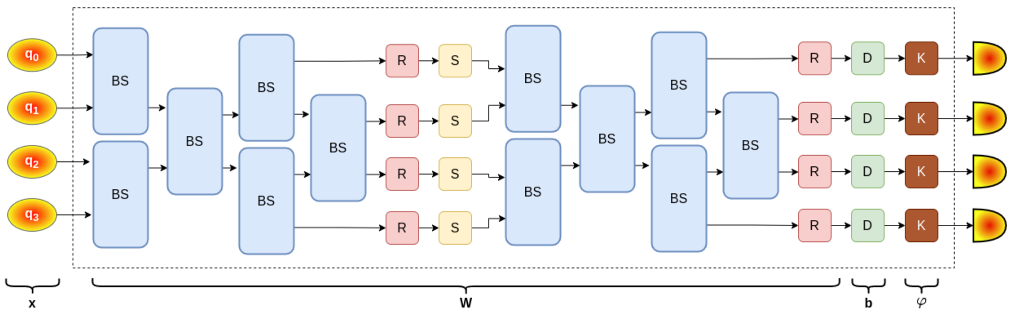


Figure 5.10: Circuital block diagram for a single layer, four q-modes, related to an optical-quantum layer-based FFNN

learning a straightforward classification task, here have specifically compared the abilities of a fundamental OQ-FFNN (Optical Quantum FFNN) with dense layers with an identical network only made up of photonic quantum components.

Since 2010, there has been a growing interest in applying machine learning techniques to quantum computing [108, 109, 110]. Early attempts to efficiently simulate the quantum world were prosperous to the extent that small physical systems (with few particles) were considered; the use of high-performance computers was necessary due to the large dimensions involved. The introduction of virtual and augmented reality to better explain the concepts of quantum mechanics is interesting from a didactic point of view [25]. Quantum machine learning [13, 14, 129] integrates quantum algorithms into machine learning programmes. The phrase is most frequently used to describe machine learning algorithms that analyse classical data and are executed on a quantum computer. Quantum machine learning uses qubits, quantum processes, or specialised quantum systems to speed up computation [112]. Machine learning techniques are used to calculate enormous amounts of data. The study of theoretical and practical parallels between particular physical processes and learning systems, such as neural networks, is known as quantum machine learning [114, 127].

A wide range of concepts with varying degrees of similarity to conventional neural networks are included in current QNN proposals [130, 131, 132]. The challenge of integrating the linear and non-linear parts and the unitary framework of quantum mechanics is at the heart of quantum neural network theory. Quantum neural networks are only one form of the most recent type of machine learning models implemented on quantum computers. They broadly use quantum phenomena like superposition, entanglement, and interference to exploit possible benefits such as quicker training and processing [133, 134, 135].

It has recently been proposed that the representation of quantum information need not be binary or discrete. Instead, it is also possible to leverage the innately "continuous" quantum features of matter, which would inevitably result in encoding the information in continuous variables (CV). The position and momentum of a particle are typical examples [136, 137, 138, 139]. It needs a quantum circuit with a universal layer structure so that one can manufacture any CV state with no more than polynomial complexity in order to do arbitrary proper transformations for the learning process by the machine. Therefore, the architecture to be selected must be composed of layers, with parameterised Gaussian and non-Gaussian gates present in each layer. The non-Gaussian gates provide the model with both non-linearity and universality [74, 140, 141]. The application of photonic quantum machine learning is proving effective in both traditional classification and regression issues, which is undoubtedly a fascinating finding [142, 143, 144, 145]. However, more importantly, it is fast improving our knowledge of quantum processes themselves [146, 147].

5.3.1 The system architecture

Three different types of OQFFNN, whose general structure is identical, have been tested; their differentiation comes from the distinct implementations of the quantum networks. All models consist of a series of quantum layers ($\mathscr{L}_i$) followed by a measurement apparatus whose outputs go into a classic dense layer that enables the binary categorisation of the input items (Figure 5.11). The Number Operator ($\hat{n}_i = \hat{a}_i^\dagger \, \hat{a}_i$, that is the sequential action of the annihilation and creation operators) has been selected as the observable in these experiments, and the measure will be its average value, $\langle \hat{n}_i \rangle$, for each i-th q-mode.

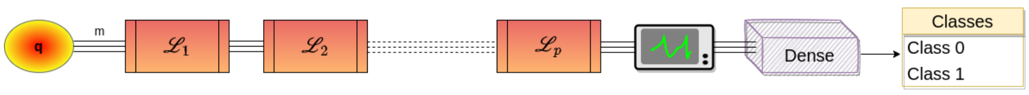


Figure 5.11: Common structure for the proposed OQFFNN (m is the input dimension)

The chosen dataset is a set of 2D points, so it can be represented as a matrix whose shape is $(number\_of\_features, 2)$.
One can now examine the three varieties of quantum layers:

1. The dataset features are entered into the network as values for the two parameters (ρ, φ) of a classical coherent state $|\alpha\rangle$, with $\alpha \in \mathbb{C} \mid \alpha = \rho \cdot e^{i \cdot \varphi}$. In order to allow a direct correspondence between the coordinates of the point and the geometric meaning of the parameters of the coherent state given in input, the pair of coordinates (x_1, x_2) that identify a point of the plane in an orthogonal Cartesian reference are transformed into polar coordinates (ρ_1, θ_1) before being transferred as input into the network. The quantum layer (Figure 5.12) is composed of a Rotational Gate (R), a Squeezing Gate (S), another Rotational Gate (R), a Displacement Gate (D) and a Kerr Gate (K).

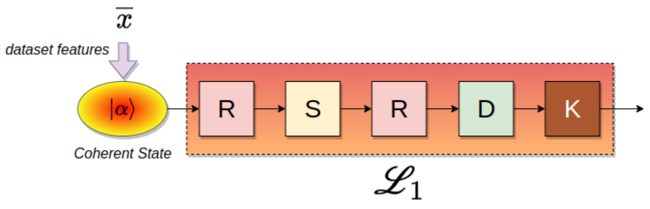


Figure 5.12: First type of layer

2. The features of the input dataset are represented in the network as the phases φ of an equal number of coherent states, whose amplitude ρ is arbitrarily set to 1, to improve the numerical simulation's efficiency without straying too far from the typical values of a physical implementation. The quantum layer (Figure 5.13) is composed of two Beam Splitters (BS), four Rotational Gates (R), two Squeezing Gates (S), two Displacement Gates (D) and two Kerr Gates (K).

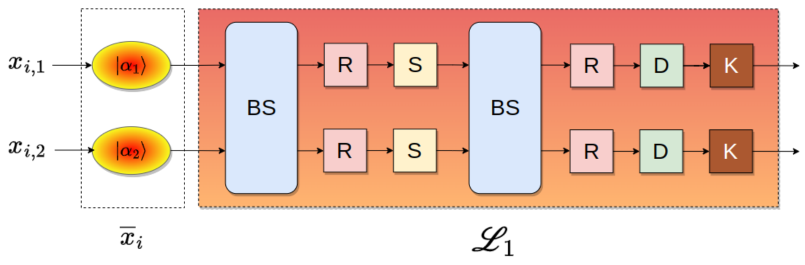


Figure 5.13: Second type of layer

3. The dataset features are entered into the network as values for the two parameters (ρ, φ) of a two-mode squeezing gate, whose inputs are Vacuum States. The quantum layer (Figure 5.14) is composed of four Rotational Gates (R), two single Squeezing Gates (S), two Displacement Gates (D) and a Cross-Kerr Gate (K).

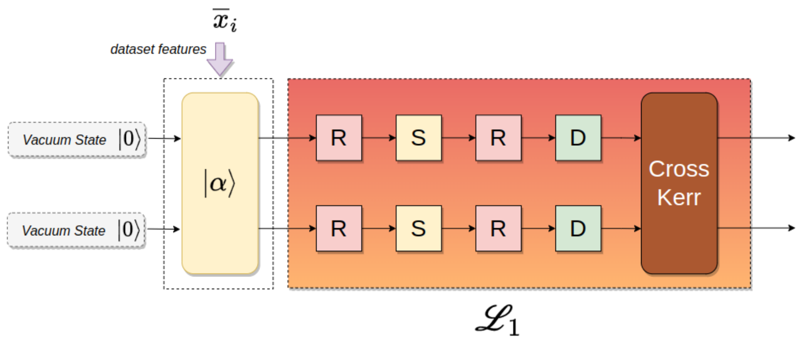


Figure 5.14: Third type of layer

5.3.2 Data extraction and processing

The *two-moons* database, which creates two interleaving semicircles of 2D points and is typical for the study of clustering and classification techniques, was chosen to solve the classification challenge.

The Python Scikit-Learn library's[6] dedicated function was used to create the dataset. The input domain is $[-2.0, 3.0] \times [-2.0, 2.5] \subset \mathbb{R}^2$. One thousand five hundred samples were randomly generated and uniformly distributed over the two classes.

[6]https://scikit-learn.org/stable/index.html

The number of samples in the dataset was divided in such a way as to ensure the following quotas: 75% for the training set, 15% for the validation set and the remaining 10% for the test set (Figure 5.15).

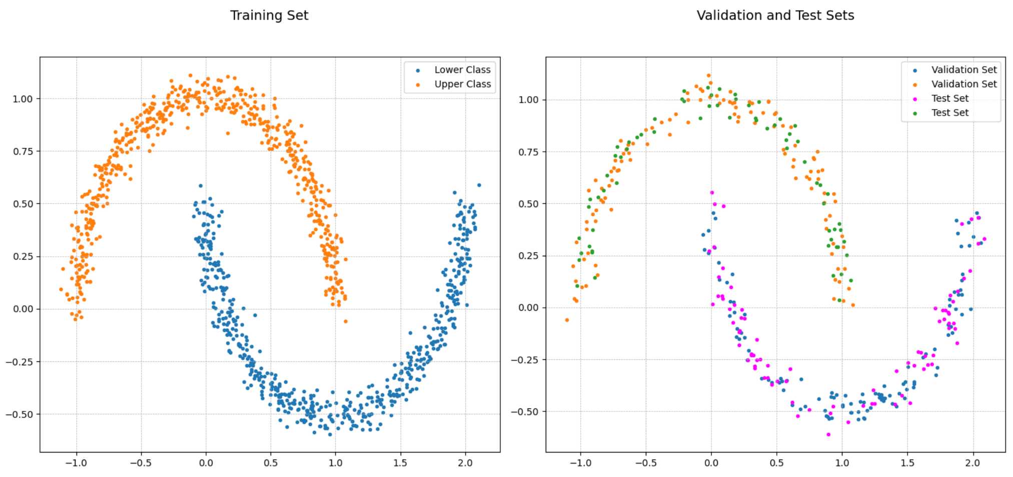


Figure 5.15: Dataset: Distribution of pattern points on the plane

Moreover, the features have been given as points of a plane in polar coordinates (Figure 5.16) to improve the operation of particular gates whose parameters operate on complex values (in set $\mathbb{C}$). The kind of input in the Cartesian or Polar form shall be defined for each model in section 5.3.3.

