POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Quantum Machine Learning Fault Injection

Supervisors

Prof. Bartolomeo MONTRUCCHIO

Res. Assistant Edoardo GIUSTO

External Prof. Paolo RECH

Candidate

Marzio VALLERO

JULY 2022

Summary

This work stems from the composition of three separate areas of research under the goal of deepening the understanding of their interaction. These fields are namely *Quantum Computing*, *Machine Learning* and *Fault Injection and Reliability testing.*

Quantum Computing is still an emerging technology in constant evolution, day by day. The true extent of the advancements it will bring to humanity as a whole are numerous and possibly unpredictable. Despite being historically relegated to theoretical physicists and mathematicians, this field has seen a recent widespread surge in research interest among engineers and computer scientists alike thanks to the development of the first publicly accessible quantum devices over the cloud in 2016. The first chapter of this *Thesis* is devoted to the introduction of the basic knowledge needed to get a grasp of understanding of the subject, alongside the main quantum circuit subroutines and the current status of research and development in the field.

Fault Injection and Reliability studies laid the foundations for many of humankind's most prized recent achievements, namely deep space exploration and transistor gate miniaturization. It consists of both systematical and statistical analysis methods aimed at stressing electronic circuitry with the purpose of spotting out fault behaviours and patterns, eventually leading to the development of devices able to resist, detect and possibly correct such anomalies.
The second chapter glimpses over the classical fault model of modern transistor-based devices and provides a definition of the structure of cosmic rays, alongside relative literature, eventually introducing one of the current models for quantum faults induced by external radiation on transmon-based devices and the issues associated with it. A metric for reliability measurement called Quantum Vulnerability Factor is introduced in depth. After that, a purposefully developed software simulation suite is presented, which will be later used to carry out injections, execute them in parallel on a number of different simulators or hardware backends and perform QVF based reliability analyses on quantum circuits.

Machine Learning is comprised of all the techniques and attempts that aim at infusing classical computation devices with what could be defined as *intelligence*. Given the large plethora of interpretations that can be given to such term, the branches of Machine Learning are likewise numerous and intertwined. Starting from the novelties provided by both Quantum Computing and Machine Learning, it has been tried to merge the two into the broader field of *Quantum Machine Learning*, in order to define a possibly more complete and general paradigm.
The third chapter of this work intends to provide the most superficial technicalities required to understand the main concepts and definitions of these models, starting from the classical computation domain and moving towards its quantum dual, with the additional introduction of the main branches of QML, the general methods for data encoding and gradient computation.

Reached this point, the true heart of this *Thesis* is unveiled in the fourth chapter, where the three fields are merged together in the fault reliability analysis of two modern quantum machine learning models, a Quantum Support Vector Machine and a Quantum Convolutional Neural Network. The aim is to dissect the interaction of these architectures with the specific radiation-induced quantum fault model discussed in the second chapter, in order to provide a baseline for further research in the years to come.
The Quantum Support Vector Machine is one of the current strongest contenders in the field of Quantum Machine Learning, since it has been proven to boast a clear quantum advantage over classical methods when dealing with specific features. The two qubit quantum circuit responsible for performing the inner product between the elements in the dataset takes advantage of the mathematical properties of the so called *kernel trick*, mapping the inputs into a higher dimensional space where the features become linearly separable. Following that, a QVF study allowed for the identification of the most critical faults in the circuit, demonstrating that, despite marginal differences, both qubits in the QSVM's circuit, called second-order Pauli-Z evolution circuit, have a similar level of reliability to radiation induced faults. The highest criticality comes from a subset of the possible phase shifts of the statevector, whilst the remainder of the fault syndromes add a periodic variation on the final output. Furthermore, an analysis of the impact of a double fault has been investigated and compared to the one of the single fault.
After that, a more thorough analysis has been conducted onto a simplified model of a Quantum Convolutional Neural Network. This time, the quantum circuit featured four qubits, thus allowing for more complex entanglement states. After glossing over the base performance capabilities of this architecture, a first QVF study has been conducted, which gave a description of the circuit's reliability pattern. Given the amplitude embedding strategy used for the inputs, it has been proven that an azimuthal angle based fault is devastating on the network's output, a behaviour that

highly different with respect to a polar angle based fault. As such, to support this hypothesis also from the QNN's accuracy side, a subset of fault positions has been studied more thoroughly, highlighting a response pattern which proved the previous assumption. From this point on, a final, deeper analysis has been carried out at the dataset level, in order to spot out the input images which either maximised or minimized the network's resilience to faults. For each proposed fault position, the two extreme cases have been presented and analyzed. Despite not being a thorough top down approach, it allowed to prove that there is a different resilience pattern depending on which qubit is affected by the fault, while at the same time showing that the depth of the fault in the circuit has a negligible impact on the final output. This study provided sufficient information to get a basic understanding of the circuit's characteristics, thus paving the way for further research in the future.

At last, in the fifth chapter, a final digression on the *Thesis* alongside suggestions for future research are appended, concluding the work.

Acknowledgements

"Guarda, i vigili dei pompieri!"
-

I thank my parents, *Silvana* and *Fabio*, for having taught me to always pursue my dreams and for the many sacrifices they have made to let me achieve them. I know that sometimes I seem detached from reality, that my mood shifts like the weather and that I give you two too much to worry about, but I really could have not done it without your omnipresent support, which persists despite all of this.
I thank my uncle *Marco* for all the time he has spent with me talking about technology, science and work perspectives in the future, and for presenting a point of view that has always been an inspiration for me, letting me practice in proving my ideas to others and to be convincing in doing so.
I thank my grandfather *Aldo* for having shown me that it's never too late or too difficult to learn something new, alongside being the prime example of the fact that one should always pursue curiosity and knowledge, whenever possible.
I thank my grandmother *Carla* for all the times she looked after me, for her endless servings of homemade pasta, agnolotti and fritto misto, and for her patience.
I wish she could have seen this day.
I thank my grandmother *Liliana* for gifting me with an artistic eye for things, for all the interest she's shown in my academic career and for letting me explain to her what I was doing in simple words, helping me in understanding that too.

I thank *Alessandro* for his constant stream of Star Wars prequels' memes, for sharing a passion for drawing despite our lack of time for it and for always being ready to crack a joke, cheering me up whenever I did not want to.
I thank *Giampietro* for the time we spent together playing games, both tabletop and digital, for the numerous delicious treats from Sicily I got the chance to taste

with him, for giving me the most rational answer to the question "What should I do next?".
I thank *Simone* for having been the most sincere teammate during these last years, especially in the tedious and tense times alongside exams and project deadlines, persisting even after our academic paths drifted away from one another.
I also thank these three guys for all the time we've spent together during lectures, never ending studying sessions and lunches (mainly at John's).

I thank *Alberto* for all the car rides spanning thousands of kilometers, for his characteristic rage at insignificant things that always makes me smile, for all the times he asked if I was alright late in the night, even if I did not feel like talking, for all the things he's done without needing an explanation.
I thank *Alex* for that one long walk around the block in the cold of December and for having been there despite not needing to, for helping me be more decisive and direct in my choices and for always making me laugh. And lots of box openings.
I thank *Riccardo* for the moments we shared together during, well, our whole life, for all the strange favours he asks me from time to time, be it a photocopy or a pair of crutches, for all the talks revolving around stuff we would both rather not have to deal with.
I thank *Valerio* for the frightening synchronicity we share from time to time, for always challenging my expectations, ideas and music taste, for his marvelous D&D settings, which gave me the chance to be elsewhere even if just for a fleeting moment (literally since I fall asleep half of the time).

I thank *Martina* for the complete trust she gave me as soon as we met, as if we had always known each other, for having been close to me over the last years, for all the songs we sang together and for all the movies we are yet to watch.
I am glad you stayed.

Table of Contents

List of Tables

List of Figures

List of Equations

Acronyms

AI

Artificial Intelligence

AVF

Architectural Vulnerability Factors

BQP

Bounded-error quantum polynomial time

CMOS

Complementary metal–oxide–semiconductor

CNN

Convolutional Neural Network

CNOT

Controlled NOT

CTD

Curch-Turing-Deutsch

ECC

Error Correction Code

EDC

Error Detection Code

EPR

Einstein-Podolsky-Rosen

FRQI

Flexible Representation of Quantum Images

IQFT

Inverse Quantum Fourier Transform

KDE

Kernel Density Estimate

MLP

Multi Layer Perceptron

ML

Machine Learning

MNIST

Modified National Institute of Standards and Technology

MSE

Mean Squared Error

NEQR

Novel Enhanced Quantum Representation of digital images

NISQ

Noisy Intermediate Scale Quantum

NMR

Nuclear Magnetic Resonance

PST

Probability of Successful Trials

PVF

Program Vulnerability Factors

QC

Quantum Computing

QEC	Quantum Error Correction
QML	Quantum Machine Learning
QNN	Quanvolutional Neural Network
QPE	Quantum Phase Estimation
QSVM	Quantum Support Vector Machine
QVF	Quantum Vulnerability Factor
QuFI	Quantum Fault Injector
Qubit	Quantum binary digit
RAMs	Random Access Memories
RSA	Rivest–Shamir–Adleman
ReLU	Rectified Linear Unit
SDK	Software Development Kit
SEU	Single Event Upset
SVM	Support Vector Machine

VQC	Variational Quantum Classifier
VQE	Variational Quantum Eigensolver
XOR	Exclusive OR

Chapter 1

Introduction

1.1 Brief history of quantum computation

Since their original theoretical inception proposed by Benioff [1, 2] and Feynman [3] during the early 1980s, quantum mechanical computation models have peaked the interest of the scientific community as a mean to exceed the limitations of classical computation in tackling exponentially complex problems [4].
In the following decade, Shor [5] and Grover [6] proposed their revolutionary self-named algorithms, which became the main vectors of public interest and research funding towards the quantum computing paradigm. They, among others, demonstrated that algorithmic speedups were possible and paved the way for the following research. However, at the time the technology was still in its infancy and these assertions had yet to be tested experimentally, as no physical devices had been built up to then. Notably, in 1998, Chuang *et al.*[7], following their implementation of an NMR based quantum device, executed experimentally for the first time Grover's algorithm.
At the verge of the new century, the first edition of the de-facto standard textbook on quantum information theory, *Quantum Computation and Quantum Information* by Nielsen and Young [8] has been published, acting as an entry gate to the topic for a broader range of students [9, 10] and being positively suggested by the most renowned names in the field, among which Shor, Grover, Aaronson and DiVincenzo. Following there on, multiple implementation architectures have been proposed and built, among which the first entanglement capable photonic device by Jian-Wei *et al.*[11], quantum dot based electron spin manipulators by Vandersypen *et al.*[12] and superconducting circuits by Plantenberg *et al.*[13].
The last ten years provided the biggest leaps in research and development, especially due to the participation of big tech giants of the industry. D-Wave developed in 2011 the first commercially available quantum annealer, the *D-Wave One.* Multiple

research teams, among which Maurer *et al.*[14], Saeedi *et al.*[15] and Zhong *et al.*[16] have improved decoherence times of certain quantum states from mere nanoseconds to possibly hours. The experimental demonstration of the quantum teleportation algorithm by Pfaff *et al.*[17] with zero error rate in 2014 laid the foundation for quantum based reliable communication. In 2016, IBM revolutionized the concept of quantum computation by carrying over the paradigm of cloud based classical computation onto their transmon based superconducting quantum devices with the launch of *Quantum Experience*, widening accessibility and giving rise to a plethora of new research among which the first were Devitt *et al.*[18] and Alsina *et al.*[19]. In the last three years especially, development of higher qubit count quantum devices increased dramatically, as a mean for achieving quantum computational supremacy over other competitors, with numerous breakthroughs from Google, IBM, D-Wave, Intel, Rigetti Systems and Xanadu just to name a few.
This sprouting interest among the scientific community and investors alike revolves around only a small fraction of the computational capabilities of the quantum paradigm. As of now we are in the NISQ era (Noisy Intermediate Scale Quantum) [20], so there are still many algorithms to be discovered and challenges to be overcome, namely noise, decoherence times and qubit counts.

1.2 Quantum information theory

Classical computation's basic information unit is the *bit*, a deterministic logical object with the ability to assume exclusively one of two binary values, commonly referred to in boolean logic as either a 0 or a 1. It has been mathematically proven for such object to be the smallest possible unit of information by Abramson and Hartley [21], and later revised by Shannon.
Quantum computation's basic information unit is the so called quantum bit, or *qubit* for short, formerly named as such by Schumacher [22] as the quantum dual to the classical bit of information. Unlike its counterpart however, the qubit exhibits properties that let it overcome the bounds of binary logic.
One major characteristic of the qubit is it's ability to carry more information with respect to the classical bit, while still only granting access to the same amount of information, following the mathematical proof of Holevo's [23] theorem of 1973. This required a generalization of Shannon's information theory, first attempted in the seminal work of Ingarden of 1975 [24] which laid the foundations of modern quantum information theory.
The main characteristics of the qubit derive from the fundamental characteristics of quantum mechanics, which can be summed up into:

- *Superposition*: qubits can be in an intermediate state which is probabilistically both 0 and 1 at the same time.

- *Entanglement*: two qubits can be coupled into a joint quantum state which exhibits manipulation causality between one qubit and the other. The two qubits thus cannot be described indipendently of one another.
- *Interference*: a qubit is able to cross and interfere with its own state.

At the foundation of quantum mechanics lies the description of the time evolution of quantum systems, defined by the *Schrödinger equation*

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle \tag{1.1}$$

where $\hbar$ is *Planck's constant* and H is the *Hamiltonian* matrix representing the quantum system's total energy. Such equation can be solved as an initial value problem for time $t = 0$, defining the state $|\psi_0\rangle = |\psi(t = 0)\rangle$, allowing for the definition of the unitary time evolution operator $U(t)$ [25] as

$$U(t) = e^{-i\frac{t}{\hbar}H} \tag{1.2}$$

By exploiting such properties, a new paradigm of computation has been created, commonly referred to as *quantum computation*.

1.3 Computational basics

The following section will go over the main concepts related to quantum computing and the relative differences with respect to its classical counterpart.

1.3.1 Quantum bit

As previously introduced, quantum bits, or *qubits*, are the basic information units of quantum computational models. They are formally described by a two-dimensional vector. We can define two basis states which act as a parallel to the classical bits' 0 and 1 states, called $|0\rangle$ (*ket-zero*) and $|1\rangle$ (*ket-one*)

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{1.3}$$

following Dirac's vector notation, also called *bra-ket* notation [26].
These two states form an orthonormal basis for the two-dimensional vector space, formally called *Z basis*. However, it's important to note that such vector space

allows for an infinite number of orthonormal bases: the Z basis simply stands out as the easiest to model and to introduce quantum computation with.
These two objects can be represented graphically as two antipodal points lying onto the surface of a unitary radius sphere, called *Bloch sphere* [27][28]:

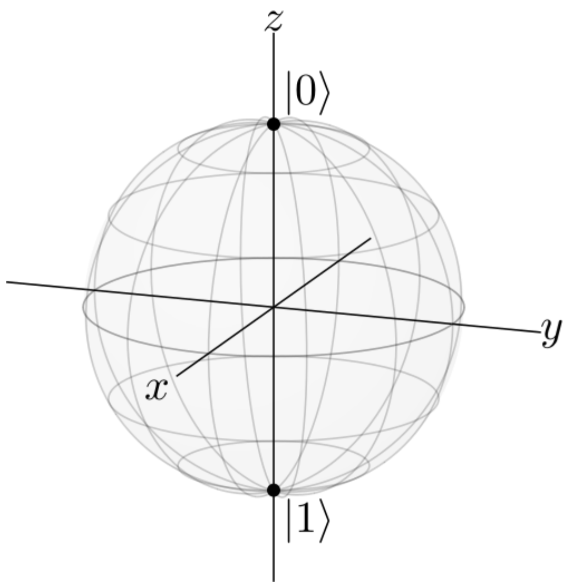


Figure 1.1: $|0\rangle$ and $|1\rangle$ on the Bloch sphere.

1.3.2 Superposition

While classical bits are forced to exist in a deterministic state, assuming either the value of 0 or 1, qubits can exist in a *superposition* state of the basis states. As such a qubit can be represented by a linear combination of basis states in the vector space as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.4}$$

where α, $\beta \in \mathbb{C}$ are the probability amplitudes of the qubit lying in either the $|0\rangle$ or $|1\rangle$ state. It follows that these amplitudes must obey the basic rule of sum from probability theory

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1.5}$$

with $|\alpha|^2$ and $|\beta|^2$ representing the probability that a measurement operation in the Z basis collapses the state of the qubit in either the $|0\rangle$ or $|1\rangle$ state. The qubit thus exists in a *continuum* of states before observation and collapses to a single

state once measured.
As an example, we may consider the superposition state

$$|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{1.6}$$

where through the distributive property of multiplication is easy to see that

$$|\alpha|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2} \quad \text{and} \quad |\beta|^2 = \frac{1}{2} \tag{1.7}$$

From this it intuitively follows that this state will collapse through measurement on state $|0\rangle$ half of the times, and on $|1\rangle$ on the other half. It is clear to see this probabilistic distribution by visualizing the $|+\rangle$ state on the Bloch sphere, as the point stands exactly on the equator (the x-y plane):

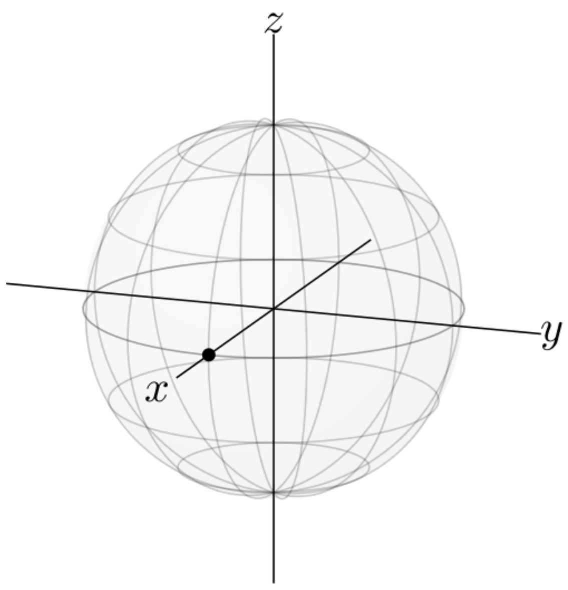


Figure 1.2: $|+\rangle$ on the Bloch sphere.

Moreover, we can define a $|-\rangle$ state which is antipodal to $|+\rangle$: together these two states represent another possible orthonormal basis for the vector space, the *X basis*, as it lies on the X axis. Similarly, we can define another axis oriented basis, the *Y basis*, composed of the states $|i\rangle$ and $|-i\rangle$. However, a qubit's state is not limited to the axes of the Bloch sphere; in fact, it is not limited to the equator on the x-y plane either.
A quantum state can lie in any point of the surface of the Bloch sphere.

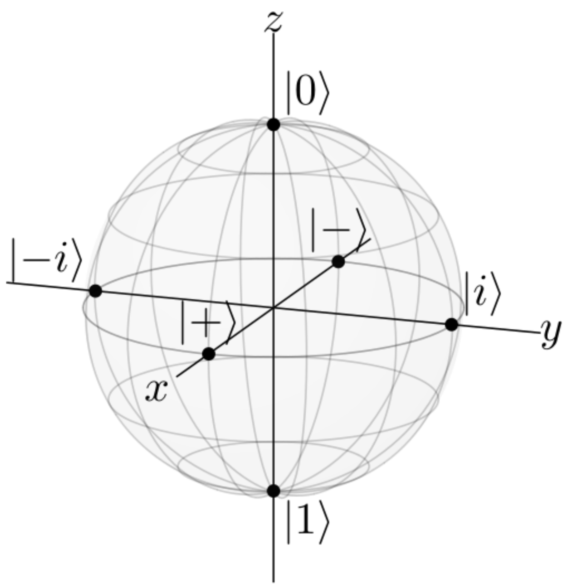


Figure 1.3: The standard basis states for the X, Y and Z bases on the Bloch sphere.