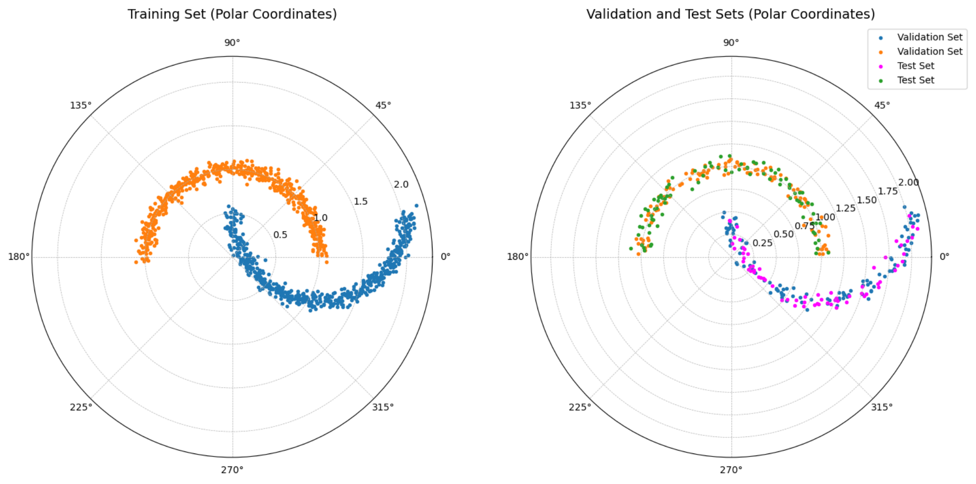


Figure 5.16: Dataset: Distribution of pattern points on the plane in polar form

5.3.3 Models' validation

All the models to be discussed have been implemented using the known open-source Python library, PennyLane[7], with StrawberryFields[8] backend for simulation, both by Xanadu [148, 149]. Both can be perfectly interfaced with the most famous Deep Learning frameworks, such as Keras, TensorFlow and PyTorch.

All tests and simulations were performed using the specialised open-source library for Deep Learning, Keras[9].

Every model has a traditional dense layer as the last layer (Figure 5.11), which serves as a classifier. Its input is the average number of photons processed by the quantum network, and its output is a float value between 0 and 1. The sigmoid function is utilised as its activation function. All the models were compiled using:

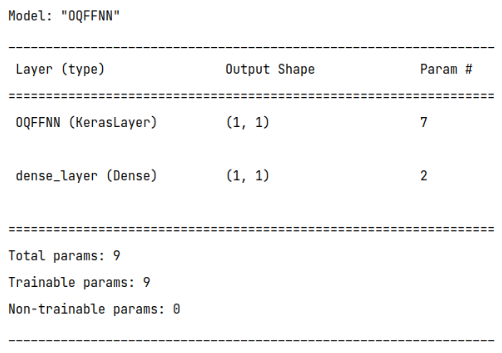
- *optimizer*: SGD (Stochastic Gradient Descent), with a learning rate equal to 0.01
- *loss function*: binary cross-entropy
- *metric*: accuracy

Model 1

The simulations were conducted initially using a simple structure, consisting of a single quantum layer, for a total of 9 parameters (Figure 5.17). Indeed, the number of parameters to be trained for the first model is given by 2 (parameters for the final Dense layer) and the product between the number of quantum layers to be used and 7 (the sum of the parameters of the single quantum gates making up the layer - see Appendix A for details).

Figures 5.18 and 5.19 show the metrics (LOSS: loss function on training set, ACCURACY: accuracy on training set, VAL_LOSS: loss function on validation set, VAL_ACCURACY: accuracy on validation set) for the Model n.1, trained for 50 epochs. The outcomes are superb. The optimal

[7]https://pennylane.ai/
[8]https://strawberryfields.ai/
[9]https://keras.io/

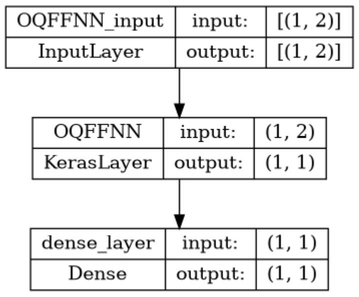


```
Model: "OQFFNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 OQFFNN (KerasLayer)         (1, 1)                    7

 dense_layer (Dense)         (1, 1)                    2

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

a ()



b

Figure 5.17: (a) Number of training parameters; (b) Model 1 - layers' scheme

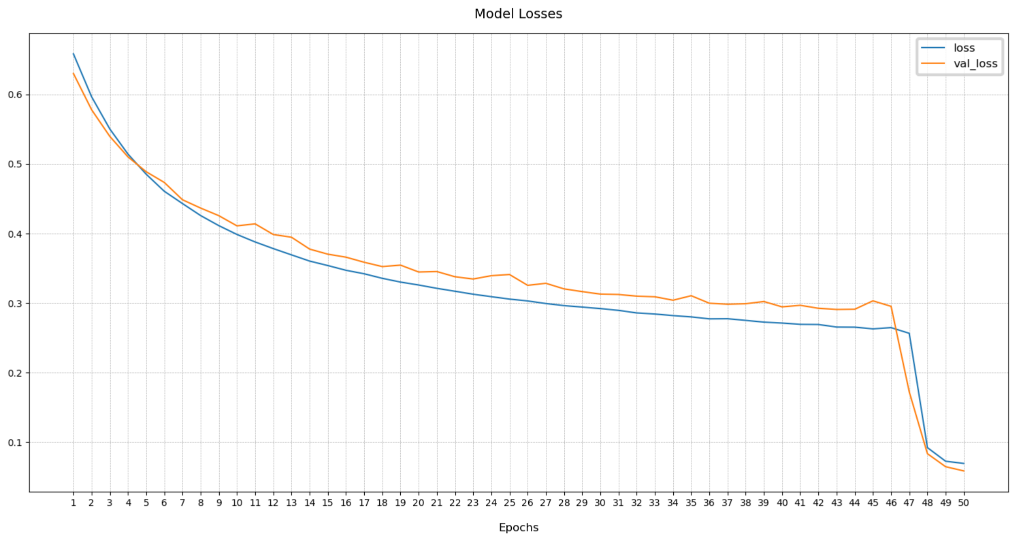


Figure 5.18: Loss and Validation Loss for Model 1

parameters' values[10] are displayed in Tab.5.3, while for the dense layer are $(w,\ b) = (-11.58,\ 6.2)$. The results provided in Figure 5.20 indicate what happens when the model is applied to the test set (150 samples), confirming the model's excellent performance (Test Accuracy: 100%).

Other simulations were run using more quantum gates in series, but none of them showed a discernible improvement over the single-gate experiment.

[10]Coefficient k is in mV^{-2}

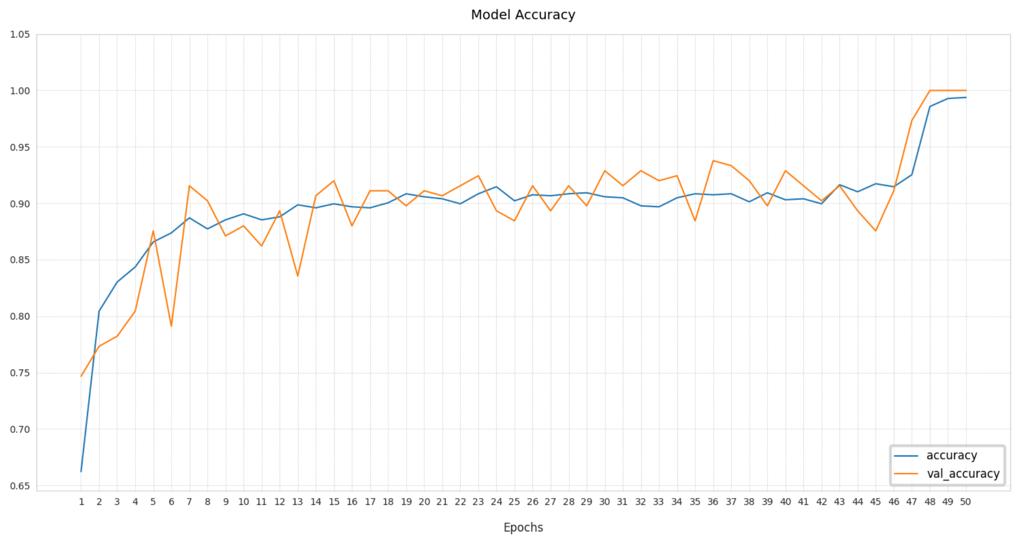


Figure 5.19: Accuracy and Validation Accuracy for Model 1

R	**S**		**R**	**D**		**K**
φ	ρ	φ	φ	ρ	φ	$\boldsymbol{k}$
1.28	0.46	0.64	2.56	0.28	0.06	$3.2 \cdot 10^{-7}$

Table 5.3: Optimised parameters' set for the quantum network

Model 2

The number of parameters to be trained for the model n.2 is given by 3 (parameters for the final Dense layer) and the product between the number of quantum layers to be used and 16 (the sum of the parameters of the single quantum gates making up the layer). Following the reasoning shown in the introductory part of this section, it was decided to utilise phaseless beamsplitters in this simulation so that each item in its representation matrix might have a real value.

The simulations were conducted initially using a simple structure, consisting of a single quantum layer, for a total of 19 parameters (Figure 5.21).

Figures 5.22 and 5.23 show the metrics (LOSS: loss function on training set, ACCURACY: accuracy on training set, VAL_LOSS: loss function on validation set, VAL_ACCURACY: accuracy on validation set) for the Model n.2, trained for 50 epochs. The outcomes are satisfactory. The optimal

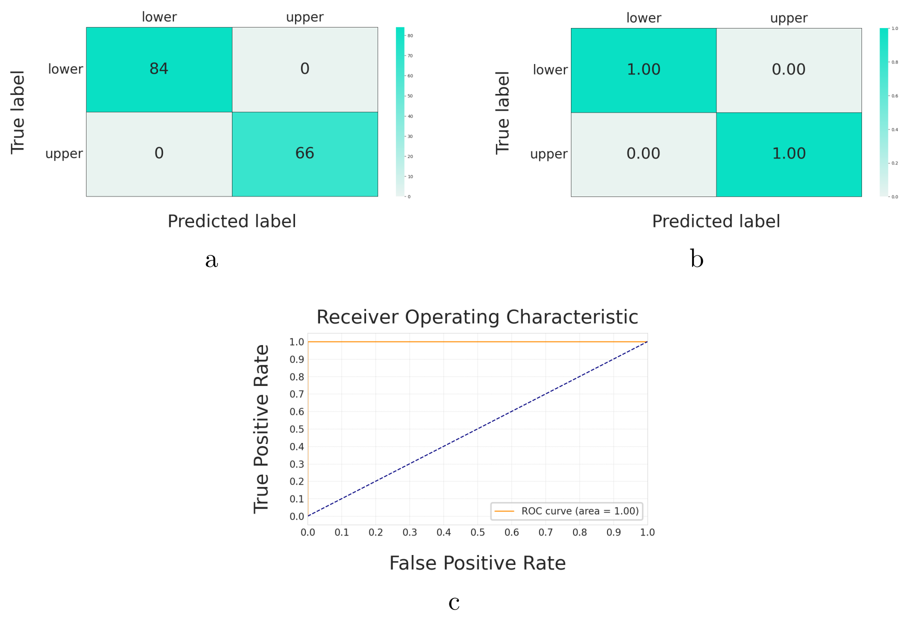
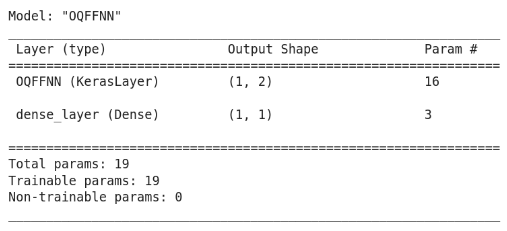


Figure 5.20: (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve

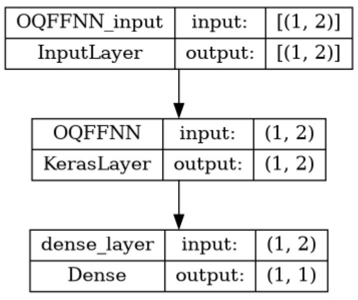


```
Model: "OQFFNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 OQFFNN (KerasLayer)         (1, 2)                    16

 dense_layer (Dense)         (1, 1)                    3

=================================================================
Total params: 19
Trainable params: 19
Non-trainable params: 0
_________________________________________________________________
```

a


OQFFNN_input | input: | [(1, 2)]
InputLayer | output: | [(1, 2)]
OQFFNN | input: | (1, 2)
KerasLayer | output: | (1, 2)
dense_layer | input: | (1, 2)
Dense | output: | (1, 1)

b

Figure 5.21: (a) Number of training parameters; (b) Model 2 - layers' scheme

parameters' values[11] are displayed in Tab.5.4, while for the dense layer, they are $w = (-7.64,\ 6.90)^T$ and $b = -0.33$. The results provided in Figure 5.24 indicate what happens when the model is applied to the test set (150 samples), confirming the model's good performance (Test Accuracy: 96.67%).

By starting with this model, it is feasible to increase the degree of cate-

[11]Coefficient k is in mV^{-2}

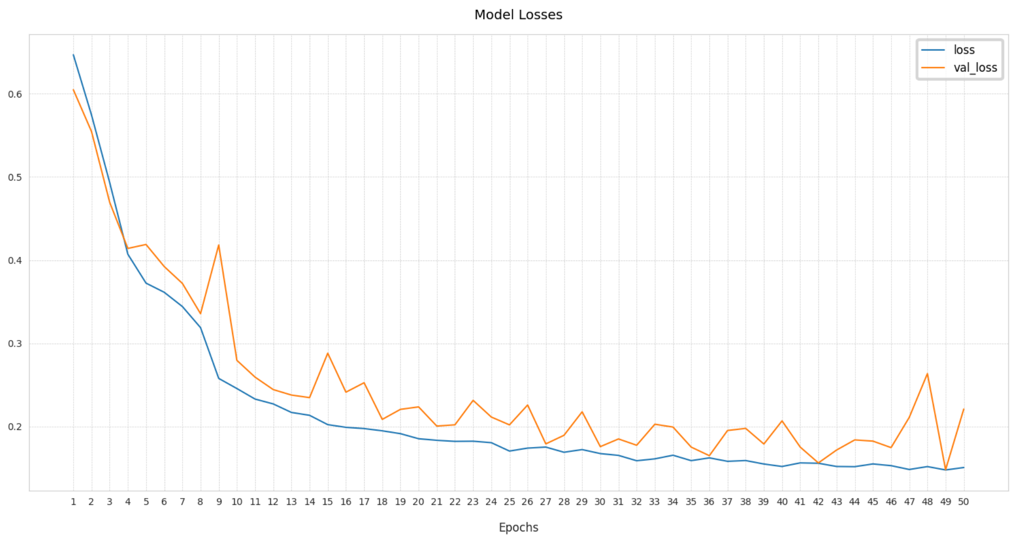


Figure 5.22: Loss and Validation Loss for Model 2

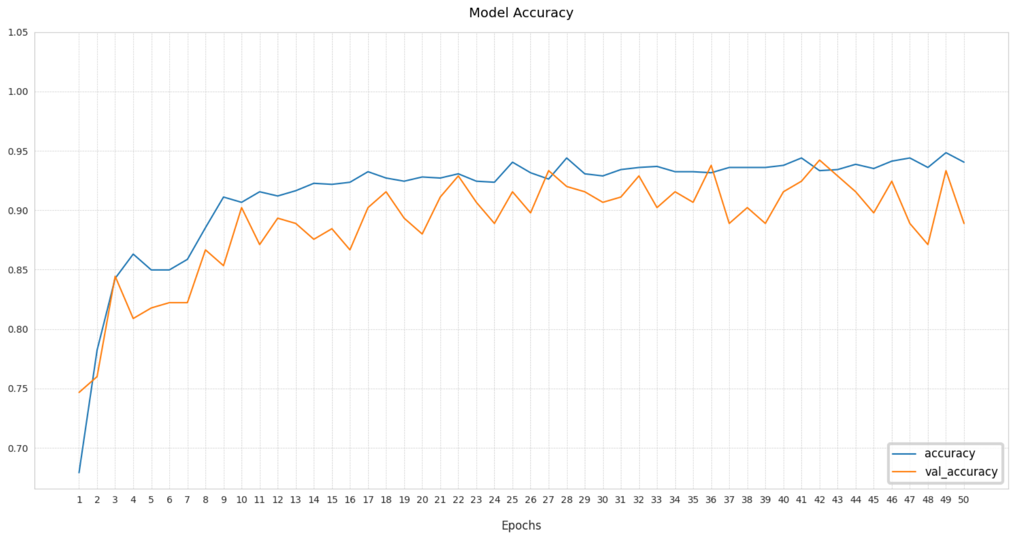


Figure 5.23: Accuracy and Validation Accuracy for Model 2

gorisation and moderate the oscillations during the solution research phase. The phase angle parameters will be added to the individual BeamSplitters, taking the reflectivity values from real to complex. The number of training parameters for the quantum layer is 18; the final number is 21.

Figures 5.25 and 5.26 show the usual metrics for the Model n.2, with

	BS		R	S		BS		R	D		K
Mode	θ	φ	φ	ρ	φ	θ	φ	φ	ρ	φ	$\boldsymbol{k}$
0	0.05	0.0	0.39	0.50	0.04	0.16	0.0	0.90	0.08	0.02	$-1.02 \cdot 10^{-15}$
1			2.60	0.02	0.01			0.20	0.88	1.10	$-1.49 \cdot 10^{-15}$

Table 5.4: Optimised parameters' set for the quantum network (Model without phase angles for beam splitters)

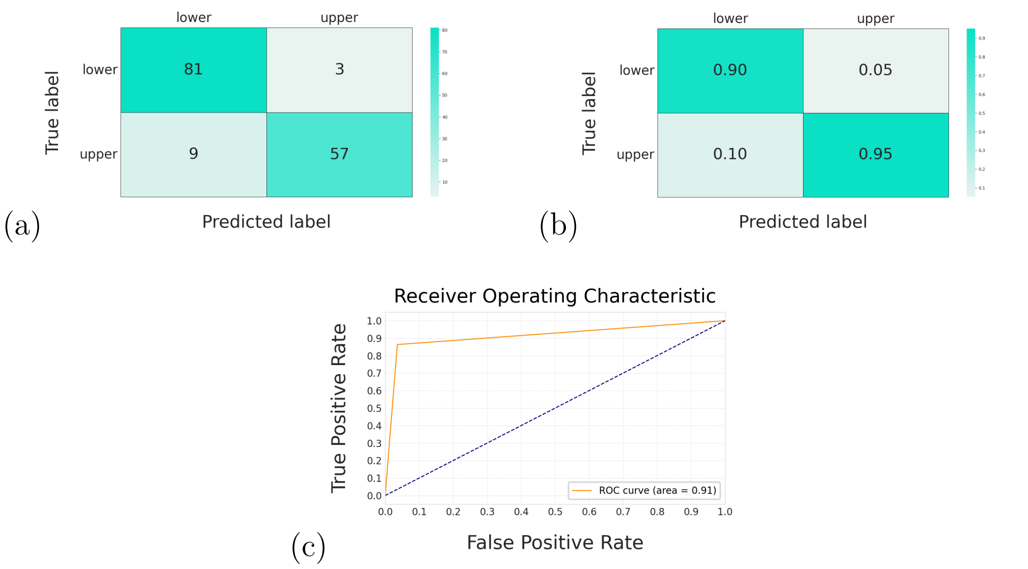


Figure 5.24: (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve

phase angles, trained for 50 epochs. The values of the best parameters[12] determined after network training are displayed in Table 5.5.

	BS		R	S		BS		R	D		K
Mode	θ	φ	φ	ρ	φ	θ	φ	φ	ρ	φ	$\boldsymbol{k}$
0	0.03	0.33	0.44	0.49	0.03	0.15	0.01	1.05	0.09	0.02	$9.30 \cdot 10^{-16}$
1			2.62	0.04	0.07			0.15	0.91	1.15	$6.66 \cdot 10^{-15}$

Table 5.5: Optimised parameters' set for the quantum network (Model with phase angles for beamsplitters)

The following values for the dense layer's parameters are gotten: $w = (-8.04,\ 7.08)^T$ and $b = -0.18$. The Test Set's scores (Figure 5.27) are likewise quite good, with an accuracy performance of 97.33%.

[12]Coefficient k is in mV^{-2}

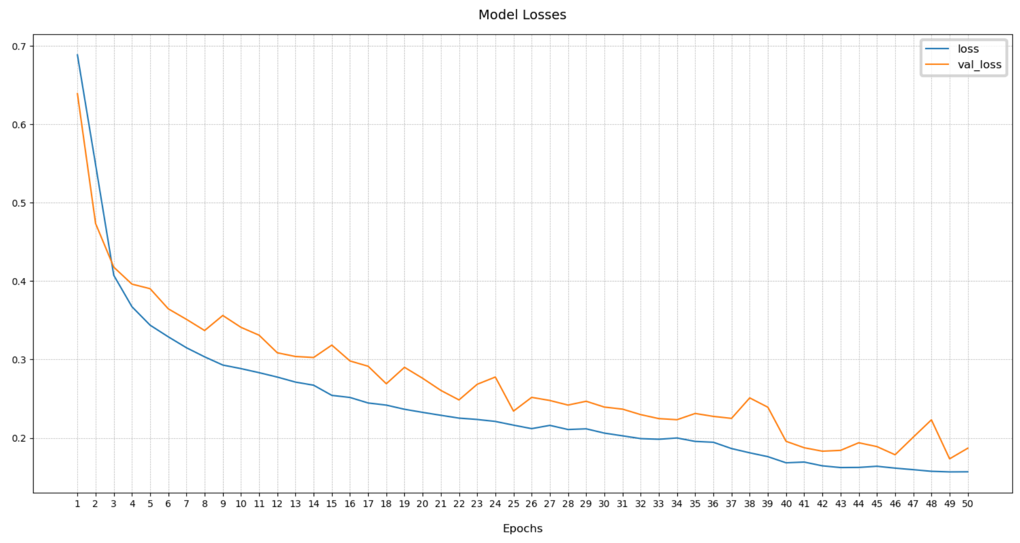


Figure 5.25: Loss and Validation Loss for Model 2 with phases

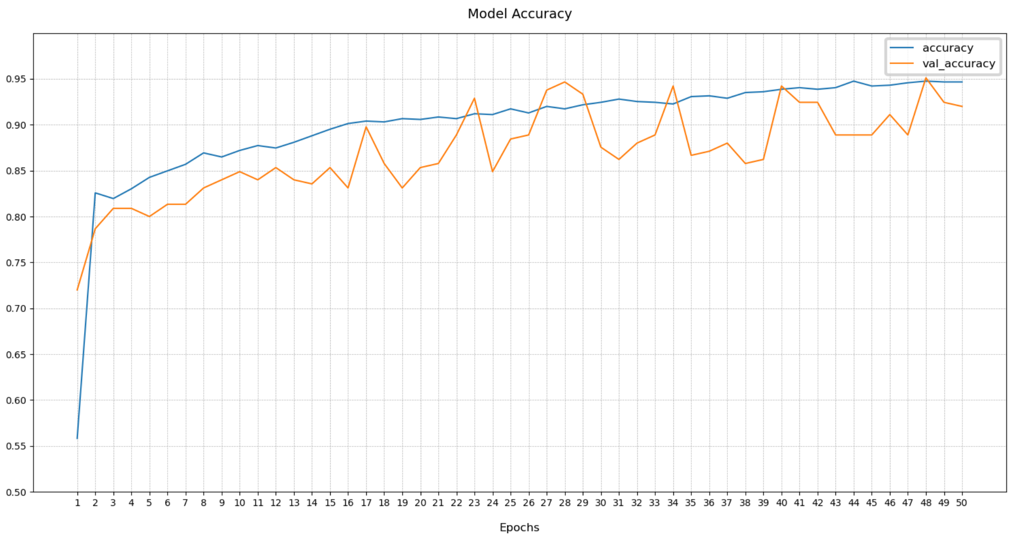
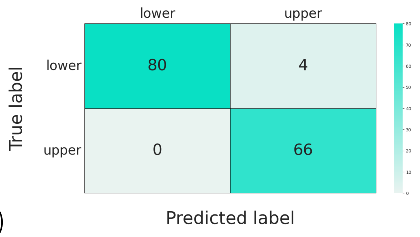
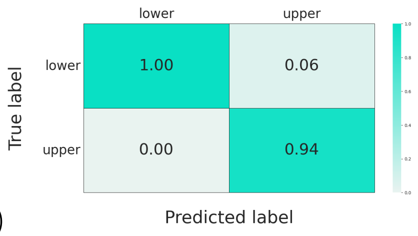


Figure 5.26: Accuracy and Validation Accuracy for Model 2 with phases

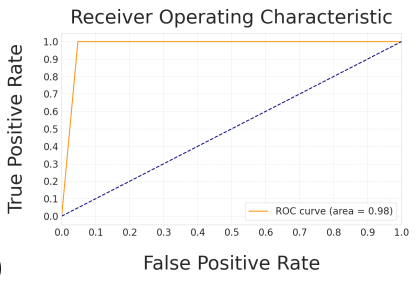
It was noticed that the model's performance did not meaningfully improve with the addition of more quantum layers to the network.



(a)



(b)



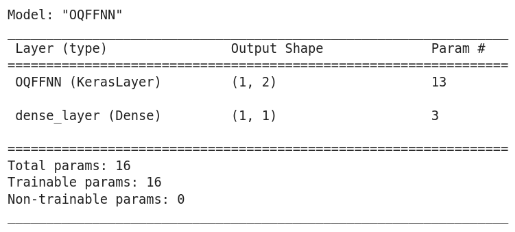
(c)

Figure 5.27: (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve

Model 3

A hybrid of the first two, the third kind of layer (Figure 5.14) contains two q-modes but does not employ beam splitters. The interaction between states is achieved by using a Two-Squeezing Gate, which serves the function of suitably encoding and modulating the input information, while the Cross-Kerr Gate is found in the final step of the Quantum Layer. They both carry out nonlinear transformations commonly utilised to generate entanglement in quantum models with continuous variables.

The simulations were conducted initially using a simple structure, consisting of a single quantum layer, for a total of 16 parameters (Figure 5.28): 13 belong to quantum layer.