In fact, it's possible to represent a point on the surface of the Bloch sphere by means of a change of coordinates, mapping between the triplet (x, y, z) and the triplet (ρ, θ, ϕ), representing respectively the radial distance, the polar angle and the azimuthal angle. This mapping lets us reparametrize the α and β probability amplitudes in terms of θ and ϕ

$$\alpha = \cos\left(\frac{\theta}{2}\right), \quad \beta = e^{i\phi}\sin\left(\frac{\theta}{2}\right) \tag{1.8}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. It's clear to see how α is real valued, whilst β is complex and the sum squared of the two terms is still normalized. The reparametrized generic quantum state can be written as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{1.9}$$

Notably, no term depends on the radial distance ρ, as all the quantum states represented up to now lie on the surface of the sphere, so have all $\rho = 1$: such states are defined *pure sates*. On the other hand, points internal to the sphere's volume represent *mixed states* [8].
The role of the angles θ and ϕ can be easily visualized onto the Bloch sphere as the angle between the $|0\rangle$ state on the positive Z axis and the qubit and the angle between the $|+\rangle$ state on the positive X axis and the qubit respectively.

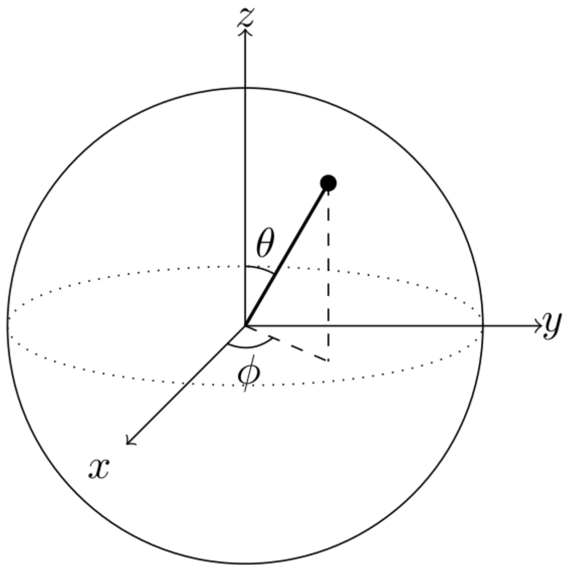


Figure 1.4: Spherical coordinates visualization.

1.3.3 Measurements

Even though the properties of quantum mechanics let the qubit exist in a superposition state, they also require for it to output a single value when measured, either 0 or 1 according to the amplitudes α and β.
The probabilistic nature of the qubit is directly linked to the *observer effect* [29], which states that the act of observing a quantum system permanently alters its state, forcing the wave function representing it to collapse in either of the states composing the measurement basis [30]. As such, it's not possible to retrieve completely the information embedded into a quantum system through a single measurement. Susbsequent measurements of the same qubit on the same basis in fact will always yield the same output as the first one with 100% probability.
This *unobservability* property is notoriously unintuitive as it doesn't translate well into the familiar cause-effect world that we are accustomed to experience in our everyday lives.
However, despite the uncertainty principle linked to quantum measurements, it's still possible to trace down the effect of manipulations to the state of a quantum system onto the measurement output.

Given an unknown quantum state $|\psi\rangle$ and a measurement operator M_m and its conjugate transpose $M_m^\dagger$, we can compute the probability of obtaining the output m through the probability function $P : \mathcal{F} \to [0,1]$ as

$$P(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{1.10}$$

where $\mathcal{F}$ represents the collection of possible outcome events. After the measurement, the state of the system will collapse according to the *Born rule* [31], conditioned on having effectively measured m, to the form

$$|\psi\rangle \to \frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}} \tag{1.11}$$

This kind of operators can be seen as projectors onto an orthonormal basis, as such they obey the property of unitarity $P^2 = P$, i.e. the inner product of the matrix representing the measurement operator with itself corresponds to the Identity matrix.
As an example, we can derive the matrix form of the measurement operators M_0 and M_1 for the Z basis as the outer product of the basis states with themselves as

$$M_0 = |0\rangle\langle 0| \ , \quad M_1 = |1\rangle\langle 1| \tag{1.12}$$

From this follows that the probability of measuring the quantum system to be in state $|1\rangle$ is

$$\begin{aligned} \langle\psi| M_1^\dagger M_1 |\psi\rangle &= \langle\psi|1\rangle\langle 1|\psi\rangle \\ &= (\alpha^* \langle 0| + \beta^* |1\rangle)(|1\rangle\langle 1|)(\alpha |0\rangle + \beta |1\rangle) \\ &= |\beta|^2 \end{aligned} \tag{1.13}$$

as expected. By substituting the newfound probability in the measurement collapse relation conditioned on having measured $|1\rangle$, we obtain that

$$|\psi\rangle \to \frac{M_1 |\psi\rangle}{\sqrt{\langle\psi| M_1^\dagger M_1 |\psi\rangle}} = \frac{1}{|\beta|} |1\rangle\langle 1| (\alpha |0\rangle + \beta |1\rangle) = \frac{\beta}{|\beta|} |1\rangle \approx |1\rangle \tag{1.14}$$

1.3.4 Multiple qubits

Quantum computational systems may be composed of any number of qubits. Due to the laws of quantum mechanics, such systems may be be described as a set of disjoint elements, while at other times these elements may be intertwined, by a process called *entanglement*.
Adding qubits to a quantum system enlarges its vector space. As such, the Bloch sphere cannot represent multi-qubit systems: in order to tackle the dimensionality

increase in the vector space, we must increase the number of computational basis states by an exponential factor with respect to the number of qubits in the system N, following the relation N qubits $\rightarrow 2^N$ bases.
For example, let's consider two qubits $|\psi_0\rangle = \alpha_0 |0\rangle + \beta_0 |0\rangle$ and $|\psi_1\rangle = \alpha_1 |0\rangle + \beta_1 |0\rangle$. The two qubit state $|\psi\rangle = |\psi_1\rangle \otimes |\psi_0\rangle$, i.e. with N = 2, is generally represented as

$$|\psi\rangle = \alpha_0\alpha_1 |00\rangle + \alpha_0\beta_1 |01\rangle + \beta_0\alpha_1 |10\rangle + \beta_0\beta_1 |11\rangle \tag{1.15}$$

where each linear combination of single qubit amplitudes represents the probability amplitude of observing its relative basis state and can be rewritten as a coefficient γ_y. Once again, the rule of sum from probability theory requires that these amplitudes are normalized, according to the condition

$$\sum_{y \in \{0,1\}^N} |\gamma_y|^2 = 1 \tag{1.16}$$

Product states, also called *simply separable states*, are quantum systems composed of single linearly separable qubits [32]. This means that each γ_y coefficient can be rewritten in terms of single qubit amplitudes α_y and β_y.

1.3.5 Entanglement

The separability property cannot be always satisfied. In 1935, Einstein, Podolsky and Rosen [33] outlined the existence of non linearly separable two *particle* (i.e. qubit) quantum states, called *entangled.* In fact, such states gave rise to the *EPR paradox*, stating that the observation and consequent collapse of one of the two *particles* would determine with certainty the state of the other without needing to act onto it, effectively forcing it to collapse too. This would allow for the apparent instantaneous transmission of information from one *particle* to the other, possibly violating one of the fundamental constraints of the theory of relativity [34]: information cannot travel at a speed faster than light. The paradox has later been solved by the postulation of the *no-communication theorem*[32]: albeit the collapse of the two qubits is instantaneous, there is no way for such action to carry information by itself, thus it's bound by classical communication limits.
These entangled states are called EPR pairs or Bell states [8], one example of which is the following

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.17}$$

By measuring one of the two qubits of $|\Phi^+\rangle$, we will know with 100% probability that the other will have collapsed to the same basis state.
It's clear how the pair $|\Phi^+\rangle$ cannot be written as the linear product of two separate

qubits $|\psi_1\rangle\, |\psi_0\rangle$. Recalling equation (1.15) and matching the coefficients, we would get this system of equations

$$\alpha_0\alpha_1 = \frac{1}{\sqrt{2}}\ , \quad \alpha_0\beta_1 = 0\ , \quad \beta_0\alpha_1 = 0\ , \quad \beta_0\beta_1 = \frac{1}{\sqrt{2}} \tag{1.18}$$

where it is easy to demonstrate that no solution exist. In fact, the two middle equations require for at least one term to be equal to 0, whilst the first and the last equation require for both to be non-zero.
Relations in (1.18) are directly linked to the proof of the *no-cloning* theorem, which states that it is impossible to copy any unknown quantum state without first performing some measurement, ultimately destroying the original state. It's important to note however how basis states $|0\rangle$ and $|1\rangle$ can be copied with a simple $CNOT$ gate, albeit such application is limited.

1.4 Quantum Logic

The following section will define the main formalisms behind quantum logic gates, drawing parallels with classical computation.

1.4.1 Single qubit gates

In order to perform transformations onto qubits, we must employ quantum gates, equivalently to logic gates in classical computing.
Quantum gates are represented by square matrices with size dependent on the number of qubits they act onto and must obey some fundamental properties, among which:

- *Linearity*: a quantum gate must be distributable among a superposition state.
- *Normalization*: a quantum gate must keep the total sum of probabilities (1.16) equal to 1.
- *Reversibility*: for any quantum gate U there must exist a reverse gate $U^\dagger$ which restores the original state, such that $UU^\dagger = I$.

Each single qubit gate is represented by a 2×2 matrix and can be interpreted as a rotation around a specific axis of the Bloch sphere.
The identity gate is the most simple gate, as it acts onto the qubit without altering its state, and it's represented by the I matrix

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.19}$$

Rotations about the main X, Y and Z axes by an angle π are called Pauli gates and assume the matrix forms

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \; , \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \; , \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1.20}$$

where the *Pauli-X* gate is commonly referred to as the *NOT* gate.
The most used single qubit gate is the Hadamard gate, also called *squared-root not*. It corresponds to a rotation by an angle π about the $x + z$ axis. It maps the $|0\rangle$ and the $|1\rangle$ states to the $|+\rangle$ and $|-\rangle$ superposition states, effectively performing a basis change.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.21}$$

It's important to note that all the quantum gates introduced up to now are

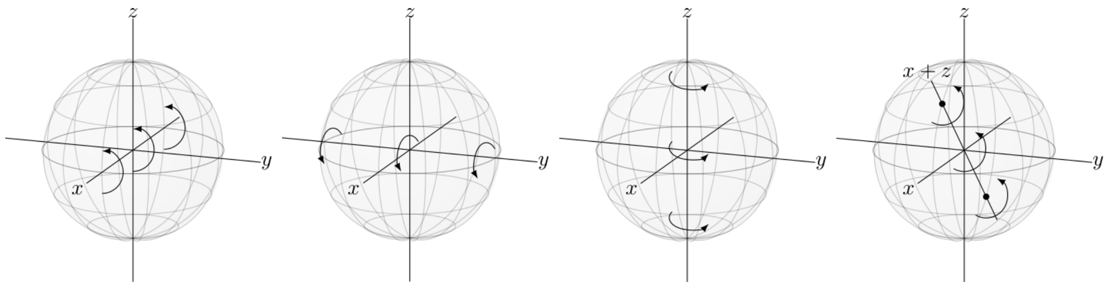


Figure 1.5: Visualization of the basic X, Y, Z and H gates' rotations.

represented by *Hermitian* matrices, so they are equal to their complex conjugate transpose:

$$XX^\dagger = XX = X^2 = I \; , \quad Y^2 = I \; , \quad Z^2 = I \; , \quad H^2 = I \tag{1.22}$$

From this follows that any even number of consecutive applications of one such matrix amounts to applying an Identity gate.
The S and T gates are specialized rotations about the Z axis by an angle of $\frac{\pi}{2}$ and $\frac{\pi}{4}$ respectively. As such, $T^2 = S$ and $S^2 = Z$. However they are not Hermitian, since $S \neq S^\dagger$ and $T \neq T^\dagger$:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix} \ , \quad S^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{i\pi}{2}} \end{bmatrix} \ , \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \ , \quad T^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{i\pi}{4}} \end{bmatrix} \tag{1.23}$$

1.4.2 Multiple qubit gates

The simplest multiple qubit gate is the *SWAP* gate, which exchanges position between two qubits. This amounts to no visible change in the probability amplitudes of the $|00\rangle$ and $|11\rangle$ bases, whilst swapping the ones of the $|01\rangle$ and $|10\rangle$ bases.

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \rightarrow \quad \begin{aligned} SWAP\,|00\rangle &= |00\rangle \\ SWAP\,|01\rangle &= |10\rangle \\ SWAP\,|10\rangle &= |01\rangle \\ SWAP\,|11\rangle &= |11\rangle \end{aligned} \tag{1.24}$$

Multiple qubit gates usually involve one or more control qubits and a target qubit. Given a generic one qubit unitary gate U, we can define the generic two qubit *controlled-U* gate as

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \ , \quad CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \quad \rightarrow \quad \begin{aligned} CU\,|00\rangle &= |00\rangle \\ CU\,|01\rangle &= |01\rangle \\ CU\,|10\rangle &= |1\rangle \otimes U\,|0\rangle \\ CU\,|11\rangle &= |1\rangle \otimes U\,|1\rangle \end{aligned} \tag{1.25}$$

One example is the *controlled-NOT* gate or *CNOT* for short. It inverts the target qubit by applying a *Pauli-X* gate conditioned on the control qubit being 1.

$$CNOT\,|a\rangle\,|b\rangle = |a\rangle\,|a \oplus b\rangle \tag{1.26}$$

As such, the *CNOT* is the quantum dual of the classical *XOR* gate. Its matrix form can be derived by substituting the *Pauli-X* matrix to the U gate in (1.25).

Multiple qubit gates are not limited to a single control and target qubit.
For example, the 3 qubit Toffoli gate, also called *controlled-controlled-NOT*, applies a *Pauli-X* gate to the third qubit if the first two are *both* 1. Since it acts onto 3 qubits, it is represented by an 8×8 square matrix, as there are $2^N = 2^3 = 8$ basis states.

$$\text{Toffoli}\, |a\rangle\, |b\rangle\, |c\rangle = |a\rangle\, |b\rangle\, |ab \oplus c\rangle \tag{1.27}$$

The Toffoli gate performs the quantum equivalent of the *AND* operator between two qubits and stores its result in the third qubit, as per required to be a reversible gate.
With the Toffoli gate, which is universal for classical computation, we can state that there exists also a universal gate set for quantum computation [8].

It can be shown that a quantum computer can perform any operation possible on a classical computer with at most a polynomial overhead, as stated by the *Church-Turing-Deutsch thesis*[32, 4].

1.4.3 DiVincenzo's criteria

There are endless possibilities for the implementation of quantum gates, however it would not make sense to implement physically all of them in order to build a working quantum computer. In the year 2000, DiVincenzo [35] proposed a set of self named criteria which defines the minimum amount of properties necessary for quantum computation: any device implementing all of them, would thus be able to perform any quantum algorithm. There are seven criteria, of which the first five regard how to create a universal set of gates, whilst the last two are relative to quantum communication.

Scalability with well-characterised qubits

To perform computations and possibly gain an advantage over classical computation, it must be possible to easily produce quantum devices with an arbitrarily high number of qubits, by adding at most a constant overhead.

Initialization of qubits to a simple fiducial state

All operations performed on quantum devices are unitary, as such the result of the computation heavily depends on the starting state. It must be possible to efficiently initialize any number of qubits in a system with a time overhead which is fast enough to allow possibly mid circuit re-initialization, to be used especially in error correction routines. Moreover, the initialization time must be lower than the decoherence time, else no computation can be possible.

Long relevant decoherence times

Current quantum devices are characterized by *T1* and *T2* relaxation times. In order to allow for the execution of deep circuits with high fidelity, the gate operation times and the decoherence factor they introduce must be many orders of magnitude smaller than the qubit's relaxation time. This can be improved both by producing better gates and improving *T1* and *T2* times.

Universal set of quantum gates

There exists a minimum number of one and two qubit gates which allow for the execution of a specific algorithm. If a set allows for the execution of any possible algorithm, like in the classical case, it is said universal. There is no specific gate set which is better than the others, but a device must support at least one of them, else it cannot effectively manipulate the quantum states to their full extent.

Qubit-specific measurement capability

It must be possible to measure the state of a qubit at the end of a quantum algorithm in order to access the results of the computation with a high enough efficiency. Given a lower efficiency, the algorithm needs to be repeated to improve the success rate.

Interconverting stationary and flying qubits

It must be possible to transfer a qubit from an encoding technology to another, while preserving the quantum information, and vice versa.

Faithful transmission of flying qubits between specified locations

When transmitting a qubit, it must be possible to do so without incurring in decoherence, preserving the quantum state from source to destination.

1.5 Quantum circuits

In order to perform computations with the gates just presented, it is necessary to define a model that regulates the interactions of these objects over time and among multiple qubits. A circuit's graphical representation, firstly introduced by Feynman [3] as an adaptation of the *Penrose notation*, is implicitly read from left to right as a temporal sequence of operations. Each qubit is represented by an horizontal line, whilst classical bits are represented by two parallel horizontal lines, which however do not represent a physical connection among devices, but rather may be read as a sequence of events on a control flow graph. Quantum gates are placed onto one or more wires depending on their number of inputs and are executed in a parallel fashion among all qubits.
An example for the qubit teleportation circuit is given in Figure 1.6, where the

unknown state of q_0 is transferred to q_2. CNOT gates are represented as a filled dot on the control qubit and as a circle with a plus sign on the target qubit. The measurement gate with respect to the *Z basis* is represented by an arrow over a meter. Operations can also be conditioned on the result of an observation, as per the Z and X gates towards the end of the circuit.

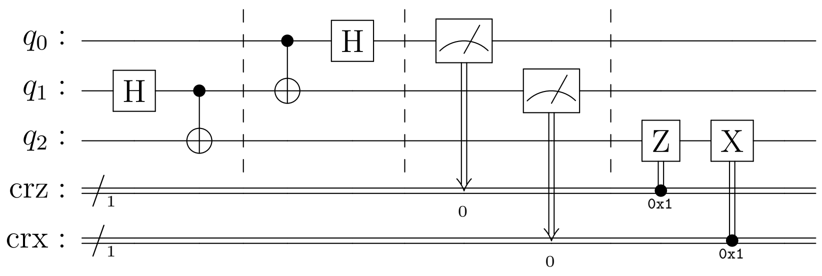


Figure 1.6: Quantum teleportation circuit.

1.5.1 Shor's algorithm

Large number factorization is one of mathematics's most computationally expensive problems to solve on classical computers. The algorithm, called *number field sieve*, in fact, given a n bits number, scales in time as $e^{n^{1/3}}$, which is a growth faster than polynomial, thus rendering the factoring inefficient for classical computers. This is the reason as for why the foundation of modern asymmetric cryptography, the RSA algorithm, is based on that same mathematical problem, taking advantage of the fact that a brute force approach would require computation times possibly longer than the current estimates for the age of the universe.
In 1994 Peter Shor [5] proposed a quantum algorithm able to solve the factorization problem in polynomial time, which is an optimal scaling factor, despite not achieving an exponential speedup. The algorithm is made of classical steps and quantum ones, which will be detailed in the following list. Given a number $N = pq$, where p and q are two prime numbers, it is possible to factor out the primes from N as

1. Choose a number $1 < a < N$, then compute the $gcd(a, N)$. If $\gcd(a, N) \neq 1$, then $p = \gcd(a, N)$, and $q = \frac{N}{p}$. Else, continue.

2. Compute the period r of $a^x \mod N$ through the use of the period finding quantum algorithm. If r is even and $a^{r/2} \mod N \neq N - 1$, continue, else restart from point 1 with a new a.

3. Given the period r, it will hold that $a^r = 1 \mod N \rightarrow a^r - 1 = 0 \mod N = kN = kpq$, since $a^r - 1$ is a multiple of N. Finally, by refactoring the first half of the equation $(a^{r/2} - 1)(a^{r/2} + 1) = kpq$.

4. Since step 2 grants that r is even, then the two terms in parentheses must be even as well. Given two integers c and d such that $cd = k$, it is possible to rewrite the equation as $(a^{r/2} - 1)(a^{r/2} + 1) = (cp)(dq) = kpq$ as the only possible combination of c, d, p and q. Thus, it is possible to extract p and q as

$$p = \gcd(a^{r/2} - 1, N) \quad , \quad q = \gcd(a^{r/2} + 1, N) \tag{1.28}$$

The second step of the algorithm is responsible for computing the period of the function $f(x) = a^x \mod N$, such that r is the smallest number for which it holds that $a^r \mod N = 1$.
Given a modular multiplication unitary operator U and a factoring target N written on n bits, a number y on n bits is chosen.

$$U|y\rangle = |ay \mod N\rangle \tag{1.29}$$

Multiple applications of the U operator produce different powers of a in output, which is simply the exponentiation of a. Since the modulus gives rise to periodic functions, after r applications of the U operator, the outputs repeat themselves. Considering an eigenstate made of the superposition of all the exponentiations of a, each multiplied by a factor $e^{-2\pi i s(n)/r}$, where $s(n) \in \mathbb{N}, s(n) \in [0, r-1]$.

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r} \left| a^k \mod N \right\rangle \tag{1.30}$$

It can be proven that the factor $e^{2\pi i s/r}$ is an eigenvalue of $|v_s\rangle$, which is an eigenvector of U.

$$U|v_s\rangle = e^{2\pi i s/r}|v_s\rangle \quad , \text{where} \quad \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle \tag{1.31}$$

The factor is dependent on r, as such it is possible to execute the Quantum Phase Estimation circuit, measuring the phase $\phi = s/r$. The circuit works by encoding the exponentiations of U by different even powers of two each conditioned on the value of a different qubit in state $|+\rangle$, which will experience a phase kickback. These control qubits are then processed by performing an IQFT subroutine, measuring in output the phase. The output is probabilistic, as such multiple estimates of ϕ might be registered, that must be tested in order to spot out the correct result. Applying the continued fractions algorithm onto ϕ will then allow to estimate the value of r up to a certain precision, dependent on the number of available qubits m for the QPE, where $m = O(n)$.

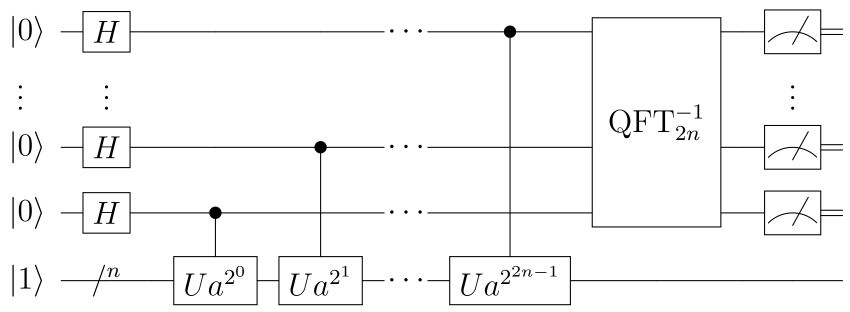


Figure 1.7: Order finding quantum subroutine for Shor's algorithm.

1.5.2 Grover's algorithm

Searching in an unordered database of size N for all the elements fitting a specific condition is classically ceiled by $O(n)$ comparisons. In 1996 Lov Grover proposed a quantum subroutine, later called *Grover iterate*, which would allow for the amplification of the amplitude associated to a desired output state whilst suppressing the undesired ones. The algorithm is described in the following list.

1. Starting from a quantum register of size n, where $N = 2^n$ is the size of the database, all qubits are set to the state $|+\rangle$, producing an equal superposition of all the output states in the quantum register.

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in 0,1^n} |x\rangle \tag{1.32}$$

2. The Grover iterate is applied $k \approx \frac{\pi}{4}\sqrt{N}$ times.

3. The result of the search is measured as the most probable output state after the circuit's execution.

The initial state superposition contains the target state, called $|w\rangle$ and all the other sates, represented by $|r\rangle$. It is possible to represent the state $|s\rangle$ in terms of polar coordinates by using $|w\rangle$ and $|r\rangle$ as axes.

$$\begin{aligned}|s\rangle &= \frac{1}{\sqrt{N}}\left(|w\rangle + \sum_{i\neq w}|i\rangle\right) = \frac{1}{\sqrt{N}}|w\rangle + \frac{1}{\sqrt{N}}\sum_{i\neq w}|i\rangle \\ &= \frac{1}{\sqrt{N}}|w\rangle + \sqrt{\frac{N-1}{N}}\frac{1}{\sqrt{N-1}}\sum_{i\neq w}|i\rangle = \frac{1}{\sqrt{N}}|w\rangle + \sqrt{\frac{N-1}{N}}|r\rangle \\ &= \sin\theta\,|w\rangle + \cos\theta\,|r\rangle\end{aligned} \tag{1.33}$$

The Grover iterate works by exploiting a phase oracle U_f in order to invert and amplify the probability amplitude associated to $|w\rangle$, all while reducing the ones associated to the other states in $|r\rangle$. In fact, given that $f(x = w) = 1$ is the only case that satisfies the condition, the effect of applying U_f corresponds to the reflection of $|s\rangle$ with respect to $|r\rangle$. Once this has been done, the new state $U_f\,|s\rangle$ is reflected with respect to $|s\rangle$, moving the state closer to $|w\rangle$. The overall observed effect is a rotation of the desired state by 2θ towards $|w\rangle$, as seen in Figure 1.8: the $|U\rangle$ is brought close and closer to the desired state *ketw*.

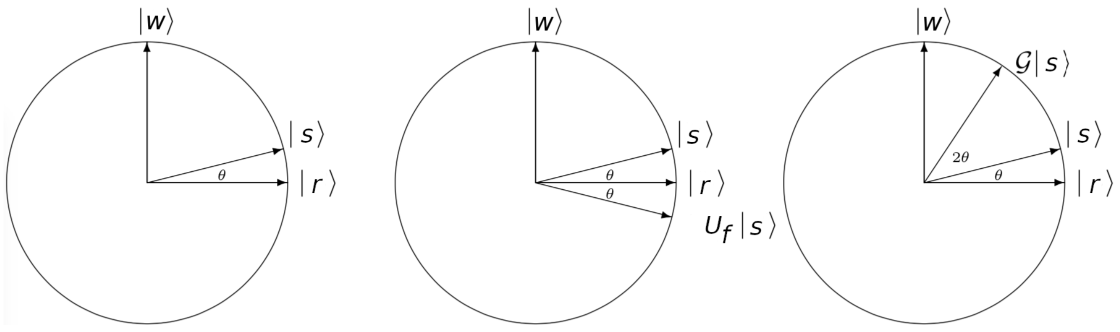


Figure 1.8: Grover iterate visualization.

The process is repeated k times, moving the target state closer and closer to $|w\rangle$. Any over repetition of the iterate will move the target state away and eventually return periodically to the initial state.

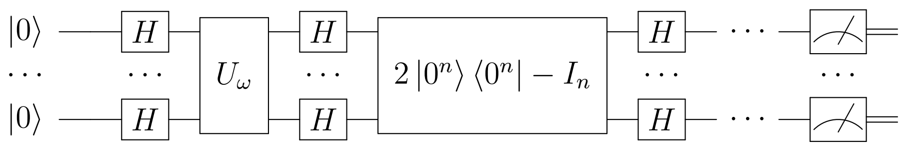


Figure 1.9: Grover's algorithm quantum circuit.

1.5.3 Bernstein-Vazirani's algorithm

This algorithm refers to the self named problem of finding the *n-bit* string encoded in an oracle which computes as output the dot product between the secret string and the input. In the classical case the solution requires to test n, one for each bit of s, by passing each time a string of all zeros apart from the current bit we are trying to determine.
The quantum solution to this problem only requires one execution of the circuit in Figure 1.10. A quantum register of size n, is initialized to the equiprobable superposition state, plus an additional ancilla qubit initialized to $|-\rangle$. The qubits

then undergo the U_f operator, which applies the custom function $f(x) = s \cdot x$. At last, measuring the *n-qubit* register yields the secret string s. By temporarily ignoring this extra qubit, it is possible to describe the state of the quantum register before measurement as

$$\sum_{z\in\{0,1\}^n} \left(\frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)+x\cdot z} \right) |z\rangle \tag{1.34}$$

Once collected the common factor x, is possible to rewrite the exponent of (-1) as $f(x) + x \cdot z = s \cdot x + x \cdot z = (s + z) \cdot x$, where the sum symbol denotes the XOR operator. If $z = s \rightarrow s + z = 0b0^n$, which means that the amplitude probability of all states different from $|s\rangle$ are zero. Measuring the registers will yield the string s with certainty.

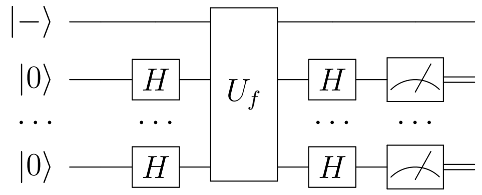


Figure 1.10: Bernstein-Vazirani's quantum circuit for an n sized bitstring.

1.6 NISQ era

At the time of writing, according to the definition of Preskill [20], quantum technology is in the Noisy Intermediate Scale Quantum era. This means that research has reached the advancements necessary to produce quantum devices which can host up to a little more than a hundred qubits, however they are still plagued by intrinsic and external device errors, respectively in the form of circuit level noise and cosmic rays, which will be discussed more in depth in the next Chapter. Moreover, no complete error correction code has been yet discovered, so one of the mitigation techniques is the use of surface codes and the usage of multiple computation iterations, averaging out the effect of noise. On top of this, the production process is still very prototypical, with gate fidelity metrics which still need substantial improvements.
Only once those issues will be solved, alongside a dramatic improvement in the number of qubits available for computation, it will be possible to consider the state of quantum computation *beyond* NISQ.

1.6.1 Qubit hardware implementation

Qubits, much like classical bits, can be encoded in a large number of physical devices by addressing specific properties. In particular, a qubit can be represented and controlled by means of any two state quantum mechanical system. The main qubit architectures used today are detailed in the following paragraphs.

Superconducting Josephson junctions

This technology is currently the best competitor for the realization of qubits. It exploits the quantum properties associated to an electronic device, called Josephson junction, made of two superconductors separated by a very thin insulating layer. Once this device is brought to cryogenic temperatures, it displays a free flow of *supercurrent* and the exchange of Cooper pairs across the insulating layer. This behaviour is modeled as a macroscopic quantum phenomenon, since it is observable at an ordinary scale. The major drawback comes from the low temperature requirements, which may hinder scalability and qubit reliability.
They are employed in quantum devices by IBM, Google, D-Wave Systems and Rigetti.

Photons

The promising paradigm of *Linear Optical QC* is based on converting the properties of photons into carriers of information. The main advantage linked to this technology is that it can be employed in both the contexts of quantum communication and quantum information processing. It provides the simplest approach for the implementation of quantum gates through the use of mirrors, phase shifters and beam splitters, and the universality of its computation model has been proven. The major drawback comes from the fact that in order to implement complex circuits the number of optical elements and operators needed might be resource inefficient, posing issues with general purpose scalability.
Photonic quantum devices are being developed by Xanadu, ORCA Computing and PsiQuantum.

Ion traps

This technology involves the usage of electromagnetic fields to suspend and confine ions in free space, addressing their electronic states to encode quantum information. Operational gates are applied through the usage of lasers to introduce a coupling between the qubit states for superposition, or between the qubit and the state of the external motion to produce entanglement. Despite being able to satisfy all of DiVincenzo's criteria (Section 1.4.3) and theoretically easy to scale, they are

intrinsically hard to implement, as most gates would need to be composed of a large number of basic CNOT and single qubit rotation gates. Moreover, the decoherence times, alongside the very short lifetime of these qubits are still a major drawback over its competitors.
These devices are currently being constructed and tested by IonQ, Alpine Quantum Technologies and Quantinuum.

Quantum dots

Sometimes referred to as artificial atoms, quantum dots are among the older technologies for the implementation of qubits. Given their larger size with respect to real atoms, they are more easy to control and to couple, encoding the qubit state in the spin of the dot. However, despite their very convenient properties, they have been superseded by other technologies which allowed simpler scalability and relatively lower costs.

1.6.2 Main SDKs

Over the years, many libraries and SDKs have been developed to allow for computing quantum simulations and interfacing programmers with quantum device backends, of which a short list will be presented hereafter.

Qiskit

The quintessential open-source Python library for quantum computing, it has allowed since 2016 to interface directly with IBM's Quantum Experience device backends over the cloud. It supports multiple simulators, even with realistic noise models collected daily from real quantum devices.

Pennylane

Xanadu's cross-platform Python library for quantum circuit simulation and quantum machine learning. It allows to access IBM, Strawberry Fileds, AWS Braket and Rigetti Forest, among many others, through the usage of a plethora of purpose-made plugins. The Strawberry Fields plugin is one of the few that allow for continuous variable quantum optical circuit simulation, altohugh it grants no access to Xanadu's physical devices.

Cirq

Google's local quantum Python simulation library, allows for the definition of simple circuits, however it does not yet grant access to any device backend.

Quantum Development Kit

Under the .NET Q# programming language, this SDK allows for definition and simulation of statevector based quantum circuits.

AWS Braket

This library, closely connected ot Pennylane, allows on-demand access to quantum simulators and devices over AWS's cloud.

D-Wave Ocean

D-Wave's Pyhton library for writing quantum circuits and runnong them over the cloud on D-Wave's quantum annealer devices.

Rigetti Forest

Made up of the pyQuil and the Forest SDK, it allows access to the wavefunction simulator, the Quantum Virtual Machine and Rigetti's transmon based devices.

Chapter 2

Quantum Fault Injection

2.1 Classical fault injection

It has become a common belief at the eyes of the public, especially in recent years, that computers are always reliable and can never fail. However, as experts in the field know very well, classic computing devices are in fact not reliable *per se*, but rather have been studied and hardened over years of research to make them so in most scenarios.
The reason as for why classical computers and in general digital circuits are not intrinsically reliable lies in the fact that the nanoscale components of which they are made of, CMOS transistors, work by harnessing extremely small electrical charges. Such charges are so small in fact that they can fluctuate by large amounts when these nanoscale devices get struck by a charged particle coming from the outside environment. These variations at the hardware level can force the logical gates of the CMOS to open or close, thus producing a logical inversion of state, called Single Event Upsets or simply *bit-flips* in the classical domain.
Earth is bombarded every second by thousands of cosmic rays per square meter, with more common and less dangerous particles having energy in the range of $1 - 10$ GeV [37], whilst other much rarer particles can amount to up to 10^{20} eV [38]. It has been shown that most of these particles possess a high enough energy to produce a SEU [39].
However, not every CMOS is equally important in the architecture of modern cyber-physical systems: it is enough to think of the harmless effect of a bit-flip onto an unused memory area compared to the same bit-flip onto a used one.
In order to understand how a digital circuit may perform under such stress and correct its behaviour, one needs to simulate the effect of a fault on a real circuit: as such, in the late 1970s, the first hardware fault injection studies have been made [40]. This approach led to the creation of Error Detection Codes (EDC) and Error

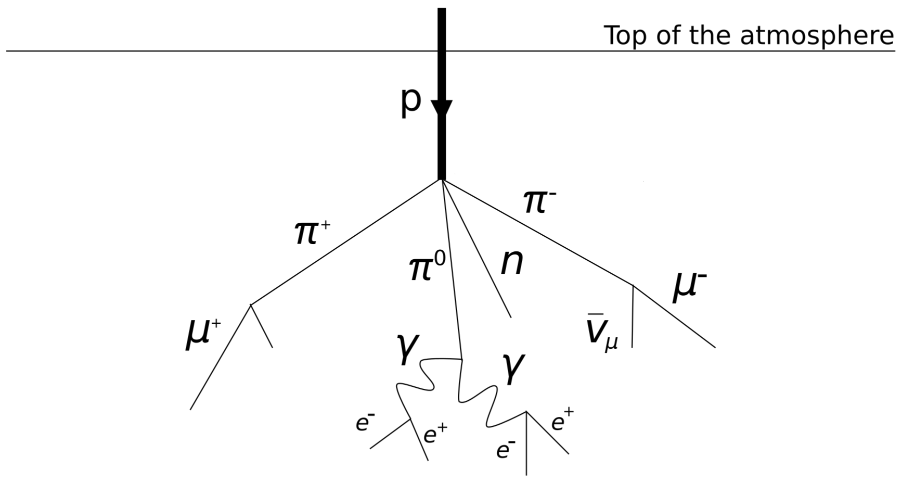


Figure 2.1: Cosmic ray collision with Earth's atmosphere and particle generation, adapted from [36].

Correction Codes (ECC).
Today, fault injection and reliability is a large field of research, branching from hardware to software and network protocols.

2.2 Quantum fault injection

NISQ computers are, as their name suggests, prone to the effects of noise in the circuit, which produces two kinds of retention errors, T1 and T2. The T1 error, or *spin-lattice coherence time*, refers to the natural decay time of an excited qubit in state $|1\rangle$ to the ground state $|0\rangle$. The T2 error, or *spin-relaxation process*, represents the minimum time delta before a qubit's state gets affected by external interference or by other neighbouring qubits.
Different hardware architectures and topologies boast a diverse range of T1 and T2 values. In the recent years retention times went from the order of a few nanoseconds to hundreds of microseconds. This is mainly due to new and improved Quantum Error Correction (QEC) codes, which compensate for the effect of noise at the cost of adding redundancy in the number of physical qubits needed to represent a fault tolerant logical qubit [41, 42].
Current quantum devices are susceptible to an even wider range of particles with respect to classical ones, since even lower energy and more frequent ones like photons and muons, which leave classical devices unaffected, can have a large impact onto quantum states [43]. A simulation of a 275 MeV particle striking Silicon, shown in Figure 2.2, produced a charge deposition in the form of electron-hole pairs. It can be noticed how the intensity of the charge is heavily dependent on the

distance from the point of impact, which is on a scale similar to the size of modern superconducting qubits: thus, being closer to the particle's leftover trail causes a rise in the magnitude of the fault. [44].

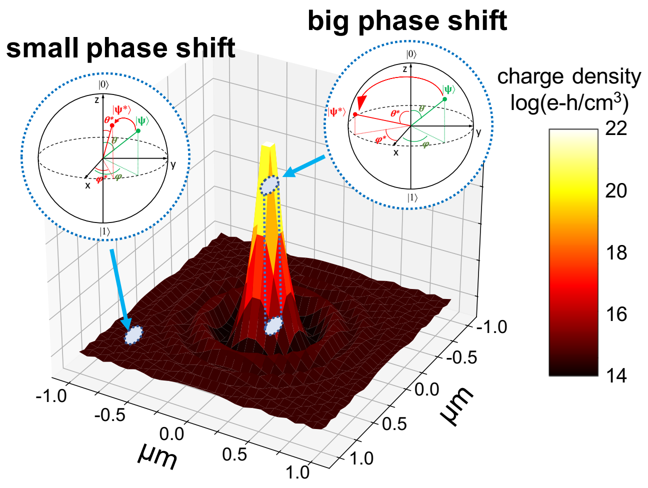


Figure 2.2: Deposited charge density due to particle impact on Silicon [44].

Recent studies have shown that such ionizing radiation has a strong effect on the state of qubits [45, 46, 47], often reducing their fault tolerance by a considerable margin. Being larger in size, trapped ion qubits have proven to be the most architecturally resilient to low energy particles [43], even if still extremely sensible to them. No study for rarer high energy particles has been conducted yet.
Cosmic particle impacts are stochastic in nature and current QEC cannot deal efficiently with their effects. As such, ionizing radiation is currently being, and will be in the future, one of the major antagonists in achieving fault resilient quantum computers.
Taking in consideration the huge financial investments which revolve around quantum computing research, on the order of billions of dollars, it is of foremost importance to start investigating the fault resilience of current devices in order to lay the research foundations for developing new and improved quantum error correcting codes.

2.2.1 Quantum fault model

The works of Vepsalainen *et al.*[47] and Wilen *et al.*[48] show the effect of ionizing radiation on qubits embedded into superconducting materials. The additional charge which gets deposited by these stray particles produces an excitement which alters the state of the qubit proportionally to the deposited charge.