```
Model: "OQFFNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 OQFFNN (KerasLayer)         (1, 2)                    13

 dense_layer (Dense)         (1, 1)                    3

=================================================================
Total params: 16
Trainable params: 16
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.28: Number of training parameters

Figures 5.29 and 5.30 show the metrics (LOSS: loss function on training set, ACCURACY: accuracy on training set, VAL_LOSS: loss function on validation set, VAL_ACCURACY: accuracy on validation set) for the Model n.3, trained for 50 epochs. The outcomes are satisfactory. The optimal

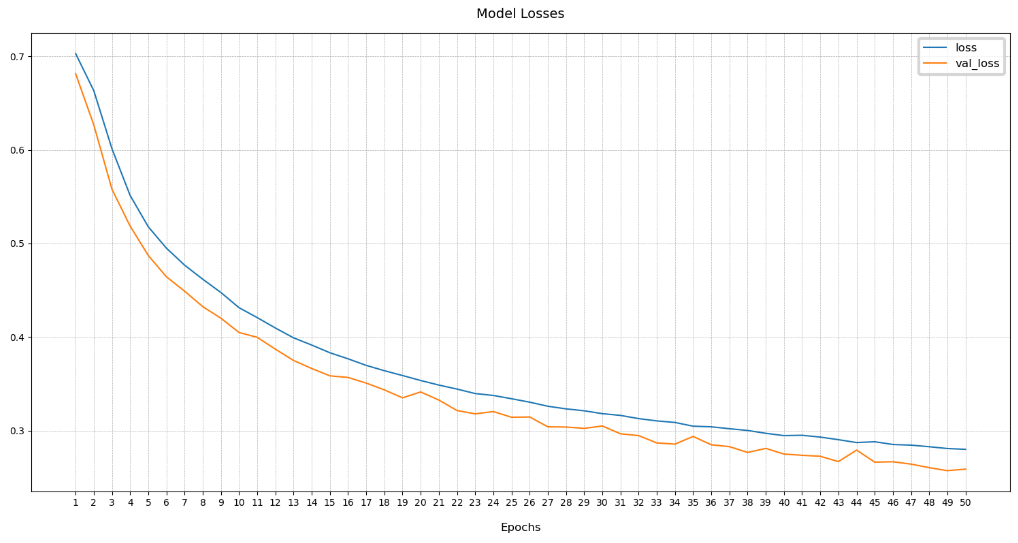


Figure 5.29: Loss and Validation Loss for Model 3

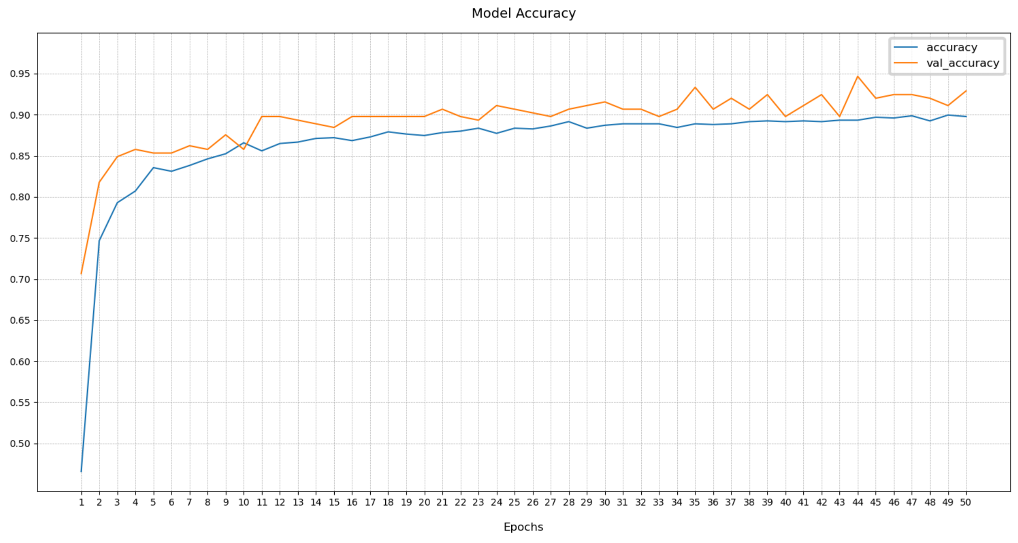


Figure 5.30: Accuracy and Validation Accuracy for Model 3

parameters' values[13] are displayed in Tab.5.6, while for the dense layer, they are $w = (-6.13, -7.14)^T$ and $b = 7.43$. The results provided in Figure 5.31 indicate what happens when the model is applied to the test set (150 samples), confirming the model's good performance (Test Accuracy: 89.33%).

	R	**S**		**R**	**D**		**CK**
Mode	ϕ	ρ	ϕ	ϕ	ρ	ϕ	κ
0	0.24	0.0	0.05	0.31	0.72	0.01	$1.58 \cdot 10^{-12}$
1	0.24	0.001	0.0	0.19	0.22	0.03	

Table 5.6: Optimised parameters' set for the quantum network

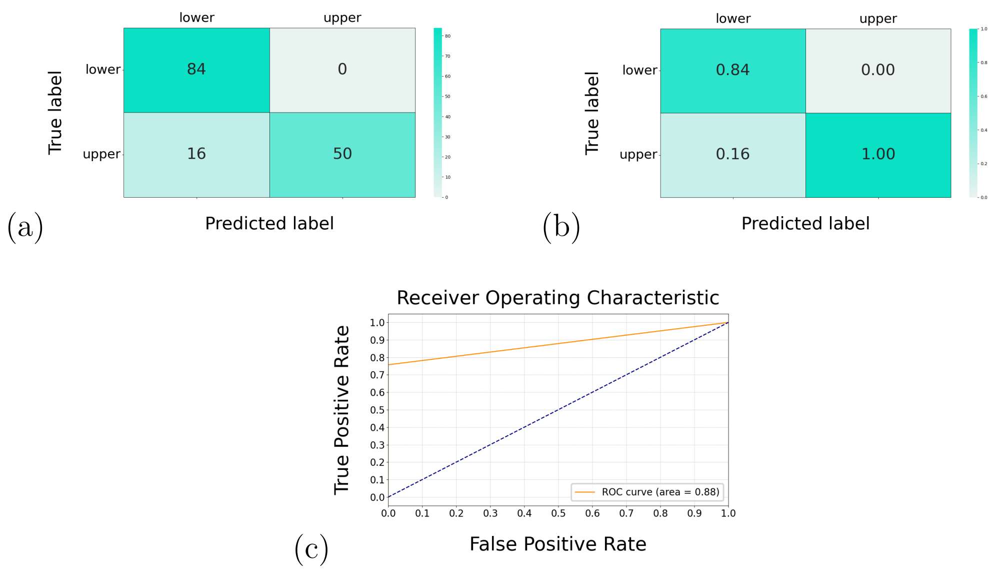


Figure 5.31: (a) Confusion Matrix for test set; (b) Normalised Confusion Matrix; (c) ROC Curve

Further investigations showed that increasing the number of layers leads to a partial but meaningful improvement in performance:

- Model 3, with 2 quantum layers
  - Total trainable parameters: 29
  - Test Accuracy: 91.33%

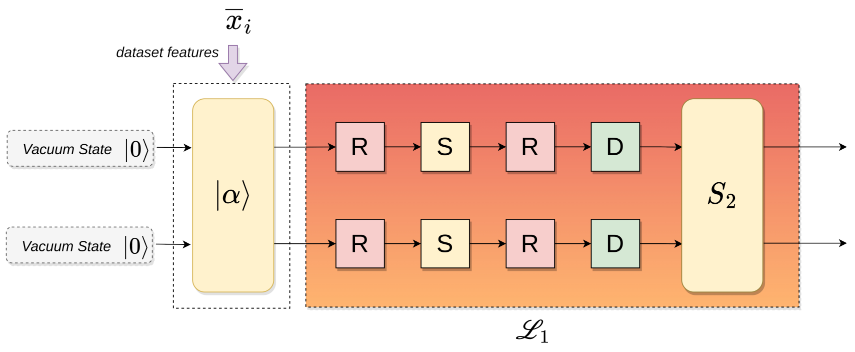
[13] Coefficient k is in mV^{-2}

  - ROC Curve Area: 0.90

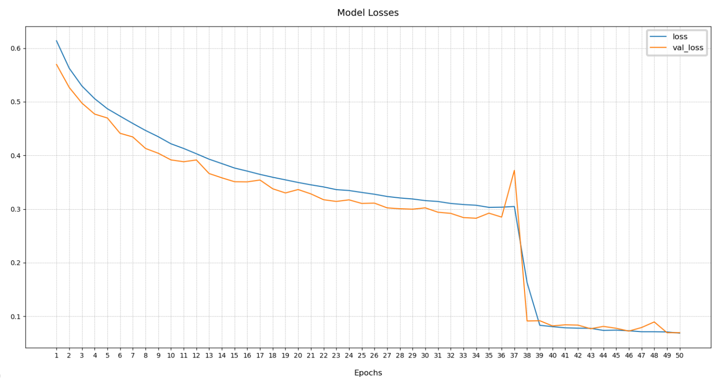
- Model 3, with 4 quantum layers
  - Total trainable parameters: 55
  - Test Accuracy: 99.33%
  - ROC Curve Area: 0.99

- Model 3, with 8 quantum layers
  - Total trainable parameters: 107
  - Test Accuracy: 100.00%
  - ROC Curve Area: 1.0

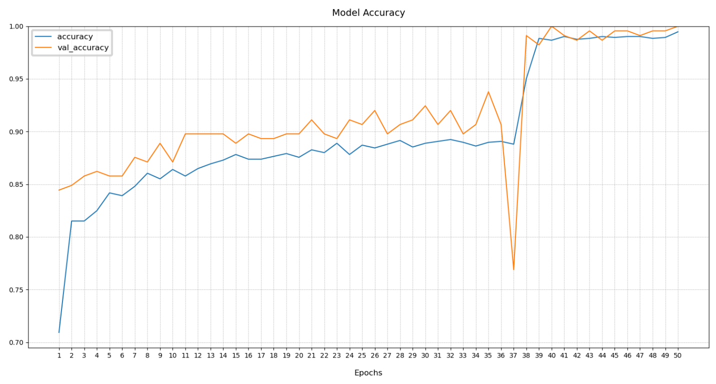
Furthermore, replacing the Cross-Kerr layer with a Two-Squeezing Gate yields an excellent result after 50 epochs of training (Figure 5.32), hitting 100% in the Accuracy Test, with only 17 trainable parameters.



(a)



(b)



(c)

Figure 5.32: (a) An interesting variant for Model 3; (b) Loss and Validation Loss; (c) Accuracy and Validation Accuracy

5.4 Conclusion

Variational circuits play a crucial role in the field of hybrid algorithms, serving as a link between classical and quantum computing. These circuits leverage a parameterised design that can be adjusted to achieve a specific outcome, like tuning a quantum circuit to simulate particular input-output relationships based on training data.

However, despite the apparent simplicity of this concept, the actual execution of variational circuits is incredibly complex. One of the most significant challenges is selecting the proper structure or "ansatz" for the efficient circuit regarding depth, width, and parameters while remaining robust and versatile. This challenge is similar to a fundamental problem in classical machine learning, which involves creating simple yet effective models. Nevertheless, the challenge continues beyond model selection. Training these quantum models differs from traditional approaches, as they are based on physical quantum algorithms rather than mathematical formulas. That raises questions about whether we can improve upon traditional numerical optimisation techniques. As a result, there are still some unanswered questions and uncertainties. In the context of quantum systems, choosing optimal parameterisation techniques is crucial to achieving high-quality results. It is imperative to consider whether classical iterative training strategies are adequate or if new techniques should be devised to suit the unique characteristics of quantum systems. As such, it needs to address whether existing parameterisation methods are fit for purpose or whether novel approaches need to be developed to meet the specific requirements of quantum systems. In essence, variational algorithms are not just another tool: they may hold the key to a whole new dimension in machine learning. They could pave the way for a broader paradigm in which we use physical devices as machine learning models, guided by our classical computers during training.

So, the aim of the research presented in this chapter was to delve deeper into the feasibility of using quantum computers for machine learning purposes in the NISQ era by attempting to train a simple network consisting of classical quantum gates (solution 1) and photonic quantum gates (solution 2). The study examined various essential aspects of a statistical classifier, such as the convergence of the proposed model, the accuracy of the result concerning a standard metric, the minimisation of the number of trainable parameters (i.e., the number of quantum gates required in the implemented circuit), and the scalability of the solution.

In order to perform the analysis, two different computation paradigms were considered: quantum computation using fermions and a bosonic model with continuous variables. The fermionic model is based on the behaviour of Fermi particles, such as electrons, while the bosonic model is based on the behaviour of bosons, such as photons. Although these two models differ significantly in nature and theoretical and practical modelling, the results obtained from this research and other similar recent studies [150] suggest that quantum computation can be used to advantage in the field of Machine Learning.

Both models operate on the same dataset, but the data semantics are different.

The first solution (Fermionic Quantum Machine Learning) leverages two different libraries to simulate the proposed network, which comprises rotation gates and Cnot. It produces consistent results as the noise level on the dataset changes. Only twenty epochs are needed to effectively train the network, which achieves a remarkable 100% recognition rate for samples belonging to the *Validation Set* when the noise level is low. Moreover, it achieves an impressive 85% recognition rate for very high noise levels. The progress of the Loss and Accuracy curves of the model suggests that it is robust to noise and immune to the drop-out phenomenon. Surprisingly, a similar network for classical Machine Learning (e.g., a simple FFNN) needs 354 trained weights (although relatively few for networks of this type) versus the 54 of the proposed solution. The degree of scalability is excellent, as the network structure is not tied by the size of the individual dataset. The most notable disadvantage during simulations is the high run-time, but typical of machine learning hybrid solutions (Classic + Quantum).

The second solution (Photonic Quantum Machine Learning) illustrates the investigation on the behaviour of a basic OQFFNN. The primary constituents of the quantum circuit are linear and non-linear optical components. The elements of the dataset modulate the incoming photons, transforming them into specific prepared quantum states. Here, the goal, in addition to the model's convergence and classification ability, was to limit the number of trainable parameters while retaining an accuracy of more than 80%. The three models had surprisingly comparable patterns, confirming their quality and validity and proving their efficacy. This investigation, which employed quantum mechanics, attempted to compare the behaviours emerging from several circuit implementations of the kernel method on a not overly complicated dataset. The results are excellent for all the models investigated, demonstrating the effectiveness of the circuit structures adopted. Some models have achieved accuracy levels of 100%, while others have never dropped below 90%, exhibiting classification robustness far beyond expectations.

The first model is straightforward: it has seven trained parameters. It receives in input a coherent state, the complex components modulated by the two input values of the data vector. It has an excellent answer: Once trained, it can recognize all new samples proposed with a 100% Test Accuracy.

The second model receives in input two consistent states, the phase of which is modulated by the input data. The confusion matrix and the ROC curve highlight a good model response, with a recognition accuracy of the test data of 91%.

The third model has a very similar structure to the second. Still, the interaction between the two photons in the final stage reduces the performance to 88%, a very high value, considering the low number of trained parameters (equal to 13). An excellent improvement is achieved by replacing the Cross-Kerr stadium with a Two-Squeezing Gate: the results obtained are comparable to those of Model 1.

The limit of the first model is its inability to be scalable if taken in its original version, as its input is a coherent state that depends solely on two parameters. Therefore, datasets with many features cannot be directly encoded in the network. The second and third models are more scalable than the first one. In these models, each input photon contains information about every feature in the dataset. However, minimising the number of gates in the structure is essential to ensure high performance. That is because other stages that stack horizontally can significantly degrade the model's performance.

The research aim to be done in the near future will concern the capacity to categorise increasingly complicated and structured data sets while leaving a suitably low number of gates needed to accomplish the task.

It is also desirable to transfer these investigations from quantum computer simulators to real quantum computers once the simple tests necessary to validate the model have been completed. Despite the excellent performance and optimisations gained so far of both hardware and software on the enormous amount of calculations to be performed, the simulators still need to be able to exploit the quantum peculiarities of matter which allow parallelism in the calculation, which is not classically attainable. Future investigations will focus on this front. This will provide insight into the problems arising from the physical implementation and the feasibility of the proposed solutions.

Conclusions

This dissertation has covered three important topics: Computing Mega Structures, Quantum Computing, and Quantum Machine Learning. Exploring these subjects in depth makes discovering many key insights and implications possible.

Computing Mega Structures are large-scale infrastructures like data centres and supercomputers. They play a critical role in processing and storing massive amounts of data. These scalable structures can perform resource-intensive tasks like simulations, data analysis, and artificial intelligence algorithms. As technology advances, Computing Mega Structures will become increasingly crucial for scientific discoveries, business operations, and computational efficiency.

Quantum Computing is a new paradigm in the field of computation. Using quantum mechanics, quantum computers can perform computations exponentially faster than classical computers. Quantum bits, or qubits, allow for the representation and manipulation of complex states, which enables the execution of quantum algorithms with superior efficiency for certain classes of problems. Quantum computing has immense potential for addressing computationally intractable problems, such as cryptography, optimisation, and material design.

Quantum Machine Learning integrates quantum computing principles into the field of machine learning. By exploiting the unique properties of quantum systems, quantum machine learning can potentially surpass classical machine learning models in terms of computational power and predictive capabilities. Quantum machine learning has the potential to unlock new frontiers in pattern recognition, data analysis, and optimisation, which can have profound implications across diverse domains.

Together, these three fields represent the forefront of computational research and offer immense opportunities for future exploration. However, there are still many challenges and technical hurdles to overcome. Further research and development efforts are needed to address error correction, scalability, and algorithm design issues. By embracing and advancing these technologies, it is possible to pave the way for a new era of computation, transforming industries, revolutionising problem-solving approaches, and shaping a more technologically advanced society.

Bibliography

[1] Gene H Golub and James M Ortega. *Scientific computing: an introduction with parallel computing.* Elsevier, 2014.

[2] David Padua. *Encyclopedia of parallel computing.* Springer Science & Business Media, 2011.