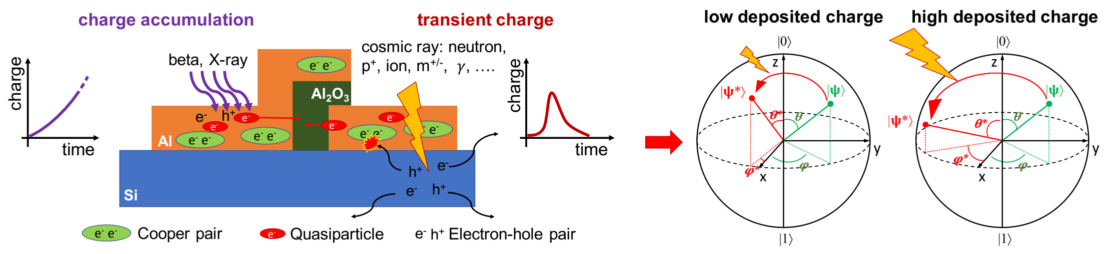


Figure 2.3: The impact of ionizing radiation onto a superconducting Josephson junction and its logical effect on the qubit's state [49].

Figure 2.3 gives insight onto the effect of ionizing radiation onto a transmon qubit: these particles increase the amount of hole-electron pairs in the *Al*-film on *Si* substrate. However, not all particles produce the same kind of fault. In fact, higher energy particles like γ-rays, β-rays and X-rays gradually contribute to a permanent phase shift [50], eventually leading to a collapse of the qubit [47]. On the other hand, lower energy particles, like neutrons and heavy ions, produce a transient shift whose effect vanishes over time.
Such modification will then be propagated through the quantum ciruit, eventually making the output diverge from the expected one.
These faults are parametrizable by the polar and azimuthal angles ϕ and θ. As such, they can be represented at the logical level of a circuit as a *U3* quantum gate.

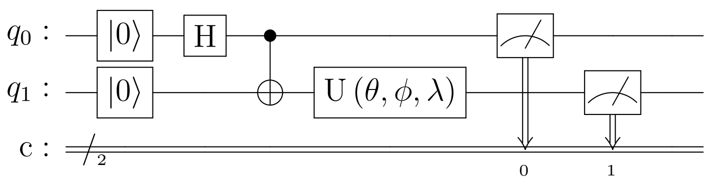


Figure 2.4: Logical representation of a fault on the Bell-pair generation circuit.

Although physical shielding of such particles has been attempted, its requirements are too impractical for building scalable quantum computers in the future. In particular, although X-rays can be easily blocked by a thin lead foil [51], shielding heavier ions and neutrons would require various meters of such a lead or concrete shield and blocking muons would mean placing the quantum device many kilometers under the Earth's crust [48]. Consequently, similarly to what has been done with classical computing devices, the shielding route cannot be the only solution.
Currently, there is no study on the incidence of cosmic ray induced faults in quantum computers, however, similarly to what is done for Architectural Vulnerability Factors and Program Vulnerability Factors, it will be assumed that a fault has occurred, regardless of its cause, studying its effects on the output of the quantum

circuit.
It has also been shown that a cosmic ray can affect more than a single qubit at once. In particular, it is easy to assume that topologically connected elements in the structure of a quantum core will be physically closer to the point of impact. As such, the double fault model accounts for this case by introducing an additional phase shift onto a secondary qubit, whose magnitude is ceiled by the one of the first phase shift.

2.2.2 Quantum Vulnerability Factor

The first step for being able to evaluate the effect of a fault onto any circuit is to have a comparison metric to work with.
The outputs of a quantum device are structured as probability distributions of quantum states.

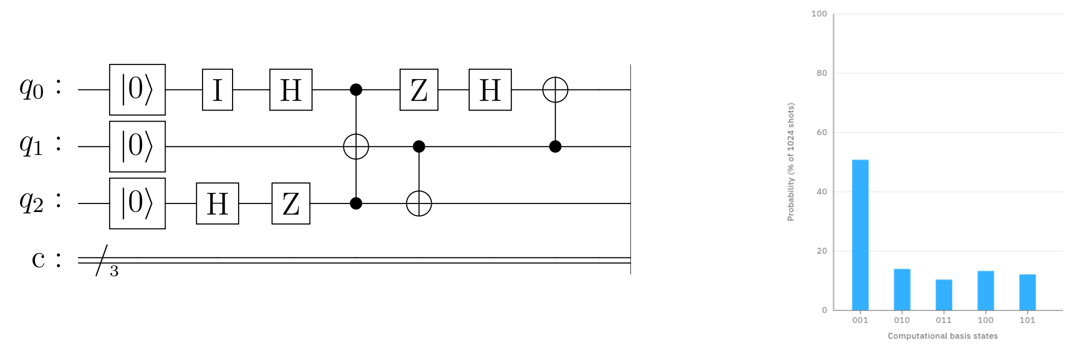


Figure 2.5: The output histogram of a random 3 qubits circuit.

The currently most used metric to perform such evaluation is the Probability of Successful Trials, which defines the probability of the correct or golden state of the circuit, as seen in [52, 53, 54, 55]. PST simply considers how probable the correct output is with respect to the total number of trials, as seen in Equation 2.1. As such, it does not provide any insight on the separation between the correct output state and the other incorrect states, because the information needed to spot out a fault over noise is absent. Moreover, the PST metric requires the definition of a *reliability threshold*, which is not known a priori or inferrable from the device or circuit: this may lead to an improper masking of the effect of some lighter faults, or on the contrary, render the metric too susceptible to noise.

$$PST = \frac{Number\ of\ successful\ trials}{Total\ number\ of\ trials} \tag{2.1}$$

The recent paper by Oliveira *et al.*[49] tried to solve this issue by proposing the Quantum Vulnerability Factor metric, QVF for short, which extends the PST through the use of the *Michelson contrast* [56]. This contrast provides a figure for the discriminability of the correct state with respect to all the others in the output histogram of a circuit. The contrast computation follows Equation 2.2, where $P(A)$ represents the percent chance of measuring the correct state and $P(B)$ represents the same chance for the most probable wrong state.

$$Contrast = \frac{P(A) - P(B)}{P(A) + P(B)} \ , \quad P(A) = PST \tag{2.2}$$

Given the fact that it is possible to measure a value of $P(B) > P(A)$, the contrast's range is $[-1, +1]$. It is furthermore possible to use the QVF with circuits which provide multiple correct outputs by simply aggregating all the correct states's porbabilities into a sum.
To better resemble the current AVF and PVF metrics's characteristics, the QVF is thus shifted to the $[0, +1]$ range and is copmputed according to Equation 2.3.

$$QVF = 1 - \frac{Contrast + 1}{2} \tag{2.3}$$

To sum up, the QVF metric is an indicator of the vulnerability of a quantum circuit ranging from zero to one: values close to zero mean that the circuit is extremely fault resilient, while values close to one mean that the circuit's output is consistently inverted with respect to the desired output. QVF values around 0.5 instead mean that the intelligibility of the output has been lost, as there is not a clear separation between correct and faulty results anymore.

2.3 QuFI

In another recent paper, Oliveira *et al.*[57] introduced a framework called *QuFI* to test their statements about QVF. This Python based tool takes advantage of IBM's Qiskit library [58] to generate single and double fault circuits and later on simulate them either through the use of the *Aer* simulators or run them directly onto real quantum devices.
Part of the work of my thesis amounted to rewriting the QuFI from scratch, using as a backend the Pennylane library [59] provided by Xanadu. The main objectives of the work were:

- Making QuFI hardware independent, as Pennylane supports backends from multiple providers, allowing testing outside of Qiskit.

- Making computations faster, also through parallelization.
- Accessing better support for QML circuits.
- Refactoring the code as an easily accessible framework for computation and visualization.

2.3.1 Injection

The injector models both single and double faults, as per demonstrated by recent papers on the topic of radiation induced effects in quantum devices. Differently from classic *bit-flips*, quantum faults are parametrized by magnitude of the impinging charge on the Josephson junction. This means that a fault injector must account for all the possible phase shift combinations allowed by the Bloch sphere.
As stated previously in subsection 2.2.1, a phase shift fault can be represented by the parametrizable U3 gate, which takes as arguments the the polar angle ϕ, the azimuthal angle θ and the global phase angle λ. Given that the coordinate space on the Bloch sphere is continuous, a quantization with a step of $\frac{\pi}{12}$ has been performed, totaling 312 angle combinations across θ and ϕ.

$$U3(\theta,\phi,\lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{2.4}$$

where the parameters of the U gate range as

- $\phi = [0{,}2\pi[$ with $\frac{\pi}{12}$ steps.
- $\theta = [0,\pi]$ with $\frac{\pi}{12}$ steps.
- $\lambda = 0$

The updated version of QuFI supports input quantum circuits defined either as Pennylane *QNode* objects, Qiskit *QuantumCircuit* objects, or as OpenQASM strings. Internally, the injector works with QNode objects, so it performs conversions accordingly.
For each circuit, the first step is deriving all its single fault versions. This is done by looping over all gates in the *tape* associated to the QNode and by inserting a U3 gate after each one of them. Each gate gives rise to a number of output fault circuits equal to the number of qubits it acts onto. For example, a Hadamard gate will have a single fault circuit derived from it, whilst a CNOT gate will have two, one with the fault gate after the control qubit and the other after the controlled qubit, as seen in Figure 2.6. However, no circuit with a gate after both qubits will be produced, as it is not contemplated by the single fault model, but rather by the

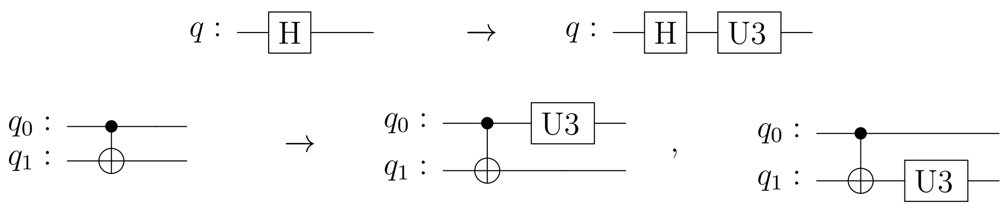
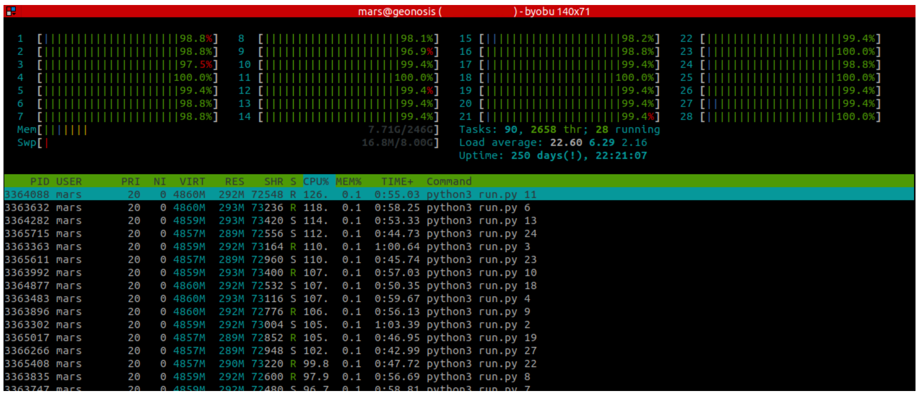


Figure 2.6: The fault insertion on a Hadamard and a CNOT gate.



```
mars@geonosis (                    ) - byobu 140x71

  1  [||||||||||||||||||||||98.8%]   8  [||||||||||||||||||||||98.1%]   15 [||||||||||||||||||||||98.2%]   22 [||||||||||||||||||||||99.4%]
  2  [||||||||||||||||||||||98.8%]   9  [||||||||||||||||||||||96.9%]   16 [||||||||||||||||||||||98.8%]   23 [|||||||||||||||||||||100.0%]
  3  [||||||||||||||||||||||97.5%]   10 [||||||||||||||||||||||99.4%]   17 [||||||||||||||||||||||99.4%]   24 [||||||||||||||||||||||98.8%]
  4  [|||||||||||||||||||||100.0%]   11 [|||||||||||||||||||||100.0%]   18 [|||||||||||||||||||||100.0%]   25 [|||||||||||||||||||||100.0%]
  5  [||||||||||||||||||||||99.4%]   12 [||||||||||||||||||||||99.4%]   19 [||||||||||||||||||||||99.4%]   26 [||||||||||||||||||||||99.4%]
  6  [||||||||||||||||||||||98.8%]   13 [||||||||||||||||||||||99.4%]   20 [||||||||||||||||||||||99.4%]   27 [||||||||||||||||||||||99.4%]
  7  [||||||||||||||||||||||98.8%]   14 [||||||||||||||||||||||99.4%]   21 [||||||||||||||||||||||99.4%]   28 [|||||||||||||||||||||100.0%]
  Mem[||||||||                                      7.71G/246G]   Tasks: 90, 2658 thr; 28 running
  Swp[|                                            16.8M/8.00G]   Load average: 22.60 6.29 2.16
                                                                  Uptime: 250 days(!), 22:21:07

    PID USER       PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
3364088 mars        20   0 4860M  292M 72548 R 126.  0.1  0:55.03 python3 run.py 11
3363632 mars        20   0 4860M  293M 73236 R 118.  0.1  0:58.25 python3 run.py 6
3364282 mars        20   0 4859M  293M 73420 S 114.  0.1  0:53.33 python3 run.py 13
3365715 mars        20   0 4857M  289M 72556 S 112.  0.1  0:44.73 python3 run.py 24
3363363 mars        20   0 4859M  292M 73164 R 110.  0.1  1:00.64 python3 run.py 3
3365611 mars        20   0 4857M  289M 72960 S 110.  0.1  0:45.74 python3 run.py 23
3363992 mars        20   0 4859M  293M 73400 R 107.  0.1  0:57.03 python3 run.py 10
3364877 mars        20   0 4860M  292M 72532 S 107.  0.1  0:50.35 python3 run.py 18
3363483 mars        20   0 4860M  293M 73116 S 107.  0.1  0:59.67 python3 run.py 4
3363896 mars        20   0 4860M  292M 72776 R 106.  0.1  0:56.13 python3 run.py 9
3363302 mars        20   0 4859M  292M 73004 S 105.  0.1  1:03.39 python3 run.py 2
3365017 mars        20   0 4857M  289M 72852 R 105.  0.1  0:46.95 python3 run.py 19
3366266 mars        20   0 4857M  289M 72948 S 102.  0.1  0:42.99 python3 run.py 27
3365408 mars        20   0 4857M  290M 73220 R 99.8  0.1  0:47.72 python3 run.py 22
3363835 mars        20   0 4859M  292M 72600 R 97.9  0.1  0:56.69 python3 run.py 8
3363747 mars        20   0 4859M  292M 72480 S 96.7  0.1  0:58.81 python3 run.py 7
```

Figure 2.7: *htop* of a 28 core machine running QuFI simulations, automatically load balanced.

double fault one.
The injector then checks for the presence of a *coupling map*, a dictionary containing the mapping between the logical to physical qubits and viceversa. If it is not found, the single fault circuits are immediately run. On the other hand, if a coupling map is defined, a second insertion pass will be done, by adding another U3 gate after each qubit physically connected to the main fault. The coupling map can be custom defined, or it can be retrieved by calling the *qufi.get_qiskit_coupling_map* method, which transpiles the circuit on a given Qiskit device and returns its map.

2.3.2 Execution

The execution can then be done sequentially or in parallel, the first suggested for very small circuits or low performance machines and the latter suggested for large circuits and high core counting machines.
The sequential version, called with the *qufi.execute_over_range* function, takes as input a list of angles for θ_0, ϕ_0, θ_1 and ϕ_1, then generates locally all possible angle combinations and circuits and executes them in sequence.
The parallel version, runs the *launch_campaign.py* script, which generates all the angle combinations and splits them into multiple lists of tuples with the form

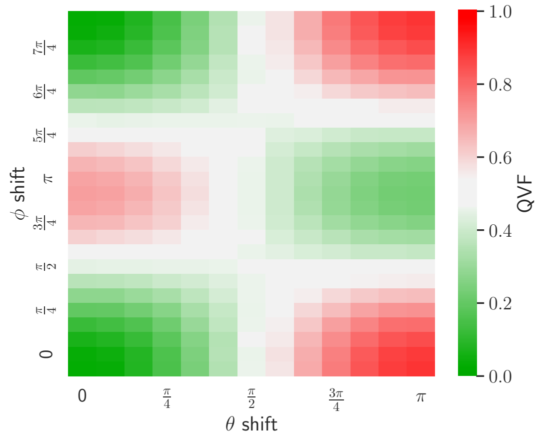
(θ_0,ϕ_0,θ_1,ϕ_1). Then it spawns multiple shells running the *run.py* script, taking as input one of the tuple lists, generating all the circuits and then running them in batches, in order to reduce as much as possible the computation times.
In both cases, the backends used to run the circuits are Pennylane's *lightning.qubit* for the noiseless executions and the Pennylane-qiskit *Aer* simulator for the noisy ones. Other backends can be specified according to Pennylane's API.

2.3.3 Visualization

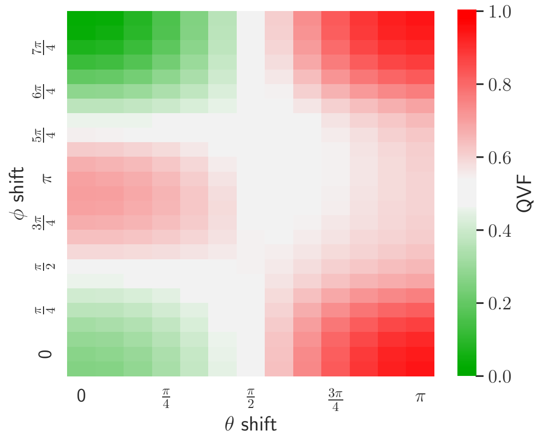
Once the simulations have been completed, the results are processed by means of other QuFI methods, which are responsible for loading the results in memory, filtering single injections from double injections and producing output histograms, heatmaps and delta maps.
The function which processes all data is called as *qufi.generate_all_statistics.*
The histograms represent the distribution of QVF values among all the angle combinations, and compare the results of single and double fault injections when present. The heatmaps present the actual QVF average values of all injections across all angle combinations, using a gradient colour scheme between green (low QVF) to red (high QVF), with relative thresholds of . The delta maps, available only for double fault injections, represent the difference between the double and single fault heatmaps, as $\Delta QVF = QVF_{Double} - QVF_{Single}$. An example for each is shown on Figure 2.8. Additional histograms' metrics are presented in Table 2.1.

Metric / Fault type	Mean	Stnd. dev
Single fault	0.46	0.20
Double fault	0.55	0.18

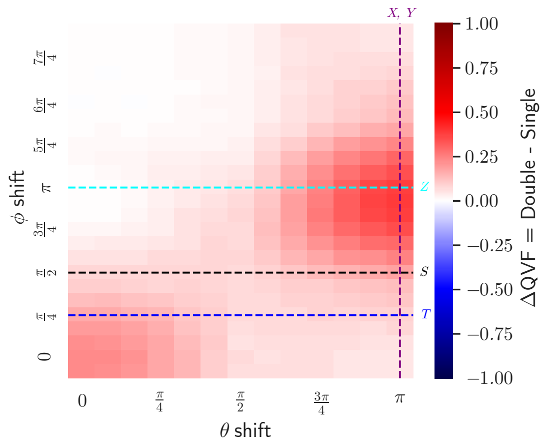
Table 2.1: Bernstein-Vazirani's Algorithm: histograms statistics.



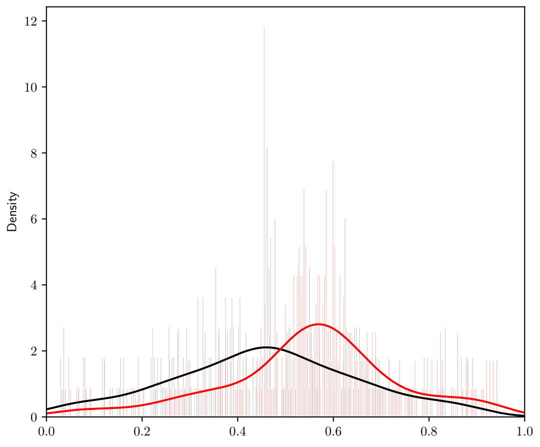
Single Fault QVF_s.



Double Fault QVF_d.



ΔQVF=QVF$_d$−QVF$_s$.



Single (black) and double (red) fault histograms.

Figure 2.8: Bernstein-Vazirani's Algorithm: Single fault injection heatmap, double fault injection heatmap, Delta-map and Histogram.

Chapter 3

Machine Learning

3.1 Classical Machine Learning

Machine Learning is a specific field of research in the context of Artificial Intelligence which is devoted to the definition of trainable computational models able to process information and autonomously extract features which let them optimize their performance on a specific task.
In the last twenty years, model performance and capabilities have improved to the point where they have permeated into almost every aspect of our lives: self-driving cars, weather forecasting, voice recognition and synthesis, stocks prediction models, medical imaging recognition, email spam filters, recommendation algorithms and many other more.
This trend is only expected to increase, as data availability and computing capabilities continue to rise. In the words of Tom Michael Mitchell [60], the forefather of ML, such *learning* approach is defined as

> *'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.'*

The basic elements needed to perform the learning process can thus be listed more formally as

- *Dataset*: the aggregation of structured data to be used for training onto a specific task, usually subdivided into training, validation and test sets.
- *Model*: the abstract structure of computation layers able to self update its internal parameters through gradient descent and backpropagation.
- *Accuracy metric*: the function which provides a numerical value representing the separation between the current performance and the expected one.

- *Optimizer*: the algorithm which allows the model to escape unfavourable fitness landscapes and speeds up the learning process.

The goal of the procedure is to process the training subset of the dataset to extract relevant features from it, compare its performance against the expected behaviour or the previous iteration on the validation set and then repeat until either a certain accuracy threshold has been reached, the model stops improving or the defined number of iterations has been completed. It generally follows these steps

1. *Data collection*: gather the data relative to a specific learning task.
2. *Data preprocessing*: perform balance checks (all classess are represented with the same frequency), normalization and possibly data augmentation if not performed directly internally by the model.
3. *Dataset partitioning*: split the dataset into training, validation and testing partitions.
4. *Model definition*: construct the model for tackling a specific task. This generally defines also the learning approach (supervised or unsupervised).
5. *Training*: define a loss function for gradient descent and an optimizer among other parameters, then train the model through backpropagation.
6. *Testing*: compute the performance metrics of the new model onto the test subset and decide whether the model is sufficiently good at completing the task. If not, either continue the training, change the model's hyperparameters or start from scratch with a new architecture.

It is important to state that the hereby proposed structure for training ML models has been simplified and limited to what is needed for the scope of this thesis, that being classification, regression and clustering. However, there are many other branches of ML which follow different paradigms, such as generative adversarial networks, reinforcement learning networks and natural language processing networks, just to name a few.
Since training times tend to be extremely long, especially when dealing with very large datasets or complex models, it has been shown that it is possible to *convert* an already trained model from one field to another, provided that the two domains share some common features. For example, it could be possible to take a model trained to recognise handwritten digits and transfer learn it with a dataset of handwritten characters. The process amounts to simply removing the last layers in the network and substituting them with new ones, according to the desired output shape. The model is then trained again, but the time required to do so will be much smaller with respect to before since a good portion of the weights comes

from the previous model. As such, computation times are significantly reduced, whilst converging to accuracy levels in the same order of magnitude as for the same model trained from scratch.

3.1.1 Neural Networks

One of the first approaches ever proposed for ML, neural networks try to mimic the structure of the human brain in order to process data. The basic element of such networks is in fact the artificial neuron, modeled as a mathematical function able to provide a specific output to a given input. It is composed of a input, a computation unit, a bias, weights, a threshold and an output. Moreover, the neuron is characterized by a nonlinear activation function, which defines whether it should forward an output or not with respect to the given threshold. The most used activation functions are the *sigmoid*, the *Rectified Linear Unit* (or ReLU), the *leaky ReLU* and the *hyperbolic tangent*.

$$\begin{aligned} \sigma(x) = \frac{1}{1+e^{-x}} \qquad &, \qquad ReLU(x) = max(0,x) \\ Leaky\ ReLU(x) = max(\alpha x, x) \qquad &, \qquad tanh = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \end{aligned} \tag{3.1}$$

As seen in Figure 3.1, it is easy to make comparisons among the components of the two neuron models. In fact the dendrites act as the input receptors (for the x_i values and the bias), the soma is the computation unit (comprised of summation and activation functions) and the synapse represents connecting the output of the neuron to all the other subsequent neurons in the network.

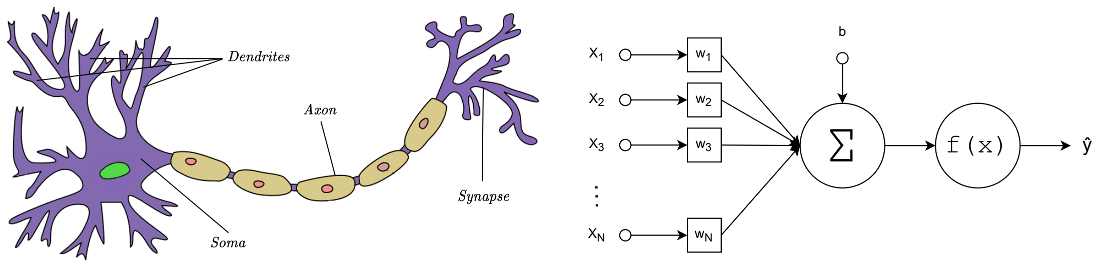


Figure 3.1: Brain and artificial neuron models side by side.

A parallel layer of such neurons is defined as a *fully connected*, or *dense*, layer. The inputs of the network are connected to all of the neurons which make up the first fully connected layer. The outputs of the first layer are connected in the same way to all the inputs of the subsequent layer, and so on until the end: this kind of architecture is defined as feedforward. Since they are not directly addressable, the layers in the middle of the network's architecture are called *hidden* layers. Once the number of hidden layer increases over a certain threshold, such a network is said to be a *deep neural network*.

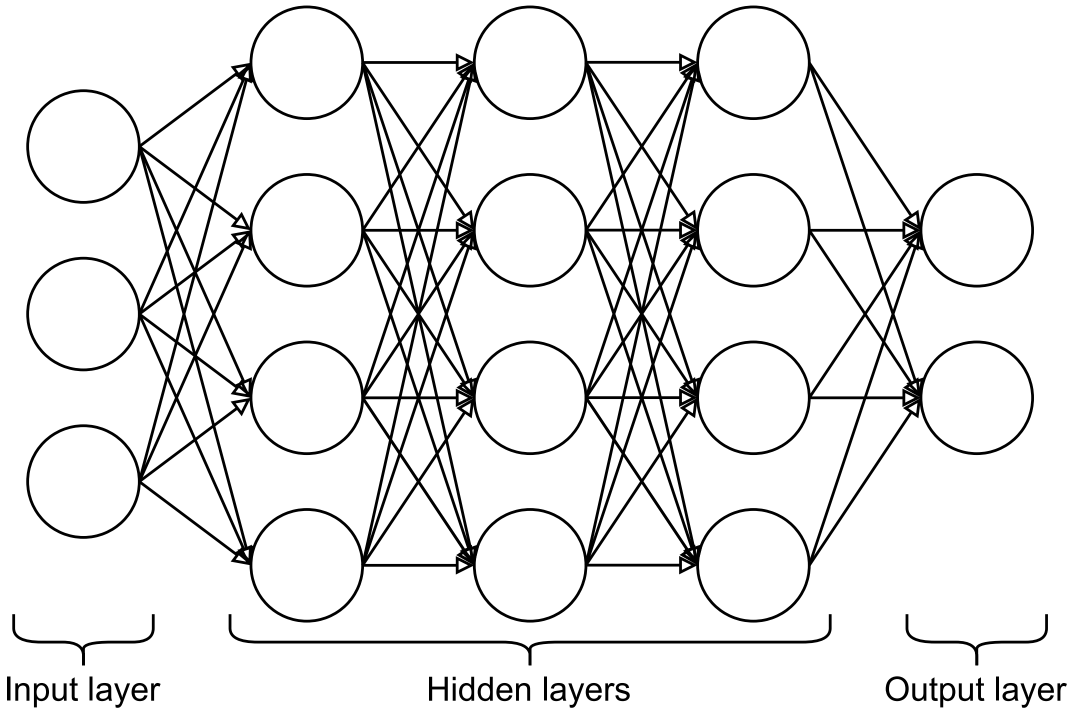


Figure 3.2: A binary classifier multilayer perceptron neural network architecture.

In the case of logistic regression, the activation of a single neuron is computed according to Equation 3.2, where N is the number of inputs received by the neuron, w_i is the weight associated to a given input, x_i is the input itself and b_i is the bias or minimum threshold associated to that neuron. $\sigma(z)$ is the activation function, whose output $\hat{y}$ determines the state of the neuron.

$$z = \sum_{i=1}^{N} w_i x_i + b_i \quad \rightarrow \quad \hat{y} = \sigma(z) \tag{3.2}$$

Once the *forward pass* has been completed, a cost function is used in order to compute the distance between the predicted labels and the expected ones in the case of supervised learning (classification) or to see if the performance improved (clustering and regression). One of such functions is the *mean squared error*, which is none other than the euclidian distance between the predicted and the actual

label in the feature space.
Finally, a backward propagation step is employed as a mean to update the weights of each neuron and the iteration is repeated, as seen in Figure 3.3 with a small binary classifier with MSE cost function.

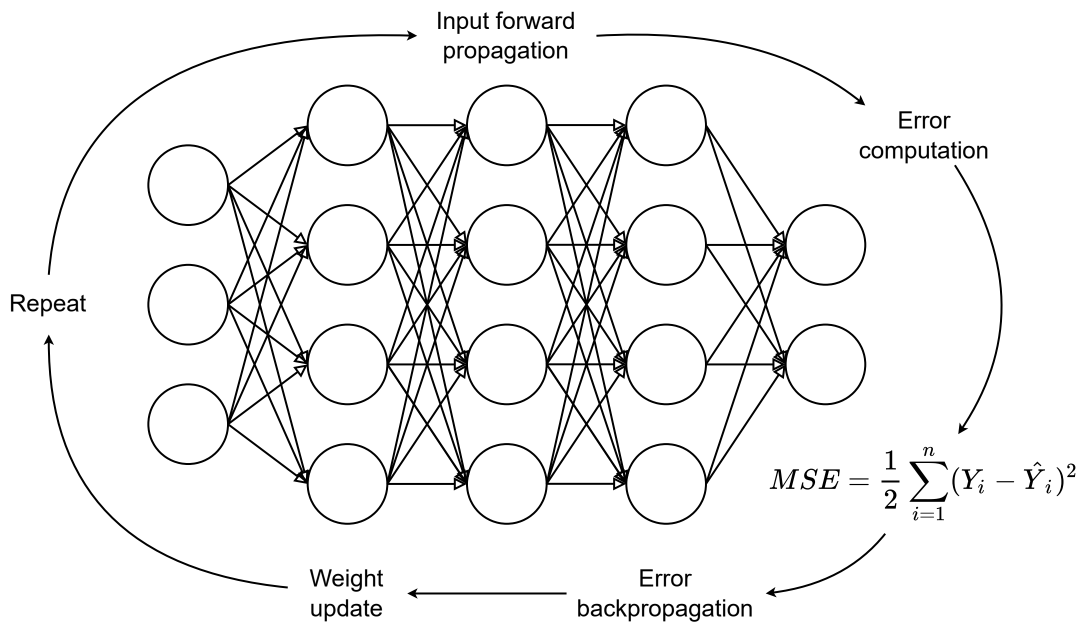

Input forward propagation
Error computation
$MSE = \frac{1}{2}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$
Error backpropagation
Weight update
Repeat

Figure 3.3: Forward and backward propagation steps in a basic neural network.

3.1.2 Support Vector Machines

Support Vector Machines, or SVMs for short, are supervised learning models first introduced by Cortes *et al.*[61] which were initially developed to be used as either regression or classification algorithms. They are amongst the best performing models for prediction and in their first inception acted as linear classifiers: given a dataset of *n-dimensional* vectors they compute the *(n-1)-dimensional* maximum margin hyperplane separating two classes. This posed a limit on the shape of the dataset to classify, as non-linear ones would not be tractable. However, thanks to the *kernel trick* (3.3) it is possible to move the dataset to a higher dimensional space with a mapping function and thus ease the computation of the hyperplane.

$$k(x, x^{'}) = \langle \phi(x) | \phi(x^{'}) \rangle \mathcal{V} \tag{3.3}$$

SVMs have been later adapted to act as unsupervised clustering algorithms [62], by employing support vector statistics to map common features in the data.

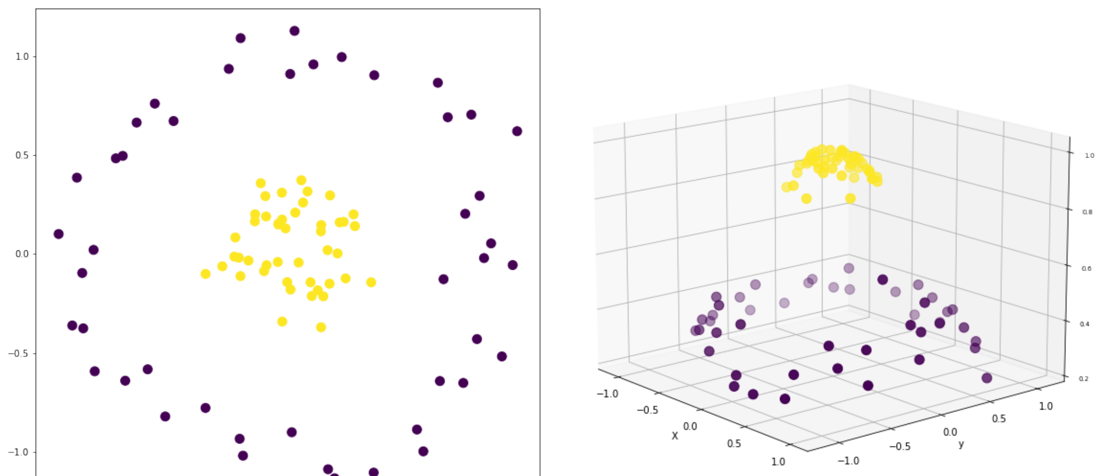


Figure 3.4: The effect of the mapping function on the dataset of an SVM. By moving from two to three dimensions, the hyperplane's complexity goes from quadratic to linear.

The kernel function $k(x, x^{'})$, in the space $\mathcal{X}$, expresses the distance between two datapoints and can be defined in terms of an inner product of elements from another inner product space $\mathcal{V}$ by employing a feature mapping function $\phi(x)$ as seen in Equation 3.3. The advantage comes from the fact that there is no need to actually derive the $\phi(x)$ function directly before computation, speeding up the process by a large margin.
Regardless of its definition, the kernel function is then used as a term in the weighted sum of similarities between the unlabeled datapoint and the labeled elements as follows.

$$\hat{y} = \operatorname{sgn} \sum_{i=1}^{n} w_i y_i k(x, x^{'}) = \operatorname{sgn} \sum_{i=1}^{n} w_i y_i \langle \phi(x) | \phi(x^{'}) \rangle \mathcal{V} \tag{3.4}$$

The main fields of application for SVMs are classification and segmentation of images, satellite data processing and biology.

3.1.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are learning models specifically developed to handle tensor-structured data, of which the most common ones are images. These networks take advantage of the mathematical multi-dimensional convolution operator, which uses a kernel tensor to filter out specialized features from the input.

Multiple subsequent applications of this filtering process allow for the extraction of higher and higher level structures, which eventually get vectorized and fed to a fully connected layer which performs the classification.
In short, CNNs are simplified feature extractors with respect to the MLP, which are less prone to overfitting thanks to regularization. Their architectures, in fact, share many similarities, with the only difference being the fully connected hidden layers of the MLP being substituted by pairs of convolutional and pooling layers.

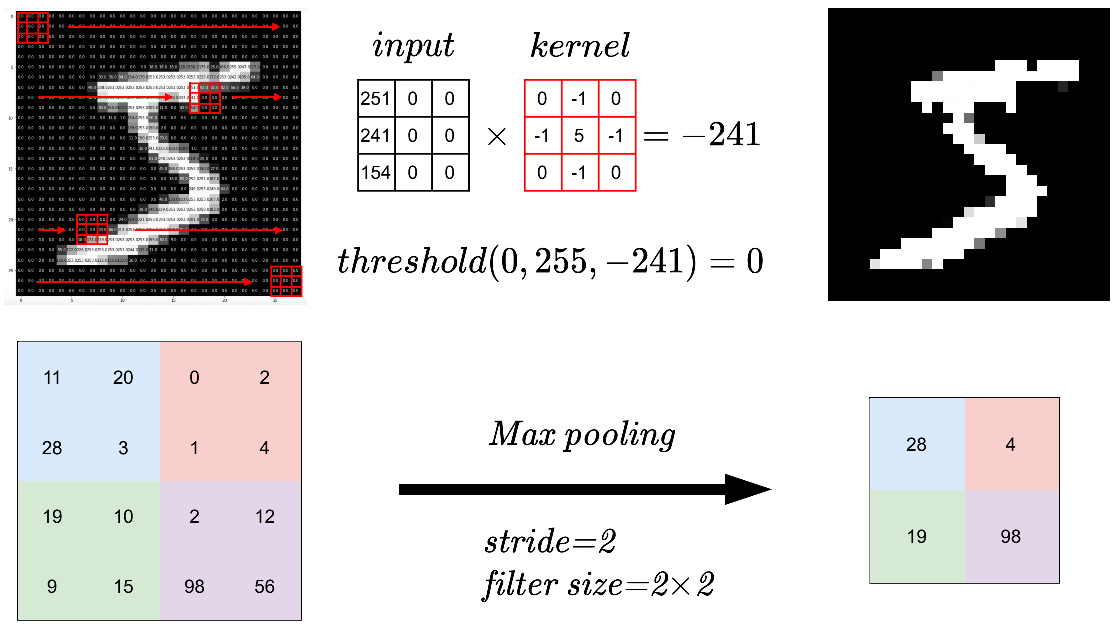


Figure 3.5: Graphical representation of the convolution (top) and pooling (bottom) operators.

The convolution operation consists of tensorial element by element multiplications and sums between the kernel, which *moves* along the input tensor, and the input tensor itself, as seen in Figure 3.5, followed by a thresholding operation. Historically, the size of the kernel is odd, varying between 3×3 and 7×7, however successive developments also demonstrated the applicability of even sized kernels [63].
Subsequently, the pooling layer is responsible for reducing the size of the tensor in order to remove lower level features from the inputs of the next layers, lowering the number of features to be learnt in the successive layers. This can be done with many approaches, the two most used ones being *max pooling*, which halves the spatial resolution of two of the dimensions of the tensor by picking the maximum value among a subgrid, and *average pooling*, which intuitively performs the same dimensionality reduction by computing the mean among the values in the subgrid. Both these operators are characterized by some not trainable hyperparameters

which are defined at the architectural level, such as:

- Number of filters: the *depth* of the convolution, not applicable to pooling layers.
- Filter size: the dimension of the kernel or of the subgrid.
- Stride: the distance in pixels between applications of the filters. Generally it equals to 1 in convolutions and equals to the filter size in pooling layers.
- Padding: the addition of a contour of pixels with value zero onto the input tensor to avoid the dimensionality reduction caused by the convolution. Not applicable to pooling layers.
- Activation function: the function which adds non linearity to the input tensor.

Depending on the size of the input, of the kernel and on the other previously described hyperparameters, the output of a convolution and pooling layer can be of almost any shape.
The first, and admittedly most famous CNN model to this day is *LeNet*, proposed by LeCun [64] and displayed in Figure 3.6. *LeNet* was proposed to classify handwritten digits, but its efficacy quickly gave rise to a plethora of applications in other fields.