[3] Zryan Najat Rashid, Subhi RM Zebari, Karzan Hussein Sharif, and Karwan Jacksi. “Distributed cloud computing and distributed parallel computing: A review”. In: *2018 International Conference on Advanced Science and Engineering (ICOASE).* IEEE. 2018, pp. 167–172.

[4] Hiroyuki Takizawa and Hiroaki Kobayashi. “Hierarchical parallel processing of large scale data clustering on a PC cluster with GPU co-processing”. In: *The Journal of Supercomputing* 36.3 (2006), pp. 219–234.

[5] Volodymyr V Kindratenko, Jeremy J Enos, Guochun Shi, Michael T Showerman, Galen W Arnold, John E Stone, James C Phillips, and Wen-mei Hwu. “GPU clusters for high-performance computing”. In: *2009 IEEE International Conference on Cluster Computing and Workshops.* IEEE. 2009, pp. 1–8.

[6] Athanasios Chassiakos and George Rempis. “Evolutionary algorithm performance evaluation in project time-cost optimization”. In: *Journal of Soft Computing in Civil Engineering* 3.2 (2019), pp. 16–29.

[7] Narges Ghorbani, Alibakhsh Kasaeian, Ashkan Toopshekan, Leyli Bahrami, and Amin Maghami. “Optimizing a hybrid wind-PV-battery system using GA-PSO and MOPSO for reducing cost and increasing reliability”. In: *Energy* 154 (2018), pp. 581–591.

[8] Xuesong Yan, Hanmin Liu, Zhixin Zhu, and Qinghua Wu. “Hybrid genetic algorithm for engineering design problems”. In: *Cluster Computing* 20.1 (2017), pp. 263–275.

[9] Mahdi Abbasi, Milad Rafiee, Mohammad R Khosravi, Alireza Jolfaei, Varun G Menon, and Javad Mokhtari Koushyar. “An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems”. In: *Journal of cloud Computing* 9.1 (2020), p. 6.

[10] Marcello Caleffi, Angela Sara Cacciapuoti, and Giuseppe Bianchi. “Quantum Internet: From communication to distributed computing!”

In: *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication*. 2018, pp. 1–4.

[11] Juan Bermejo-Vega, Dominik Hangleiter, Martin Schwarz, Robert Raussendorf, and Jens Eisert. "Architectures for quantum simulation showing a quantum speedup". In: *Physical Review X* 8.2 (2018), p. 021010.

[12] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. "Toward the first quantum simulation with quantum speedup". In: *Proceedings of the National Academy of Sciences* 115.38 (2018), pp. 9456–9461.

[13] Damiano Perri, Marco Simonetti, Andrea Lombardi, Noelia Faginas Lago, and Osvaldo Gervasi. "Binary Classification of Proteins by a Machine Learning Approach". In: *Computational Science and Its Applications - ICCSA 2020 - 20th International Conference, Cagliari, Italy, July 1-4, 2020, Proceedings, Part VII*. Vol. 12255. Lecture Notes in Computer Science. Springer, 2020, pp. 549–558. DOI: 10.1007/978-3-030-58820-5_41. URL: https://doi.org/10.1007/978-3-030-58820-5%5C_41.

[14] Priscilla Benedetti, Damiano Perri, Marco Simonetti, Osvaldo Gervasi, Gianluca Reali, and Mauro Femminella. "Skin Cancer Classification Using Inception Network and Transfer Learning". In: *Computational Science and Its Applications - ICCSA 2020 - 20th International Conference, Cagliari, Italy, July 1-4, 2020, Proceedings, Part I*. Vol. 12249. Lecture Notes in Computer Science. Springer, 2020, pp. 536–545. DOI: 10.1007/978-3-030-58799-4_39. URL: https://doi.org/10.1007/978-3-030-58799-4%5C_39.

[15] Giulio Biondi, Valentina Franzoni, Osvaldo Gervasi, and Damiano Perri. "An Approach for Improving Automatic Mouth Emotion Recognition". In: *Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part I*. Vol. 11619. Lecture Notes in Computer Science. Springer, 2019, pp. 649–664. DOI: 10.1007/978-3-030-24289-3_48. URL: https://doi.org/10.1007/978-3-030-24289-3%5C_48.

[16] Valentina Franzoni, Sergio Tasso, Simonetta Pallottelli, and Damiano Perri. "Sharing Linkable Learning Objects with the Use of Metadata and a Taxonomy Assistant for Categorization". In: *Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part II*. Vol. 11620. Lecture Notes in Computer Science. Springer, 2019, pp. 336–348. DOI: 10.1007/978-3-030-24296-1_28. URL: https://doi.org/10.1007/978-3-030-24296-1%5C_28.

[17] Kun-Long Ho, Y-Y Hsu, and C-C Yang. "Short term load forecasting using a multilayer neural network with an adaptive learning algo-

rithm". In: *IEEE Transactions on Power Systems* 7.1 (1992), pp. 141–149.

[18] Yong Li, Yang Fu, Hui Li, and Si-Wen Zhang. "The improved training algorithm of back propagation neural network with self-adaptive learning rate". In: *2009 International Conference on Computational Intelligence and Natural Computing*. Vol. 1. IEEE. 2009, pp. 73–76.

[19] Spiros V Georgakopoulos and Vassilis P Plagianakos. "A novel adaptive learning rate algorithm for convolutional neural network training". In: *International Conference on Engineering Applications of Neural Networks*. Springer. 2017, pp. 327–336.

[20] Shaibal Chakrabarty and Daniel W Engels. "Secure Smart Cities Framework Using IoT and AI". In: *2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. IEEE. 2020, pp. 1–6.

[21] Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, and Shuming Shi. "Generating multiple diverse responses for short-text conversation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6383–6390.

[22] Daniel W Surry, Robert M Gray Jr, and James R Stefurak. *Technology Integration in Higher Education: Social and Organizational Aspects: Social and Organizational Aspects*. IGI Global, 2010.

[23] G. Bulman and R.W. Fairlie. "Chapter 5 - Technology and Education: Computers, Software, and the Internet". In: vol. 5. Handbook of the Economics of Education. Elsevier, 2016, pp. 239–280. DOI: https://doi.org/10.1016/B978-0-444-63459-7.00005-1. URL: https://www.sciencedirect.com/science/article/pii/B9780444634597000051.

[24] Sumedha Chauhan. "A meta-analysis of the impact of technology on learning effectiveness of elementary students". In: *Computers & Education* 105 (2017), pp. 14–30.

[25] Marco Simonetti, Damiano Perri, Natale Amato, and Osvaldo Gervasi. "Teaching Math with the Help of Virtual Reality". In: *Computational Science and Its Applications - ICCSA 2020 - 20th International Conference, Cagliari, Italy, July 1-4, 2020, Proceedings, Part VII*. Vol. 12255. Lecture Notes in Computer Science. Springer, 2020, pp. 799–809. DOI: 10.1007/978-3-030-58820-5_57. URL: https://doi.org/10.1007/978-3-030-58820-5%5C_57.

[26] Manuel Au-Yong-Oliveira, Ramiro Gonçalves, José Martins, and Frederico Branco. "The social impact of technology on millennials and consequences for higher education and leadership". In: *Telematics and Informatics* 35.4 (2018), pp. 954–963. ISSN: 0736-5853. DOI: https://doi.org/10.1016/j.tele.2017.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0736585317303465.

[27] Damiano Perri, Marco Simonetti, Sergio Tasso, and Osvaldo Gervasi. "Learning Mathematics in an Immersive Way". In: *Software Usability*. IntechOpen, 2021.

[28] Anne Marie McEwan. *Smart working: Creating the next wave*. Routledge, 2013, p. 288. ISBN: 9781409404569. URL: https://www.routledge.com/Smart-Working-Creating-the-Next-Wave/McEwan/p/book/9781409404569#.

[29] Marta Angelici and Paola Profeta. "Smart-working: work flexibility without constraints". In: CESifo Working Paper N. 8165 (2020). URL: https://ssrn.com/abstract=3556304.

[30] Damiano Perri, Marco Simonetti, Sergio Tasso, Federico Ragni, and Osvaldo Gervasi. "Implementing a Scalable and Elastic Computing Environment Based on Cloud Containers". In: *Computational Science and Its Applications – ICCSA 2021*. Ed. by Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Chiara Garau, Ivan Blečić, David Taniar, Bernady O. Apduhan, Ana Maria A. C. Rocha, Eufemia Tarantino, and Carmelo Maria Torre. Cham: Springer International Publishing, 2021, pp. 676–689. ISBN: 978-3-030-86653-2.

[31] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth Schoen, and Brad Warren. "Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 2473–2487. ISBN: 9781450367479. DOI: 10.1145/3319535.3363192. URL: https://doi.org/10.1145/3319535.3363192.

[32] Kamran Karimi, Neil G. Dickson, and Firas Hamze. "A Performance Comparison of CUDA and OpenCL". In: *CoRR* abs/1005.2581 (2010). arXiv: 1005.2581. URL: http://arxiv.org/abs/1005.2581.

[33] Jianbin Fang, Ana Lucia Varbanescu, and Henk Sips. "A Comprehensive Performance Comparison of CUDA and OpenCL". In: *2011 International Conference on Parallel Processing*. 2011, pp. 216–225. DOI: 10.1109/ICPP.2011.45.

[34] Abu Asaduzzaman, Alec Trent, S. Osborne, C. Aldershof, and Fadi N. Sibai. "Impact of CUDA and OpenCL on Parallel and Distributed Computing". In: *2021 8th International Conference on Electrical and Electronics Engineering (ICEEE)*. 2021, pp. 238–242. DOI: 10.1109/ICEEE52452.2021.9415927.

[35] Osvaldo Gervasi, Diego Russo, and Flavio Vella. "The AES Implantation Based on OpenCL for Multi/many Core Architecture". In: *2010 International Conference on Computational Science and Its Applications*. 2010, pp. 129–134. DOI: 10.1109/ICCSA.2010.44.

[36] Jesús Pérez Serrano, Edans Flavius De Oliveira Sandes, Alba Cristina Magalhaes Alves de Melo, and Manuel Ujaldón. "Smith-Waterman

Acceleration in Multi-GPUs: A Performance per Watt Analysis". In: *Bioinformatics and Biomedical Engineering.* Cham: Springer International Publishing, 2017, pp. 512–523. ISBN: 978-3-319-56154-7.

[37] Rod Burns, John Lawson, Duncan McBain, and Daniel Soutar. "Accelerated Neural Networks on OpenCL Devices Using SYCL-DNN". In: *Proceedings of the International Workshop on OpenCL.* IWOCL'19. Boston, MA, USA: Association for Computing Machinery, 2019. ISBN: 9781450362306. DOI: 10.1145/3318170.3318183. URL: https://doi.org/10.1145/3318170.3318183.

[38] Francesca Santucci, Federico Frenguelli, Alessandro De Angelis, Ilaria Cuccaro, Damiano Perri, and Marco Simonetti. "An Immersive Open Source Environment Using Godot". In: *Computational Science and Its Applications - ICCSA 2020 - 20th International Conference, Cagliari, Italy, July 1-4, 2020, Proceedings, Part VII.* Vol. 12255. Lecture Notes in Computer Science. Springer, 2020, pp. 784–798. DOI: 10.1007/978-3-030-58820-5_56. URL: https://doi.org/10.1007/978-3-030-58820-5%5C_56.