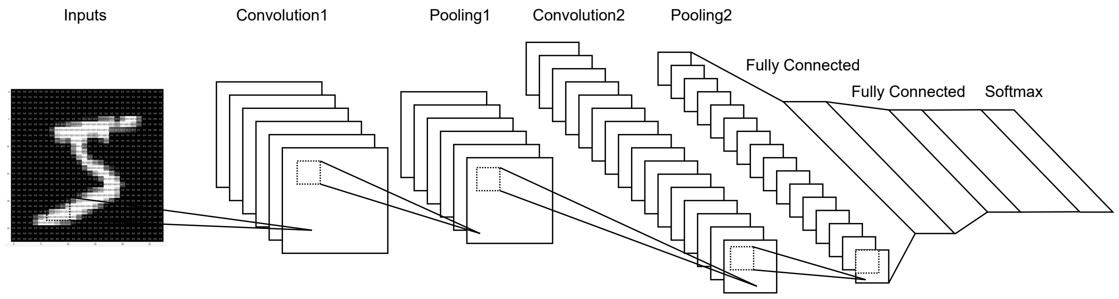


Figure 3.6: The LeNet-5 architecture.

The main idea behind this approach is that the first layer trains its kernels in order to extract low level features like edges, while the following convolutions extract higher level features such as connected components or patterns. Once the highest level features have been extracted, they are flattened and fed to a regular MLP, which trains its neuron activations in order to associate certain convolutional features with specific classes.

3.2 Quantum Machine Learning

As a natural consequence to the rapid advances in both the fields of classical machine learning and quantum computing, the last five years saw an astonishing increase in the research and development of techniques to merge these two disciplines into the framework of *Quantum Machine Learning*. It is enough to take a look at Google's *NGram Viewer* [65] for that same keyword, as shown in Figure 3.7, which displays the frequency of a keyword over an extremely large dataset of books and papers. It is important to take into consideration that the available data only accounts for the English language and is limited up to 2019.

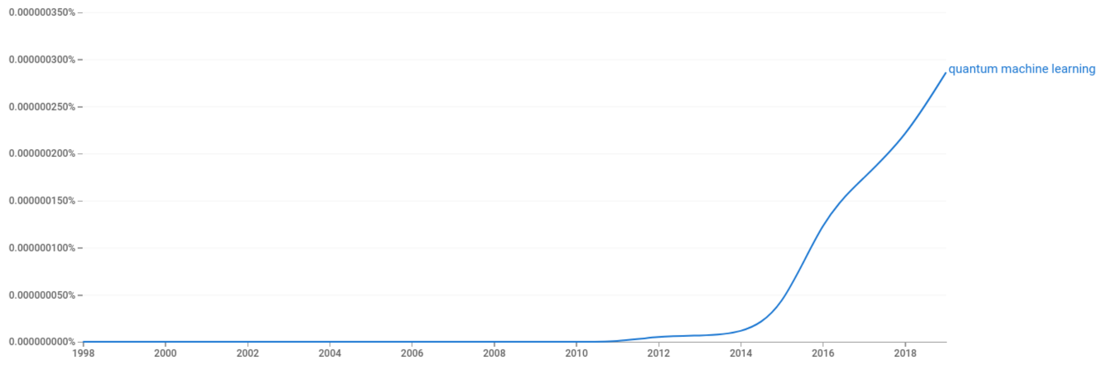


Figure 3.7: NGram frequency of the phrase *Quantum Machine Learning.*

QML is nowadays an extremely rich and branching area of research and as such can embody many different interpretations. As per the words of Wittek and Dunjko [66] in their *non review* of the field's status at the time of writing

> *'QML is not one settled and homogeneous field; partly, this is because machine learning itself is quite diverse.'*

It is however possible to separate QML into four main *corpora* of research [25], following the combinations of the words *classical* and *quantum* over tuples of the form *(data type - processing type)*, as depicted in Figure 3.8.
Classical-Classical (CC) refers to classical data processed with classical computing devices. In the context of QML, some techniques have been adapted from the quantum information domain to build new algorithms which are however completely classical, such as quantum-inspired networks or evolutionary algorithms [67, 68].
Quantum-Classical (QC) is a branch of research trying to extract information from quantum data through the usage of classical algorithms. It is particularly used for purposes such as exploratory research and data mining, paving the way for the discovery of new patterns or quantum algorithms. Another application for QC is the computation of the gradient of quantum states, necessary for the development

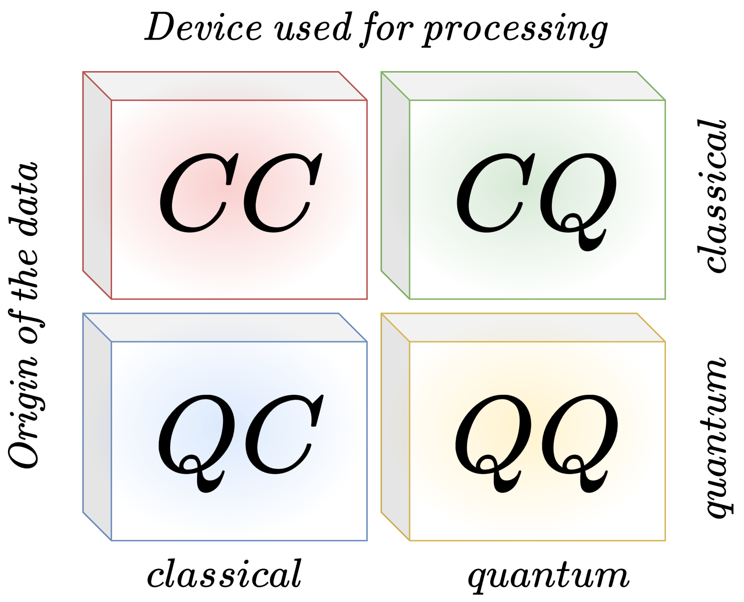


Figure 3.8: The four main branches of QML.

of hybrid quantum-classical networks such as VQEs.
Classical-Quantum (CQ) is the direct application of quantum processing to classical data. In order to do so, it is necessary to define quantum data representations of classical objects such as images, as discussed in the next subsection 3.2.1. These networks are either direct translations of their classical counterparts into the quantum domain, or new architectural paradigms altogether.
Quantum-Quantum (QQ) is, intuitively, the analysis of quantum-generated data through the use of quantum devices. This may be done either by executing quantum circuits, measuring their outputs and later encoding such data again into another quantum device to perform analyses, or by generating the data and performing the QML computations directly into a single run of a quantum circuit. The first approach cuts down on the time needed for computation, as the input state is only computed once, but suffers from the need for measuring and re-encoding. The second reverses advantages and disadvantages, as the data does not need to be transferred, possibly granting a speedup, but the initialization must be performed before each execution.
In the context of this thesis's work, the main focus will be on CQ architectures, as it is currently the most prolific branch of QML. The topic of quantum *advantage* will however not be investigated, but rather the one of fault *resilience*, as it will be discussed in Chapter 4.

3.2.1 Quantum data representations

In order to work with classical data onto quantum devices it is first necessary to embed the inputs into an Hilbert space through the aid of a *quantum feature map* [69, 70]. This general approach follows the same reasoning behind feature maps in SVMs, as seen previously in Subsection 3.1.2. In most of the cases this is done through parametrizable quantum gates. The following paragraphs will focus on some of the most commonly used representations for images and data points.

Basis Embedding

It is the conceptually simplest embedding possible and amounts to encoding each classical bit into a separate qubit, according to the following mapping, where the subscript c refers to classical. As such, the length of the bitstrings directly determines the necessary amount of qubits to embed a state.

$$0_c \xrightarrow{\Phi} |0\rangle\,, \quad 1_c \xrightarrow{\Phi} |1\rangle \tag{3.5}$$

Superposition is then exploited in order to account for all possible elements in a classical dataset $\mathcal{D}$. Given a quantum register of size N and a dataset size M, the whole embedding process will follow Equation 3.6, assigning to each element the same probability amplitude α. If $M \ll 2^N$, then the basis embedding will be sparse. Algorithms that work with this embedding, like Grover's search (1.5.2), operate on the probability associated to each state, trying to separate solution states with respect to the others and then performing multiple executions, revealing the correct output through statistics.

$$\mathcal{D} \xrightarrow{\Phi} |\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} \left|x^{(m)}\right\rangle \tag{3.6}$$

Amplitude Embedding

This embedding converts an N-dimensional datapoint x into the probability amplitude of a quantum state composed of $n = log_2(N)$ qubits, by associating each *i-th* element of x to the *i-th* computational basis state, as seen in Equation 3.7. This approach expands the previous encoding capabilities to real numbers and integers. Since the overall total sum of probabilities must be preserved, it is necessary to normalize the input vector. If the dataset size is M, the number of amplitudes to be encoded will be $N \times M$, as such the total number of qubits required in the quantum register will be $n \geq log_2(NM)$.

$$\mathcal{D} \xrightarrow{\Phi} |\mathcal{D}\rangle = \sum_{i=1}^{2^n} \alpha_i \, |i\rangle \tag{3.7}$$

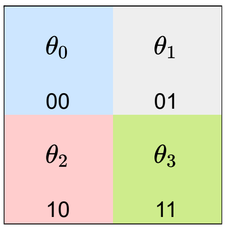
Angle Embedding

The following approach encodes an N-dimensional input feature vector x into the polar angles associated to each qubit of a quantum register [71]. The choice of using exponential and sinusoidal functions is not strictly linked to any property, as such this embedding may well be considered a variant of the amplitude embedding, among many others. Due to the fact that two input features are encoded at a time, this embedding is also defined *dense*.

$$x \xrightarrow{\Phi} \bigotimes_{i=1}^{\lceil N/2 \rceil} cos(\pi x_1)\,|0\rangle + e^{2\pi i x_2} sin(\pi x_1)\,|1\rangle \tag{3.8}$$

Flexible Representation of Quantum Images

One of the first encodings ever proposed for images in the quantum domain, FRQI uses two sets of quantum registers, one for representing the colour and the other for the pixel's position in the image [72]. The state formulation in expressed in Equation 3.9, along with an example in Figure 3.9. The θ_i angles encode in the probability amplitudes the colour associated to each pixel, whilst the $|i\rangle$ vector encodes the pixel's position following a custom addressing method, such as by lexicographical order or by block. The FRQI state is normalized, as such $\|\,|I(\theta)\rangle\,\| = 1$.



$$\begin{aligned} |I\rangle =& \frac{1}{2}[(cos\theta_0|0\rangle + sin\theta_0|1\rangle) \otimes |00\rangle \\ &+ (cos\theta_1|0\rangle + sin\theta_1|1\rangle) \otimes |01\rangle \\ &+ (cos\theta_2|0\rangle + sin\theta_2|1\rangle) \otimes |10\rangle \\ &+ (cos\theta_3|0\rangle + sin\theta_3|1\rangle) \otimes |11\rangle] \end{aligned}$$

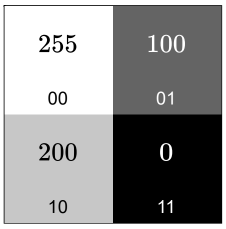
Figure 3.9: The FRQI encoding of a 2×2 image.

$$\begin{gathered} |I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (cos(\theta_i)\,|0\rangle + sin(\theta_i)\,|1\rangle) \otimes |i\rangle \\ \theta_i \in \left[0, \frac{\pi}{2}\right] \;, i = 0, 1, \ldots, 2^{2n} - 1 \end{gathered} \tag{3.9}$$

This embedding has been demonstrated to be extremely easy to achieve in current quantum devices, requiring up to a polynomial number of simple rotation gates, depending on the size of the image.

Novel Enhanced Quantum Representation of digital images

As an evolution of FRQI, NEQR improved on many of the critical aspects of the previous representation [73]. Given a 2^q grayscale range in an image, it is possible to associate through a function $f(Y, X)$ the value of a pixel in an image given its coordinates as per seen in Equation 3.10, along with an example in Figure 3.10.
The time and gate complexity required for preparing the state has a quadratic decrease with respect to FRQI, boasting higher compression ratio capabilities. Moreover, the retrieval of the image from the quantum state is accurate and not probabilistic as before, making it more reliable and requiring a lower number of executions.



$$
\begin{aligned}
|I\rangle =& \frac{1}{2}[|255_d\rangle \otimes |00\rangle + |100_d\rangle \otimes |01\rangle \\
&+ |200_d\rangle \otimes |10\rangle + |0_d\rangle \otimes |11\rangle]
\end{aligned}
$$

Figure 3.10: The NEQR encoding of a 2 × 2 image.

$$
\begin{aligned}
&f(Y, X) = C^0_{YX} C^1_{YX} \dots C^{q-1}_{YX},\ C^k_{YX} \in [0{,}1] \quad f(Y, X) \in [0, 2^q - 1] \\
&|I\rangle = \frac{1}{2^n} \sum_{Y=0}^{2^n-1} \sum_{X=0}^{2^n-1} |f(Y, X)\rangle\, |YX\rangle = \frac{1}{2^n} \sum_{Y=0}^{2^n-1} \sum_{X=0}^{2^n-1} \bigotimes_{i=0}^{q-1} \left|C^i_{YX}\right\rangle |YX\rangle
\end{aligned}
\tag{3.10}
$$

3.2.2 Gradient computation

Section 3.1 expressed the need to compute the gradient of a function in order to *descend* the fitness landscape associated to the optimization problem. Currently, it is possible to estimate the expectation of the output of a quantum observable by taking the average over the probabilistic results coming from measurements, as such, the gradient of a quantum computation can be formalized as the derivative of expectations [74].
Given an *ansatz* quantum routine consisting of a sequence of parametrized set sequence $U(\theta)$ gates repeated K times, where θ is a set of m real gate parameters, the expectation value of its measurements $\hat{B}$ can be defined as a variational quantum circuit. As such, this computation may be reformulated as a function that maps the gate parameters θ to an expectation.

$$f : \mathbb{R}^m \to \mathbb{R}^n \ , \quad f(\theta) := \langle \hat{B} \rangle = \langle 0| \, U^\dagger(\theta) \hat{B} U(\theta) \, |0\rangle \tag{3.11}$$

Equation 3.11 allows for the exact numerical computation of the gradient on a simulator, however applying it onto a real quantum device would only yield a *estimate* of $f(\theta)$. By computing the partial derivatives of $f(\theta)$ with respect to all the θ gate parameters, it is possible to *assemble* the gradient ∇f.
One of the approaches that can be used to do so is *numerical differentiation*, which performs an approximation through blackbox evaluations of g according to Equation 3.12, by exploiting the properties of the *parameter shift rule*.

$$\nabla g(x) \approx [g(x + \Delta x/2) - g(x - \Delta x/2)]/\Delta x \tag{3.12}$$

Other approaches can be adopted, such as *automatic* differentiation an *symbolic* differentiation, however they will not be further analyzed.

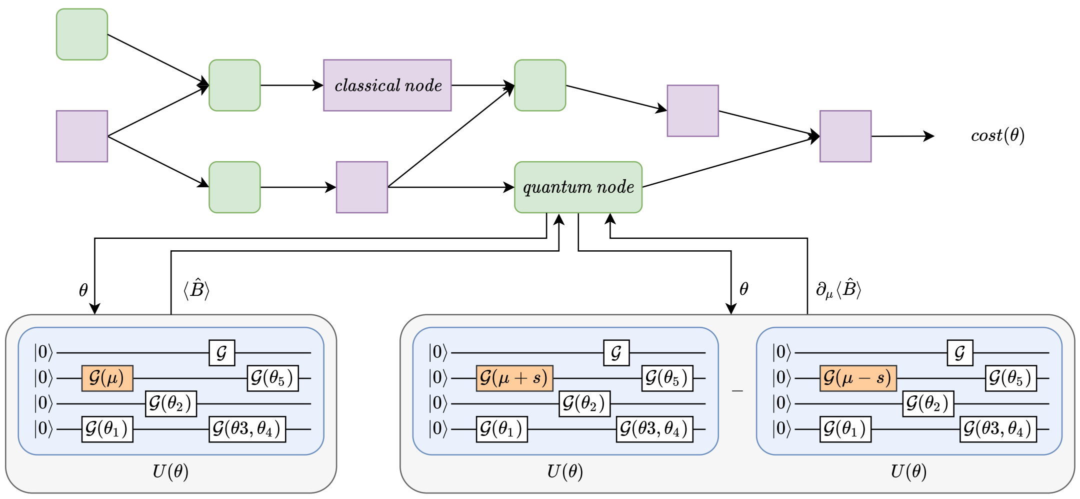


Figure 3.11: Parameter shift rule and numerical differentiation in an hybrid quantum-classical framework.

Figure 3.11 shows how the internal processing of each quantum node in an hybrid network, where $\mathcal{G}$ is a gate generated by an Hermitian operator with two unique eigenvalues of the form

$$\mathcal{G}(\mu) = e^{-i\mu G} \quad \xrightarrow{\frac{\partial}{\partial \mu}} \quad \partial_\mu \mathcal{G} = -iGe^{-i\mu G} \tag{3.13}$$

Once all partial derivatives have been computed, it's just a matter of computing the overall gradient of the circuit and from there on perform classical gradient descent operations with the aid of a cost function on a classical coprocessor.

3.2.3 Quantum Neural Networks

A Quantum Neural Networks is a broad term referring to the direct attempt to convert the techniques and models which make up modern classical ML into the *quantum realm*. For this very reason, the term *Quantum Neural Network* has been used more or less appropriately across the literature that spans over the last ten years to define any learning model which features a quantum device, varying from actual one-to-one translations of classical models to completely new and unforeseen architectures.

The classical ML neuron is an object capable of processing an input and transmitting an output, encoded in a binary fashion as *active* or *resting*, which could be translated easily to the basis states $|0\rangle$ and $|1\rangle$ of a qubit. This would allow these learning models to exploit quantum features like superposition and entanglement, possibly providing speedups or new processing approaches in the big data field. A simple model for the quantum neuron (*quron* [75]) can be obtained through amplitude encoding (Subsection 43).

$$|quron\rangle = rest\,|0\rangle + active\,|1\rangle \tag{3.14}$$

The main classical to quantum translation issue is that, by definition, quantum computing systems have been theorized to work with unitary and linear operators only. As such, a new interpretation of the concept of non-linearity linked to activation functions needs to be investigated, for example through the usage of a universal set of gates made of Gaussian and non-Gaussian elements [76].

However, as time goes on, the field is diverging more and more from the *quron* model, merging together under the same name other approaches which do not share any characteristic with classical models, like hybrid computation models and variational circuits applied to QML.

3.2.4 Variational Quantum Eigensolver

A Variational Quantum Eigensolver is an hybrid learning paradigm composed of a parametrized quantum circuit that can encode the state represented by an Hamiltonian and a classical optimizer. The role of the quantum circuit is to approximate through the Ritz method the lowest energy state associated to the Hamiltonian of the system without needing to explore the whole space of solutions. The circuit is represented as $U(\theta)$, where θ is a vector of features used as parameters in the internal gates which encode the input state. The circuit is then queried in order to estimate the energy eigenstate associated to that quantum system, descending the fitness landscape thanks to a cost function which will allow to reach the lowest energy Hamiltonian.