[39] Damiano Perri, Paolo Sylos Labini, Osvaldo Gervasi, Sergio Tasso, and Flavio Vella. "Towards a Learning-Based Performance Modeling for Accelerating Deep Neural Networks". In: *Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part I.* Vol. 11619. Lecture Notes in Computer Science. Springer, 2019, pp. 665–676. DOI: 10.1007/978-3-030-24289-3_49. URL: https://doi.org/10.1007/978-3-030-24289-3%5C_49.

[40] Saeed S. Alahmari, Dmitry B. Goldgof, Peter R. Mouton, and Lawrence O. Hall. "Challenges for the Repeatability of Deep Learning Models". In: *IEEE Access* 8 (2020), pp. 211860–211868. DOI: 10.1109/ACCESS.2020.3039833.

[41] Damiano Perri, Marco Simonetti, Andrea Lombardi, Noelia Faginas-Lago, and Osvaldo Gervasi. "A New Method for Binary Classification of Proteins with Machine Learning". In: *Computational Science and Its Applications – ICCSA 2021.* Ed. by Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Chiara Garau, Ivan Blečić, David Taniar, Bernady O. Apduhan, Ana Maria A. C. Rocha, Eufemia Tarantino, and Carmelo Maria Torre. Cham: Springer International Publishing, 2021, pp. 388–397. ISBN: 978-3-030-87016-4.

[42] Richard P Feynman. "Simulating physics with computers". In: *Feynman and computation.* CRC Press, 2018, pp. 133–153.

[43] Deutsch, D. "Quantum theory, the Church–Turing principle and the universal quantum computer". In: *Proc. R. Soc. Lond.* 400 (1985). https://doi.org/10.1098/rspa.1985.0070, pp. 97–117.

[44] David Deutsch. "Quantum theory, the Church–Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society*

of London. A. Mathematical and Physical Sciences 400.1818 (1985), pp. 97–117.

[45] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019), pp. 505–510.

[46] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. "Quantum computational advantage using photons". In: *Science* 370.6523 (2020), pp. 1460–1463.

[47] Michael A Nielsen and Isaac L Chuang. "Quantum computation and quantum information". In: *Phys. Today* 54.2 (2001), p. 60.

[48] N David Mermin. *Quantum computer science: an introduction.* Cambridge University Press, 2007.

[49] Benjamin Schumacher. "Quantum coding". In: *Physical Review A* 51.4 (1995), p. 2738.

[50] Gregg Jaeger. *Quantum information.* Springer, 2007.

[51] Lajos Diósi. *A short course in quantum information theory: an approach from theoretical physics.* Vol. 827. Springer, 2011.

[52] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing.* OUP Oxford, 2006.

[53] Colin P Williams, Scott H Clearwater, et al. *Explorations in quantum computing.* Springer, 1998.

[54] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information.* 2002.

[55] Richard P Feynman. "Quantum mechanical computers". In: *Optics news* 11.2 (1985), pp. 11–20.

[56] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. "Elementary gates for quantum computation". In: *Physical review A* 52.5 (1995), p. 3457.

[57] Adriano Barenco. "A universal two-bit gate for quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 449.1937 (1995), pp. 679–683.

[58] Isaac L Chuang and Yoshihisa Yamamoto. "Simple quantum computer". In: *Physical Review A* 52.5 (1995), p. 3489.

[59] Márton Karácsony, László Oroszlány, and Zoltán Zimborás. "Efficient qudit based scheme for photonic quantum computing". In: *arXiv preprint arXiv:2302.07357* (2023).

[60] Suren A Fldzhyan, M Yu Saygin, and Sergei P Kulik. "Optimal design of error-tolerant reprogrammable multiport interferometers". In: *Optics Letters* 45.9 (2020), pp. 2632–2635.

[61] Niccolo Somaschi, Valerian Giesz, Lorenzo De Santis, JC Loredo, Marcelo P Almeida, Gaston Hornecker, S Luca Portalupi, Thomas

Grange, Carlos Anton, Justin Demory, et al. “Near-optimal single-photon sources in the solid state”. In: *Nature Photonics* 10.5 (2016), pp. 340–345.

[62] Hélène Ollivier, Ilse Maillette de Buy Wenniger, Sarah Thomas, Stephen C Wein, Abdelmounaim Harouri, Guillaume Coppola, Paul Hilaire, Clément Millet, Aristide Lemaitre, Isabelle Sagnes, et al. “Reproducibility of high-performance quantum dot single-photon sources”. In: *ACS photonics* 7.4 (2020), pp. 1050–1059.

[63] Juan C Loredo, Nor A Zakaria, Niccolo Somaschi, Carlos Anton, Lorenzo De Santis, Valerian Giesz, Thomas Grange, Matthew A Broome, Olivier Gazzano, Guillaume Coppola, et al. “Scalable performance in solid-state single-photon sources”. In: *Optica* 3.4 (2016), pp. 433–440.

[64] Max Tillmann, Borivoje Dakić, René Heilmann, Stefan Nolte, Alexander Szameit, and Philip Walther. “Experimental boson sampling”. In: *Nature photonics* 7.7 (2013), pp. 540–544.

[65] Craig S Hamilton, Regina Kruse, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, and Igor Jex. “Gaussian boson sampling”. In: *Physical review letters* 119.17 (2017), p. 170501.

[66] Justin B Spring, Benjamin J Metcalf, Peter C Humphreys, W Steven Kolthammer, Xian-Min Jin, Marco Barbieri, Animesh Datta, Nicholas Thomas-Peter, Nathan K Langford, Dmytro Kundys, et al. “Boson sampling on a photonic chip”. In: *Science* 339.6121 (2013), pp. 798–801.

[67] Daniel J Brod, Ernesto F Galvão, Andrea Crespi, Roberto Osellame, Nicolò Spagnolo, and Fabio Sciarrino. “Photonic implementation of boson sampling: a review”. In: *Advanced Photonics* 1.3 (2019), pp. 034001–034001.

[68] Nicolas Heurtel, Andreas Fyrillas, Grégoire de Gliniasty, Raphaël Le Bihan, Sébastien Malherbe, Marceau Pailhas, Eric Bertasi, Boris Bourdoncle, Pierre-Emmanuel Emeriau, Rawad Mezher, Luka Music, Nadia Belabas, Benoît Valiron, Pascale Senellart, Shane Mansfield, and Jean Senellart. “Perceval: A Software Platform for Discrete Variable Photonic Quantum Computing”. In: *Quantum* 7 (Feb. 2023), p. 931. ISSN: 2521-327X. DOI: 10.22331/q-2023-02-21-931. URL: https://doi.org/10.22331/q-2023-02-21-931.

[69] Michael Reck, Anton Zeilinger, Herbert J Bernstein, and Philip Bertani. “Experimental realization of any discrete unitary operator”. In: *Physical review letters* 73.1 (1994), p. 58.

[70] William R Clements, Peter C Humphreys, Benjamin J Metcalf, W Steven Kolthammer, and Ian A Walmsley. “Optimal design for universal multiport interferometers”. In: *Optica* 3.12 (2016), pp. 1460–1465.

[71] Hubert de Guise, Olivia Di Matteo, and Luis L Sánchez-Soto. "Simple factorization of unitary transformations". In: *Physical Review A* 97.2 (2018), p. 022328.

[72] Shreya P Kumar and Ish Dhand. "Unitary matrix decompositions for optimal and modular linear optics architectures". In: *Journal of Physics A: Mathematical and Theoretical* 54.4 (2021), p. 045301.

[73] Timothy C Ralph, Nathan K Langford, TB Bell, and AG White. "Linear optical controlled-NOT gate in the coincidence basis". In: *Physical Review A* 65.6 (2002), p. 062324.

[74] Emanuel Knill, Raymond Laflamme, and Gerald J Milburn. "A scheme for efficient quantum computation with linear optics". In: *nature* 409.6816 (2001), pp. 46–52.

[75] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

[76] Elizabeth Gibney. "D-wave upgrade: How scientists are using the world's most controversial quantum computer". In: *Nature News* 541.7638 (2017), p. 447.

[77] David P DiVincenzo. "The physical implementation of quantum computation". In: *Fortschritte der Physik: Progress of Physics* 48.9-11 (2000), pp. 771–783.

[78] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.

[79] Lov K Grover. "Quantum mechanics helps in searching for a needle in a haystack". In: *Physical review letters* 79.2 (1997), p. 325.

[80] Arjen K Lenstra, Hendrik W Lenstra, Mark S Manasse, and John M Pollard. "The number field sieve". In: *The development of the number field sieve*. Springer, 1993, pp. 11–42.

[81] Daniel J Bernstein and Arjen K Lenstra. "A general number field sieve implementation". In: *The development of the number field sieve*. Springer, 1993, pp. 103–126.

[82] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[83] Aamir Mandviwalla, Keita Ohshiro, and Bo Ji. "Implementing Grover's Algorithm on the IBM Quantum Computers". In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, pp. 2531–2537. DOI: 10.1109/BigData.2018.8622457.

[84] Chong-Ki Hong, Zhe-Yu Ou, and Leonard Mandel. "Measurement of subpicosecond time intervals between two photons by interference". In: *Physical review letters* 59.18 (1987), p. 2044.

[85] Frédéric Bouchard, Alicia Sit, Yingwen Zhang, Robert Fickler, Filippo M Miatto, Yuan Yao, Fabio Sciarrino, and Ebrahim Karimi. "Two-photon interference: the Hong–Ou–Mandel effect". In: *Reports on Progress in Physics* 84.1 (2020), p. 012402.

[86] Chien-Hung Cho, Chih-Yu Chen, Kuo-Chin Chen, Tsung-Wei Huang, Ming-Chien Hsu, Ning-Ping Cao, Bei Zeng, Seng-Ghee Tan, and Ching-Ray Chang. "Quantum computation: Algorithms and applications". In: *Chinese Journal of Physics* 72 (2021), pp. 248–269.

[87] Bikram Khanal, Javier Orduz, Pablo Rivas, and Erich Baker. "Supercomputing leverages quantum machine learning and Grover's algorithm". In: *The Journal of Supercomputing* 79.6 (2023), pp. 6918–6940.

[88] Peter Nimbe, Benjamin Asubam Weyori, and Adebayo Felix Adekoya. "Models in quantum computing: a systematic review". In: *Quantum Information Processing* 20.2 (2021), p. 80.

[89] Fulvio Flamini, Nicolo Spagnolo, and Fabio Sciarrino. "Photonic quantum information processing: a review". In: *Reports on Progress in Physics* 82.1 (2018), p. 016001.

[90] Sergei Slussarenko and Geoff J Pryde. "Photonic quantum information processing: A concise review". In: *Applied Physics Reviews* 6.4 (2019).

[91] Jianwei Wang, Fabio Sciarrino, Anthony Laing, and Mark G Thompson. "Integrated photonic quantum technologies". In: *Nature Photonics* 14.5 (2020), pp. 273–284.

[92] Amit Kumar Sharma and Ashish Ghunawat. "A Review on Quantum Computers with Emphasize on Linear Optics Quantum Computing". In: *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*. Vol. 1. IEEE. 2019, pp. 1–3.