Given an observable H, a trial wave function $|\psi\rangle$ is defined, which will be used to

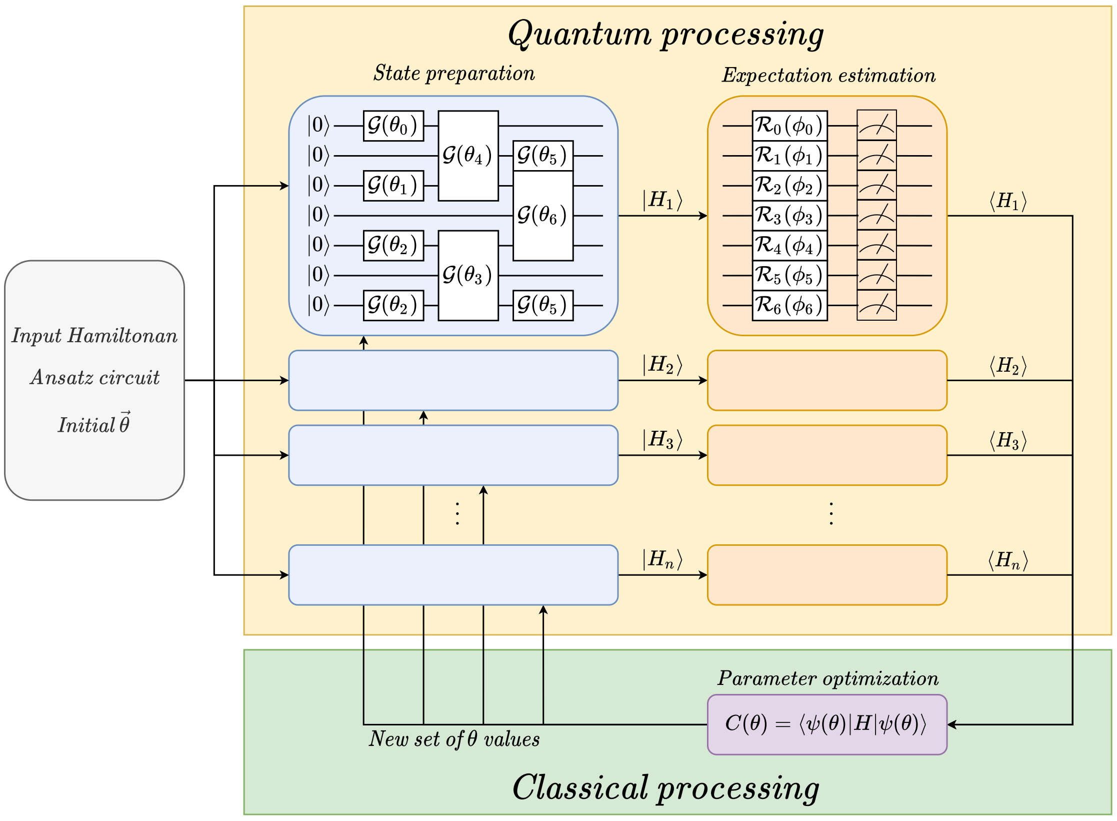


Figure 3.12: The architecture of a VQE.

compute an estimate of the expectation value $\hat{H}$ of the quantum state. Equation 3.15 shows that the $|\psi\rangle$ approximation can reach the ground energy state.

$$E_0 \leq \frac{\langle\psi|\, H\,|\psi\rangle}{\langle\psi|\psi\rangle} \tag{3.15}$$

Such $|\psi\rangle$ function must respect some boundary conditions, however its exact form is not know *a priori*. Consequently, it is represented in the quantum circuit by an ansatz architecture and optimized through its parameters, the *state preparation* process in Figure 3.12. At last, the circuit is measured according to different bases and an estimate of the expectation is produced. The quantum step is generally performed on multiple devices at the same time to further parallelize the computation. The optimization process, carried out onto a classical coprocessor, tries to minimize the cost function $C(\theta)$ with an iterative approach as seen in Figure 3.12, up until an estimate close enough to the ground state is reached.

$$C(\theta) = \langle\psi(\theta)|\, H\, |\psi(\theta)\rangle \tag{3.16}$$

The quantum device can encode H directly and compute its expectation, however this is not feasible for large Hamiltonians due to the increasingly higher number of measurements required. However, it is possible to define H as a weighted sum of local operators, such that multiple local and partial energy expectations can be computed separately and later merged into an estimate of the expectation of H.
The VQEs are mainly used in exploratory chemistry research and can be efficiently executed on current NISQ devices.

3.2.5 Variational Quantum Classifier

The VQEs can be adapted and expanded to build hybrid binary classifiers. Given a measurement operator, the expectation value $\langle O\rangle$ of an observable O can be interpreted as a classical output, since it represents the probability associated to measuring a quantum state. In the Equations 3.17 the chosen measurement operator is the Pauli σ_z, which determines the probability for a quantum state to be in $|1\rangle$.

$$\begin{aligned} f(x;\theta) &= \langle\psi(x:\theta)|\, \sigma_z\, |\psi(x:\theta)\rangle \\ C(\theta) &= \sum_{m=1}^{M} (y - \langle\psi(x^m:\theta)|\, \sigma_z^j\, |\psi(x^m:\theta)\rangle)^2 \end{aligned} \tag{3.17}$$

The cost function $C(\theta)$ is modified accordingly, by substituting $f(x:\theta)$ as the predicted output $\hat{y}$ in the MSE function presented in Subsection 3.1.1. As such, the VQC cost is a function of the expectation and not the expectation itself as previously seen in the VQE case.

3.2.6 Quantum Support Vector Machines

Stemming from their standard counterparts, quantum SVMs aim at performing the binary separation of a dataset through an hyperplane with a consistent speedup with respect to classical ones.
An SVM can be interpreted as a quadratic programming problem solvable in a time proportional to $O(log(\epsilon^{-1})poly(N,M))$, whilst a quantum SVM approach can be carried out in $O(log(NM))$, where N is the dimensionality of the dataset, M is the number of samples in the training partition and ϵ is the classification accuracy [77]. This *quantum speedup* is achieved thanks to a reformulation of the SVM as an approximate least-squares problem and by successively applying the matrix inversion algorithm on the embedded quantum state. Lastly, matrix exponentiation

and principal component analysis are performed to extrapolate one of the lower order approximations of the final kernel matrix. The main limitations of the hereby proposed approach is that it requires efficient quantum RAMs with a number of access operations on the order of $O(log(MN))$ and that the data is encoded as a coherent superposition, something which is not always attainable when working with classical datasets.
Another approach would be to use parametrized variational quantum circuits to either generate the separating hyperplane in the feature space or to estimate the kernel function in the feature space and use it in a conventional SVM [70]. The main caveat required to demonstrate quantum advantage is to adopt feature maps which are classically hard to compute or intractable, like the *ZZFeature map* in Figure 3.13. The P gates are parametrized *U1* gates: the encoding gates are $P1 = U1(2\phi(x[n]))$, while the rotation gates are $P2 = U1(2\phi(x[n], x[n+1]))$, where n is the qubit index.

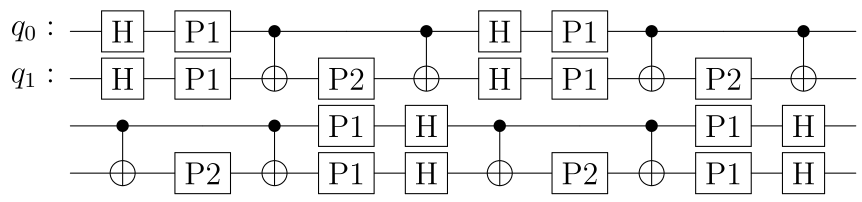


Figure 3.13: The second-order Pauli-Z evolution circuit with two repetitions.

3.2.7 Quanvolutional Neural Networks

Given their astounding performance on most image based ML tasks, CNNs saw a constant evolution over the last twenty years, maintaining their status as state of the art architectures. This success is due to the usage of correlated convolution filters, which abstract features from the inputs, providing probabilistic results on output. As such, some of the recent QML research tried to adapt this approach on quantum devices, which provide themselves probabilistic results, whilst also allowing for massive speedups thanks to the parallelization capabilities of the quantum computation paradigm [78], of which a simple hybrid architecture example is proposed in Figure 3.14.
The quantum convolutional layer, shortly also called *quanvolutional* layer or *qLayer*, encodes a convolution kernel in the structure of a BQP-type *ansatz* circuit and applies it to local subsections of an input, producing an output of higher level features. The layer's application can be then repeated multiple times, as in the classical case, to heighten the abstraction of the features. The locality in the application of the layer derives from the fact that each single classical matrix

multiplication between the kernel and a subgrid of the input tensor is converted to the execution of a small and shallow parametrized quantum circuit, avoiding the issues associated with noise propagation and decoherence of current NISQ devices, whilst still providing a speedup. The output probability distributions are encoded as expectation values for each qubit in the circuit, repeated with different ansatzes for each output channel of the same spatial position in the image.

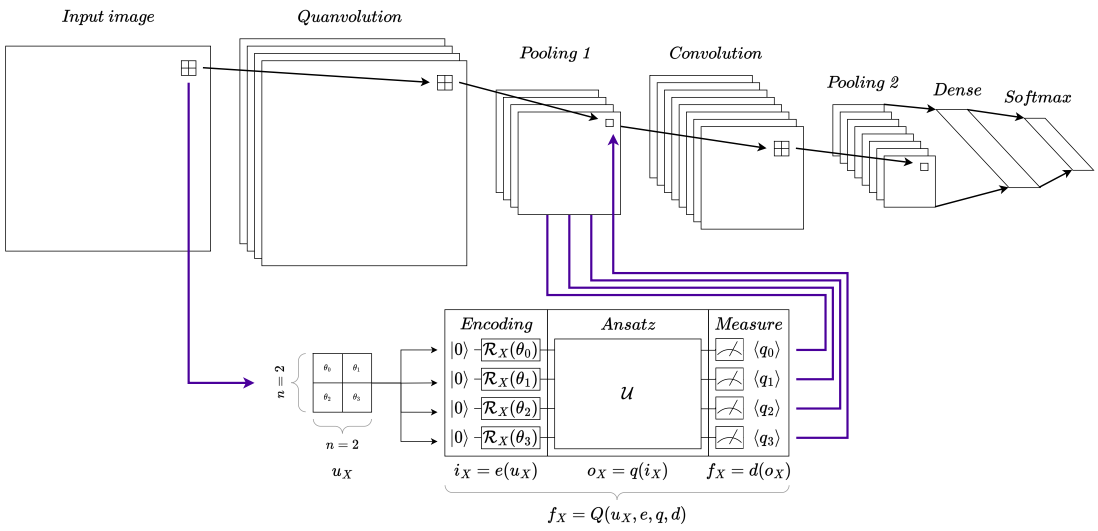


Figure 3.14: The architecture of an hybrid QNN and detail on the quanvolutional layer.

The quanvolution filter is first composed by an encoding portion, which starting from the coherent state $|0\rangle^{\otimes n}$, embeds the input subgrid u_X of size $n \times n$ into the quantum circuit through the usage of a function $i_X = e(u_X)$. One possible approach is amplitude embedding through the use of parametrized rotations around the X axis of the Bloch sphere: in the case of grayscale images, the inputs are remapped linearly from [0,255] to $[0, \pi]$.
The central portion contains a random circuit which performs the quantum dual of the classical element by element multiplication among kernel and input matrices as the application of a function $o_X = q(i_X) = q(e(u_X))$.
At last, the outputs are decoded through measurement, and their expectation is transferred in the output tensor in the form of a scalar ranging from [−1,1]. The whole quantum processing step can thus be described as the application of nested functions as seen in Equation 3.18.

$$f_X = Q(u_X, e, q, d) = d(q(e(u_X))) \tag{3.18}$$

An example of the qLayer's outputs have been presented in Figure 3.15. It is noticeable how it corresponds a padded convolution with kernel size 2×2 and a number of filters equal to four, thus requiring four different random circuit *kernels* [79]. It is notable how the input's dimensions are reduced from 28×28 to 14×14, accounting also for the pooling layer.

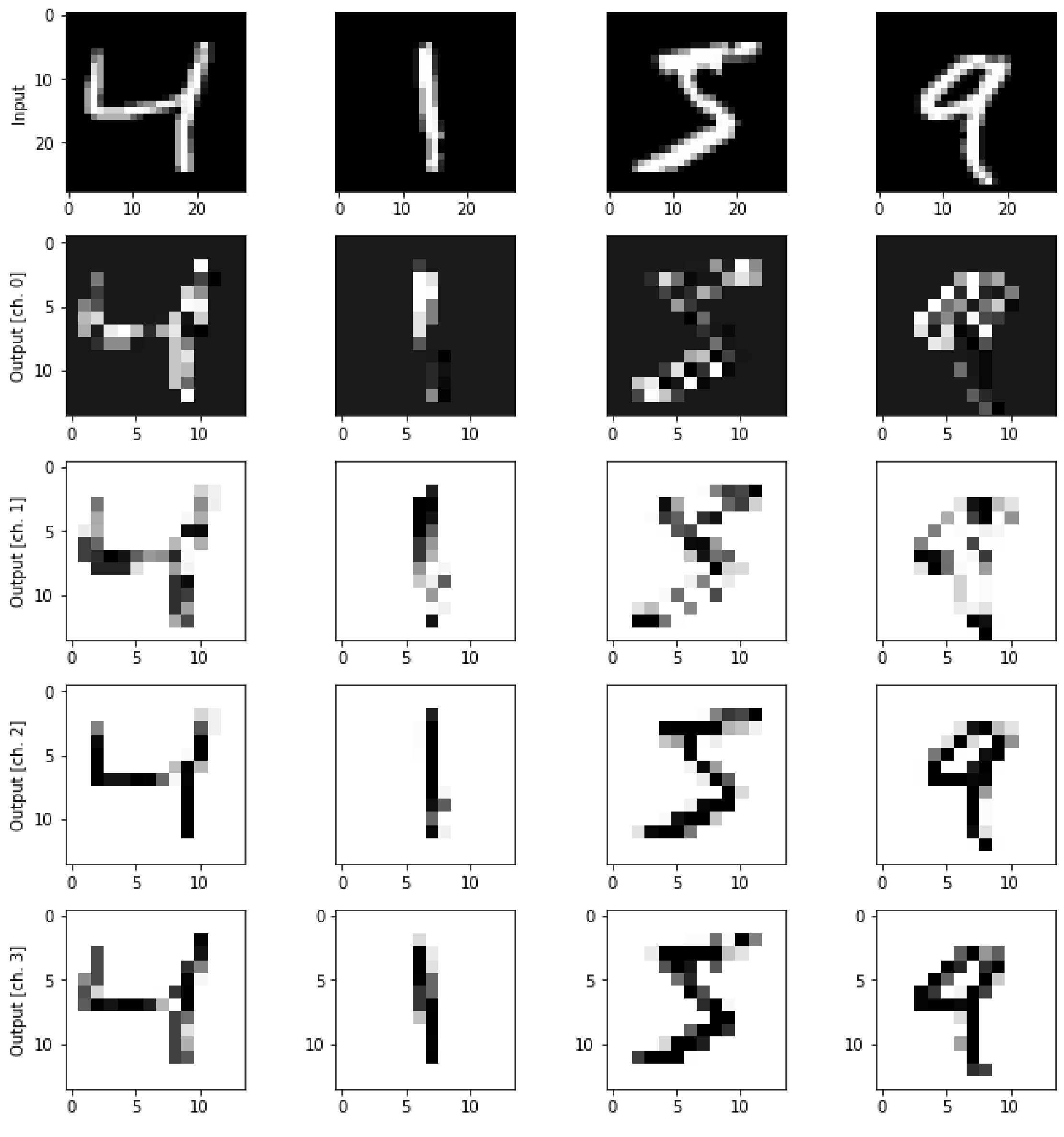


Figure 3.15: Output example of the quanvolutional layer on the MNIST dataset. Images

Chapter 4

QML Reliability Evaluation

4.1 Motivation

Over the last three years, there has been an astounding rush to interface the world of quantum computing and machine learning, under the definition of quantum machine learning. Given the novelty of this emergent field, it is of foremost importance to tackle the reliability of the basic components which make up QML, in order to better understand and predict their behaviours at a lower level.
QuFI can provide this kind of insight and as such it has been used to perform simulations on a few QML related subroutine circuits. However, another point of research was to study the overall effect of injections onto the accuracy of the neural networks which use these quantum layers. As such, QuFI has been expanded in order to allow for the computation of accuracy metrics on some specific architectures.
This work aims to act as a first tentative step into QML reliability evaluation and by no means must be considered the only possible approach. Rather, it tries to pave the way for further study on the topic to deepen our knowledge in this field.

4.2 Approaches

At first, for each subroutine circuit, a QVF study has been conducted. This allowed for a better understanding of the most critical injection phase shift values and discovering any possible hidden relationships among angle rotations and circuit output. Then, a subset of the most critical fault points and fault angles has been selected, in order to provide more detailed information regarding how a specific fault can affect the inference process of a quantum neural network by computing the accuracy metrics and prediction confidence deltas for each image in the test dataset. This allowed for finding patterns in the circuit's reliability to faults and for the selection of specific images in the dataset to further investigate the effect of injection on the numerical output of the networks. It is important to note that the main focus of the study is not to detect how the accuracy changes with respect to the introduced faults, but rather to compare the faulty performance with the golden one. As such, the outputs used for comparison may well be wrong class predictions.

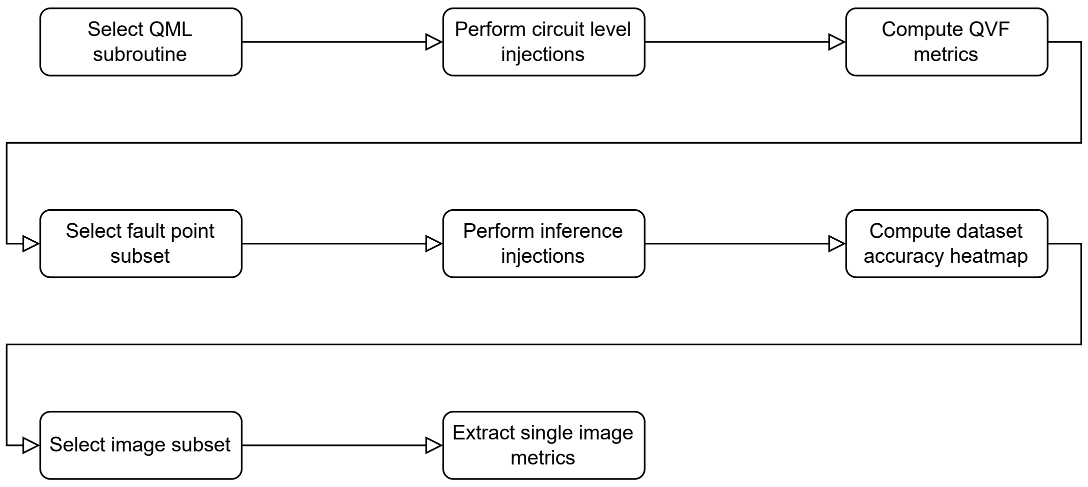

Select QML subroutine
Perform circuit level injections
Compute QVF metrics
Select fault point subset
Perform inference injections
Compute dataset accuracy heatmap
Select image subset
Extract single image metrics

Figure 4.1: The process diagram of the proposed case study approach.

4.3 Experiments

Two sets of experiments have been conducted, focusing on a QSVM and on a the quanvolutional layer of a QNN. The first case has been investigated up to the QVF metrics step of Figure 4.1, whilst the second case has been investigated more thoroughly.

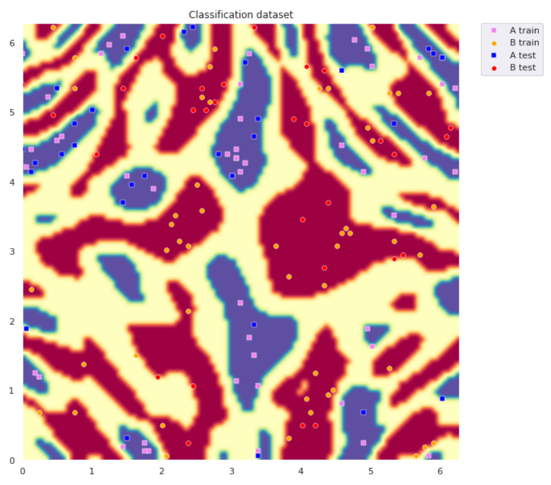
4.3.1 QSVM

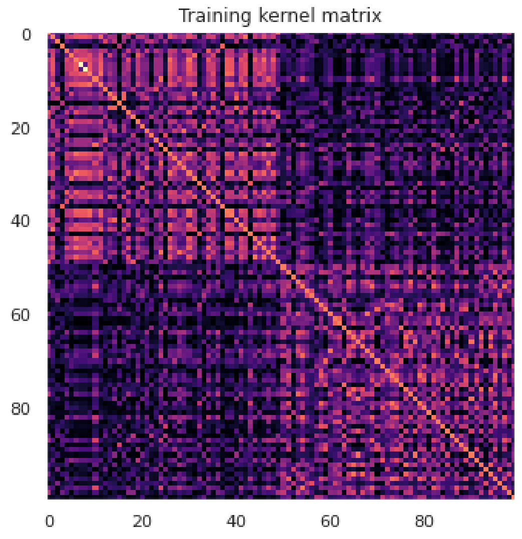
The QSVM structure is that of a second-order Pauli-Z evolution circuit, repeated twice, whose architecture has been previously presented in Figure 3.13.
The circuit has been built by invoking *Qiskit*'s *ZZFeatureMap* class with a linear entanglement structure and a feature size of two. This architecture can both be used for classification and clustering problems. An ad-hoc dataset has been built, where its features have been remapped purposefully to become hard to compute on classical devices. The performance in both problems easily saturates to values close to 1.0, given the small size of the dataset and the advantageous characteristics of the feature vectors in the quantum domain. In [70], this approach is justified by the fact that there is no advantage to be gained in using QSVMs on problems

where the inner product with the kernel function is already simple to compute in the classical domain.