[93] Frank Leymann and Johanna Barzen. "The bitter truth about gate-based quantum algorithms in the NISQ era". In: *Quantum Science and Technology* 5.4 (2020), p. 044007.

[94] Marie Salm, Johanna Barzen, Frank Leymann, and Benjamin Weder. "About a criterion of successfully executing a circuit in the NISQ era: what $wd \ll 1/\epsilon_{\mathrm{eff}}$really means". In: *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*. 2020, pp. 10–13.

[95] Eberhard Zeidler. *Nonlinear functional analysis and its applications: III: variational methods and optimization*. Springer Science & Business Media, 2013.

[96] Donald R Smith. *Variational methods in optimization*. Courier Corporation, 1998.

[97] Riccardo Borghi. "The variational method in quantum mechanics: an elementary introduction". In: *European Journal of Physics* 39.3 (2018), p. 035410.

[98] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (2021), pp. 625–644.

[99] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. "Data encoding patterns for quantum computing". In: *Proceedings of the 27th Conference on Pattern Languages of Programs*. 2020, pp. 1–11.

[100] Vlatko Vedral, Adriano Barenco, and Artur Ekert. "Quantum networks for elementary arithmetic operations". In: *Physical Review A* 54.1 (1996), p. 147.

[101] John A Cortese and Timothy M Braje. "Loading classical data into a quantum computer". In: *arXiv preprint arXiv:1803.01958* (2018).

[102] Ryan LaRose and Brian Coyle. "Robust data encodings for quantum classifiers". In: *Physical Review A* 102.3 (2020), p. 032420.

[103] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical review letters* 103.15 (2009), p. 150502.

[104] Maria Schuld, Mark Fingerhuth, and Francesco Petruccione. "Implementing a distance-based classifier with a quantum interference circuit". In: *EPL (Europhysics Letters)* 119.6 (2017), p. 60002.

[105] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. "Expanding data encoding patterns for quantum algorithms". In: *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. IEEE. 2021, pp. 95–101.

[106] Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G Green, and Simone Severini. "Hierarchical quantum classifiers". In: *npj Quantum Information* 4.1 (2018), pp. 1–8.

[107] Marco Simonetti, Damiano Perri, and Osvaldo Gervasi. "An Example of Use of Variational Methods in Quantum Machine Learning". In: *Computational Science and Its Applications – ICCSA 2022 Workshops*. Ed. by Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Ana Maria A. C. Rocha, and Chiara Garau. Cham: Springer International Publishing, 2022, pp. 597–609. ISBN: 978-3-031-10592-0.

[108] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum algorithms for supervised and unsupervised machine learning". In: *arXiv preprint arXiv:1307.0411* (2013).

[109] Nathan Wiebe, Ashish Kapoor, and Krysta Svore. "Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning". In: *arXiv preprint arXiv:1401.2142* (2014).

[110] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (2014), pp. 631–633.

[111] Damiano Perri, Marco Simonetti, and Osvaldo Gervasi. "Synthetic Data Generation to Speed-Up the Object Recognition Pipeline". In: *Electronics* 11.1 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11010002. URL: https://www.mdpi.com/2079-9292/11/1/2.

[112] Seokwon Yoo, Jeongho Bang, Changhyoup Lee, and Jinhyoung Lee. "A quantum speedup in machine learning: finding an N-bit Boolean function for a classification". In: *New Journal of Physics* 16.10 (2014), p. 103014.

[113] Yingkai Ouyang. "Quantum storage in quantum ferromagnets". In: *Physical Review B* 103.14 (2021), p. 144417.

[114] Kwok Ho Wan, Oscar Dahlsten, Hlér Kristjánsson, Robert Gardner, and MS Kim. "Quantum generalisation of feedforward neural networks". In: *npj Quantum information* 3.1 (2017), pp. 1–8.

[115] Edward Farhi and Hartmut Neven. "Classification with quantum neural networks on near term processors". In: *arXiv preprint arXiv:1802.06002* (2018).

[116] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke, et al. "Noisy intermediate-scale quantum algorithms". In: *Reviews of Modern Physics* 94.1 (2022), p. 015004.

[117] Suguru Endo, Zhenyu Cai, Simon C Benjamin, and Xiao Yuan. "Hybrid quantum-classical algorithms and quantum error mitigation". In: *Journal of the Physical Society of Japan* 90.3 (2021), p. 032001.

[118] Maria Schuld and Nathan Killoran. "Quantum machine learning in feature hilbert spaces". In: *Physical review letters* 122.4 (2019), p. 040504.

[119] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

[120] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. "A quantum algorithm to train neural networks using low-depth circuits". In: *arXiv preprint arXiv:1712.05304* (2017).

[121] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. "Towards quantum machine learning with tensor networks". In: *Quantum Science and technology* 4.2 (2019), p. 024001.

[122] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. "Quantum autoencoders for efficient compression of quantum data". In: *Quantum Science and Technology* 2.4 (2017), p. 045001.

[123] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. "Circuit-centric quantum classifiers". In: *Physical Review A* 101.3 (2020), p. 032308.

[124] Dan Shepherd and Michael J Bremner. “Instantaneous quantum computation”. In: *arXiv preprint arXiv:0809.0847* (2008).

[125] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. “Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms”. In: *Advanced Quantum Technologies* 2.12 (2019), p. 1900070.

[126] Thomas Hubregtsen, Josef Pichlmeier, Patrick Stecher, and Koen Bertels. “Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability”. In: *Quantum Machine Intelligence* 3.1 (2021), pp. 1–19.

[127] Maria Schuld and Francesco Petruccione. *Machine learning with quantum computers*. Springer, 2021.

[128] Nathan Killoran, Thomas R Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. “Continuous-variable quantum neural networks”. In: *Physical Review Research* 1.3 (2019), p. 033063.

[129] Damiano Perri, Marco Simonetti, Alex Bordini, Simone Cimarelli, and Osvaldo Gervasi. “IoT to Monitor People Flow in Areas of Public Interest”. In: *Computational Science and Its Applications - ICCSA 2021 - 21st International Conference, Cagliari, Italy, September 13-16, 2021, Proceedings, Part X*. Ed. by Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Chiara Garau, Ivan Blecic, David Taniar, Bernady O. Apduhan, Ana Maria A. C. Rocha, Eufemia Tarantino, and Carmelo Maria Torre. Vol. 12958. Lecture Notes in Computer Science. Springer, 2021, pp. 658–672. DOI: 10.1007/978-3-030-87016-4_47. URL: https://doi.org/10.1007/978-3-030-87016-4%5C_47.

[130] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. “Quantum neuron: an elementary building block for machine learning on quantum computers”. In: *arXiv preprint arXiv:1711.11240* (2017).

[131] Ajit Narayanan and Tammy Menneer. “Quantum artificial neural network architectures and components”. In: *Information Sciences* 128.3-4 (2000), pp. 231–255.

[132] Stefano Mangini, Francesco Tacchino, Dario Gerace, Daniele Bajoni, and Chiara Macchiavello. “Quantum computing models for artificial neural networks”. In: *Europhysics Letters* 134.1 (2021), p. 10002.

[133] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. “The power of quantum neural networks”. In: *Nature Computational Science* 1.6 (2021), pp. 403–409.

[134] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. “Quantum machine learning: a classical perspective”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (2018), p. 20170551.

[135] Vedran Dunjko and Hans J Briegel. “Machine learning & artificial intelligence in the quantum domain: a review of recent progress”. In: *Reports on Progress in Physics* 81.7 (2018), p. 074001.

[136] Gerardo Adesso, Sammy Ragy, and Antony R Lee. “Continuous variable quantum information: Gaussian states and beyond”. In: *Open Systems & Information Dynamics* 21.01n02 (2014), p. 1440001.

[137] Alessio Serafini. *Quantum continuous variables: a primer of theoretical methods*. CRC press, 2017.

[138] Alessandro Ferraro, Stefano Olivares, and Matteo GA Paris. “Gaussian states in continuous variable quantum information”. In: *arXiv preprint quant-ph/0503237* (2005).

[139] Christian Weedbrook, Stefano Pirandola, Raúl García-Patrón, Nicolas J Cerf, Timothy C Ralph, Jeffrey H Shapiro, and Seth Lloyd. “Gaussian quantum information”. In: *Reviews of Modern Physics* 84.2 (2012), p. 621.

[140] Juan Miguel Arrazola, Thomas R Bromley, Josh Izaac, Casey R Myers, Kamil Brádler, and Nathan Killoran. “Machine learning method for state preparation and gate synthesis on photonic quantum computers”. In: *Quantum Science and Technology* 4.2 (2019), p. 024004.

[141] Hoi-Kwan Lau, Raphael Pooser, George Siopsis, and Christian Weedbrook. “Quantum machine learning over infinite dimensions”. In: *Physical review letters* 118.8 (2017), p. 080501.

[142] Karol Bartkiewicz, Clemens Gneiting, Antonín Černoch, Kateřina Jiráková, Karel Lemr, and Franco Nori. “Experimental kernel-based quantum machine learning in finite feature space”. In: *Scientific Reports* 10.1 (2020), pp. 1–9.

[143] Juan M Arrazola, Ville Bergholm, Kamil Brádler, Thomas R Bromley, Matt J Collins, Ish Dhand, Alberto Fumagalli, Thomas Gerrits, Andrey Goussev, Lukas G Helt, et al. “Quantum circuits with many photons on a programmable nanophotonic chip”. In: *Nature* 591.7848 (2021), pp. 54–60.

[144] Robert Prevedel, Philip Walther, Felix Tiefenbacher, Pascal Böhi, Rainer Kaltenbaek, Thomas Jennewein, and Anton Zeilinger. “High-speed linear optics quantum computing using active feed-forward”. In: *Nature* 445.7123 (2007), pp. 65–69.

[145] Nathan K Langford, Sven Ramelow, Robert Prevedel, William J Munro, Gerard J Milburn, and Anton Zeilinger. “Efficient quantum computing using coherent photon conversion”. In: *Nature* 478.7369 (2011), pp. 360–363.

[146] Alexey A Melnikov, Hendrik Poulsen Nautrup, Mario Krenn, Vedran Dunjko, Markus Tiersch, Anton Zeilinger, and Hans J Briegel. “Active learning machine learns to create new quantum experiments”. In: *Proceedings of the National Academy of Sciences* 115.6 (2018), pp. 1221–1226.

[147] Andrea Rocchetto, Scott Aaronson, Simone Severini, Gonzalo Carvacho, Davide Poderini, Iris Agresti, Marco Bentivegna, and Fabio Sciarrino. “Experimental learning of quantum states”. In: *Science advances* 5.3 (2019), eaau1946.

[148] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. “Pennylane: Automatic differentiation of hybrid quantum-classical computations”. In: *arXiv preprint arXiv:1811.04968* (2018).

[149] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. “Strawberry fields: A software platform for photonic quantum computing”. In: *Quantum* 3 (2019), p. 129.

[150] Maria Schuld and Nathan Killoran. “Is Quantum Advantage the Right Goal for Quantum Machine Learning?” In: *PRX Quantum* 3 (3 July 2022), p. 030101. DOI: 10.1103/PRXQuantum.3.030101. URL: https://link.aps.org/doi/10.1103/PRXQuantum.3.030101.