Classification performance

In Figure 4.2 we can notice the two kernel matrices for the classification problem. They represent the inner product among different elements. The dataset used featured 50 training datapoints and 30 testing datapoints.
In the case of the training partition, an evident diagonal line is present, as the inner product of an element with itself is always going to be equal to 1. The testing partition instead shows the correlation between the training weights and the testing dataset: the top left and bottom right portions of the kernel matrix show a correlation between the two datasets and justify the high classification accuracy of this QSVM. The first halves of both partitions belong in fact to class A, whilst the second halves belong to class B. This behaviour is noticeable, albeit to a lesser extent, also in the training kernel matrix.





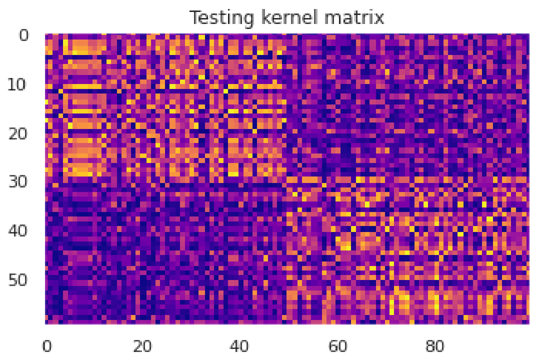


Figure 4.2: QSVM classification dataset and kernel matrices.
Top left: the dataset used for training and testing, with visualization of the mapping function used for generation. Top right: the kernel matrix containing the result of the inner product between all elements of the dataset. Bottom right: the kernel matrix representing the inner products between the training weights and the testing datapoints.

Clustering performance

Similarly, in the clustering dual of the problem depicted in Figure 4.3, the dataset is split in two halves, according to their class, with a total number of 50 datapoints. As such, the kernel matrix reflects a high correlation among elements in the top left and bottom right inner products. The highlighted diagonal in the figure is once again caused by the computation of the inner product of an element with itself, yileding the maximum possible value.

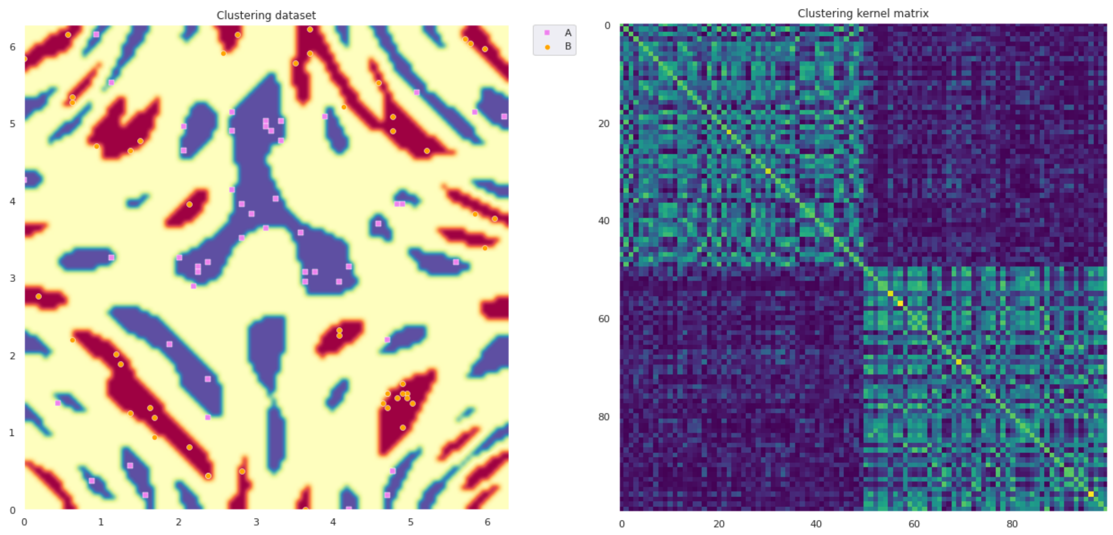
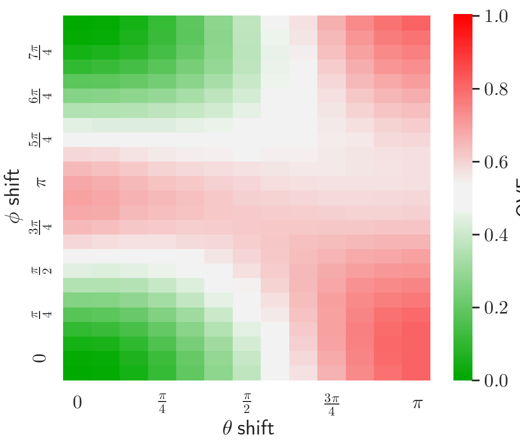


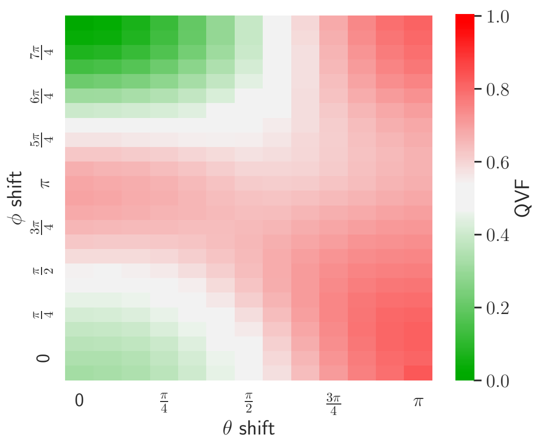
Figure 4.3: QSVM clustering dataset and kernel matrix.
Left: the dataset to be clustered, alongside the decision boundaries of the mapping function used to generate it. Right: the kernel matrix containing the values of the inner products among all elements in the dataset.

QVF study

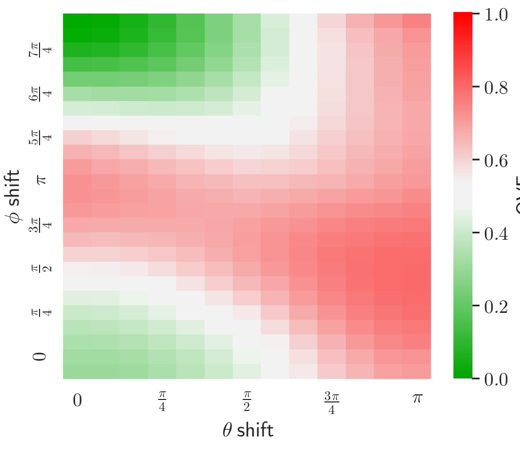
The role of the quantum circuit is to compute the inner product between each element of the dataset. One random input has been chosen as a constant in order to parametrize the *ZZFeatureMap* circuit during all the injections needed to extract the QVF metrics. Following [49], the value range for the injections varied between $[0, \pi]$ for ϕ and between $[0, 2\pi]$ for θ, both with steps of $\frac{\pi}{12}$.
The results of the injections are presented in Figure 4.4. Starting from the top left, it is noticeable how the circuit is quite resilient to single faults with amplitude combinations of ϕ ranging between $[-\frac{\pi}{2}, \frac{\pi}{2}]$, as the polar angle corresponds to the rotation quantity around the Z-axis and thus it is periodic, given that $\mathcal{R}_Z(\frac{6\pi}{4}) = \mathcal{R}_Z(-\frac{\pi}{2})$, and of θ ranging between $[0, \frac{\pi}{2}]$. These two ranges of reliability can be



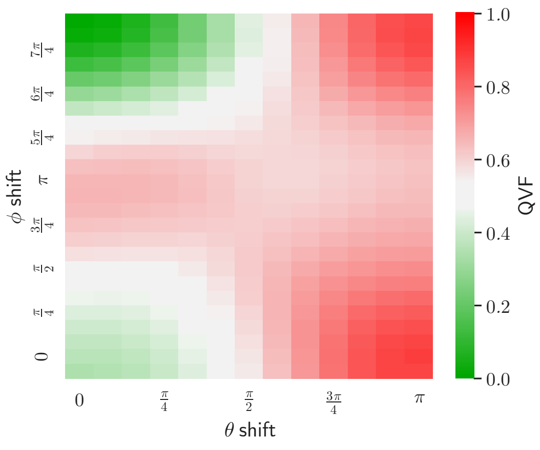
Total QVF_s.



Total QVF_d.



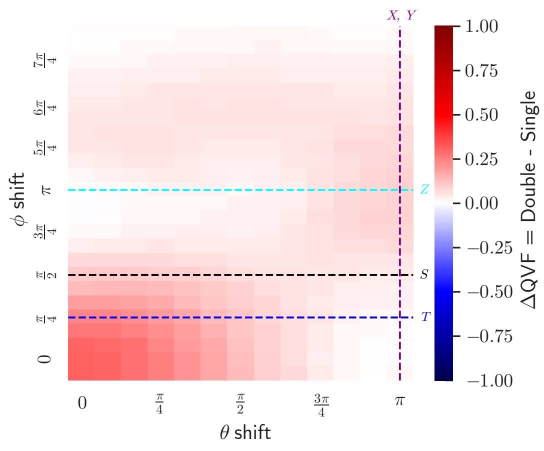
QVF_d on qubit 0.



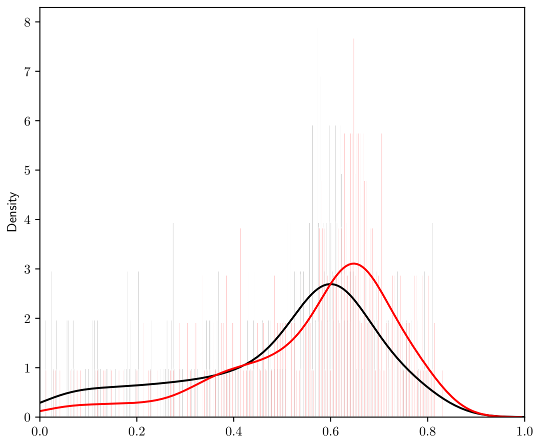
QVF_d on qubit 1.

Figure 4.4: QSVM: Single and double fault injection heatmaps.

visualized on the Bloch sphere as a spherical dome whose top is centered on the statevector encoding the quantum state before undergoing the injection.
The double fault behaviour, as seen in the top right heatmap, shows an increase in the overall QVF metric's values, especially in faults where θ ranges between $[\frac{\pi}{2}, \pi]$, in accordance with the fact that higher single fault amplitudes will statisctically result in a wider number of secondary faults with amplitudes up to the values of the primary fault and as such potentially more disruptive.
Single qubit double fault heatmaps, in the lower portion of the figure, show that, generally, the *ZZFeatureMap* circuit is more resilient to faults onto qubit 0, boasting an overall lower QVF, whilst a fault on qubit 1 is slightly more likely to cause a detrimental effect.



ΔQVF=QVF$_d$−QVF$_s$.



Single (black) and double (red) fault histograms.

Figure 4.5: QSVM: Delta heatmap and histogram.

Figure 4.5 shows the delta between the double and the single fault injection heatmaps and the histogram distribution of the QVF values. In the delta heatmap on the left, red tiles represent a worsening in the performance with the double fault when compared to the single fault. The largest QVF loss takes place in the lower left of the heatmap, for faults ranging in the lower values of θ and ϕ. The histogram on the right depicts in black the single faults and in red the double faults, over which their respective kernel density estimates have been plotted for ease of interpretation. These estimated curves show that the double fault increases the overall QVF of the circuit reducing by a significant margin the entries with values ranging between 0 and 0.4, as it can be evicted by the shifted and squeezed red peak when compared to the black one. However, despite the shift towards the right, no value greater than 0.85 is ever achieved. Detailed statistics on the two histograms are available in Table 4.1.

Metric / Fault type	Mean	Stnd. dev
Single fault	0.50	0.20
Double fault	0.57	0.17

Table 4.1: QSVM: histograms statistics.

4.3.2 QNN

Much like its classical counterpart, a QNN can refer to any network architecture containing quanvolutional layers. The following analysis will take into consideration the model previously proposed in Chapter 3, Figure 3.14.
The quanvolutional layer implementation follows the specifications presented by Henderson *et al.*[78]. It is composed of a four qubit circuit, subdivided into three sections: an amplitude embedding section used to encode 2×2 grayscale images through a parametrized *Pauli-Y* rotation with a linear mapping [0,255] $\rightarrow [0, \pi]$, followed by an ansatz which computes the actual convolution, at last followed by a measure layer which extracts the expectation value of each qubit, mapped onto a different output channel of the final image and implicitly performing a max pooling operation.

Classification performance

In order to test the network's baseline performance, it has been trained on a classification problem over a subset of the MNIST handwritten digits dataset, with 50 training images and 30 validation images, and compared with a CNN featuring a similar architecture where the only difference was a classical convolutional layer with max pooling substituting the *qLayer*. Figure 4.6 shows that the two architectures attain similar performances, both during training and validation. The only noticeable difference is a faster convergence time on the side of the qLayer. This similarity is easily justifiable, both because the two architectures share most of the layers and because the quanvolutional and convolutional steps are performing the same tensor transformation, with a margin of difference dictated by numerical accuracy and random initialization parameters. In any case, the training times for both networks vary significantly, with the QNN requiring ~ 30 seconds instead of the ~ 6 seconds of the CNN, due to the fact that the qLayer requires the simulation of a quantum circuit, which is inherently slower than the tensorflow-based convolution.

QVF study

The quantum circuit is responsible for computing a single convolution *block* between the kernel, embodied by the structure of the quantum circuit itself shown in detail in Figure 4.7, and a subgrid of the input image. In order to run the QVF study, a fixed 2×2 image has been used, where the topmost right and leftmost pixels represent a white pixel with value 255 and the other two represent a black one with value 0. This image corresponds to an ecoding where qubits 0 and 3 are encoded in state $|0\rangle$, whilst qubits 1 and 2 are encoded in state $|1\rangle$. Moreover, to fit the analysis requirements of QVF, the final measurement section of the circuit has been

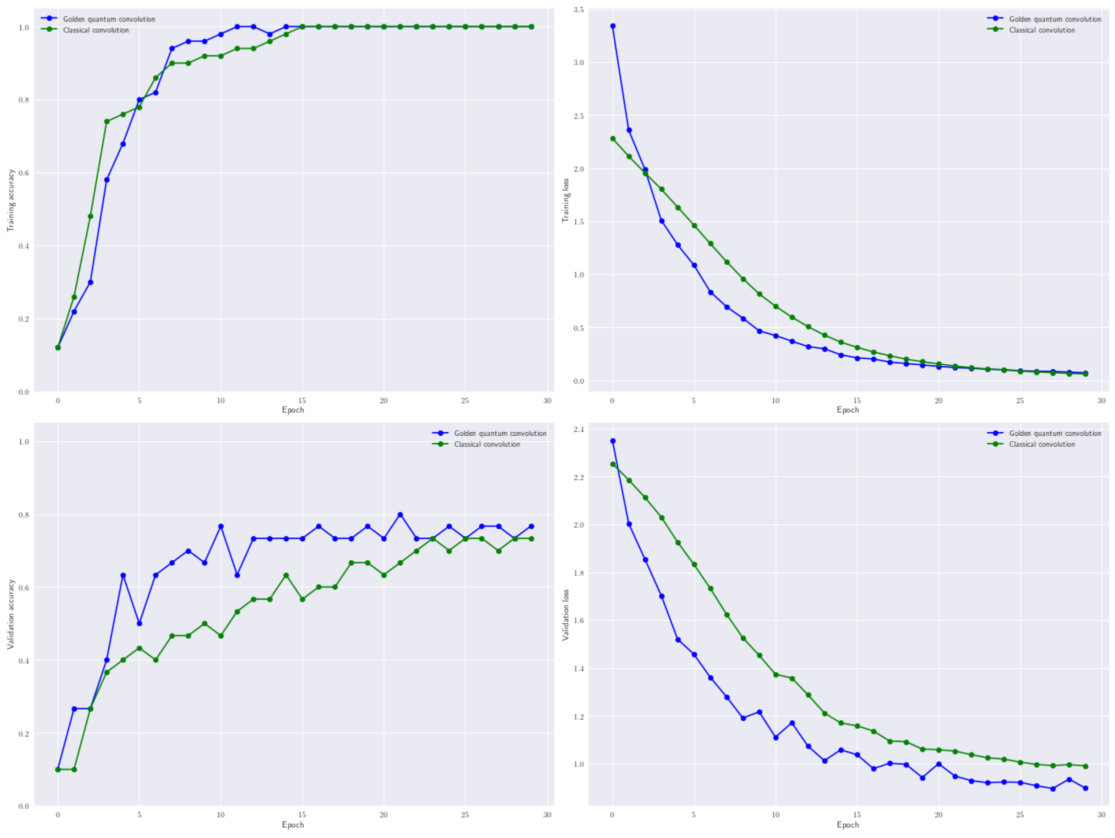


Figure 4.6: Training comparison between a QNN and a CNN.

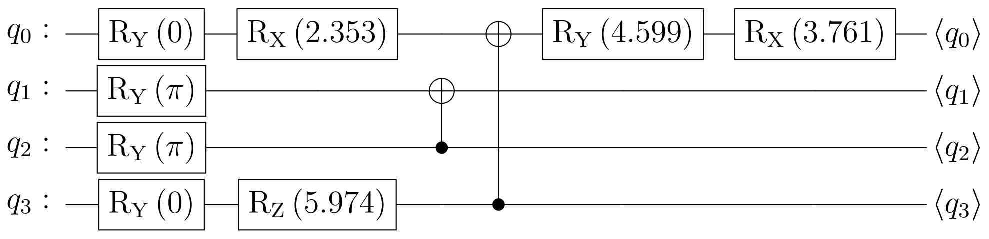
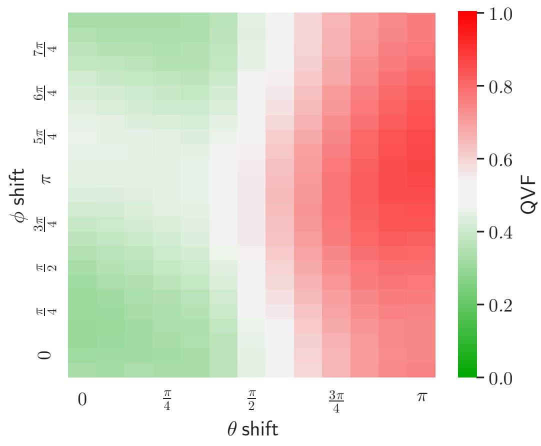
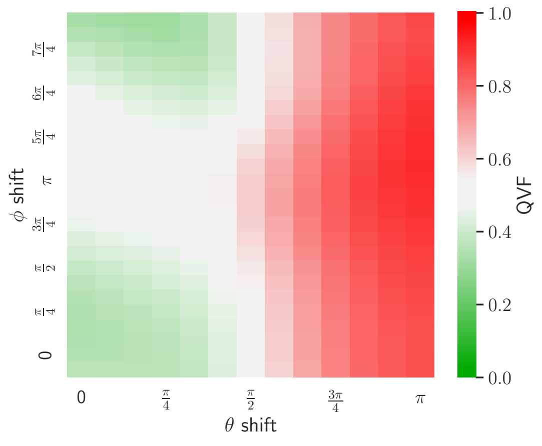


Figure 4.7: Quanvolutional layer circuit.

switched from the *Pauli-Z* expectancy operator to the distribution of probabilities for each measured output state.
The results are presented in Figure 4.8. In the left single fault heatmap, qLayer shows a relatively low variance with respect to the polar angle ϕ, albeit a small QVF rise between $\frac{3\pi}{4}$ and $\frac{5\pi}{4}$, whilst it seems to be much more affected by the azimuthal faults on θ values strictly greater than $\frac{\pi}{2}$. The double fault graph follows the same description, with heightened values in the right part of the heatmap and



Total QVF_s.



Total QVF_d.

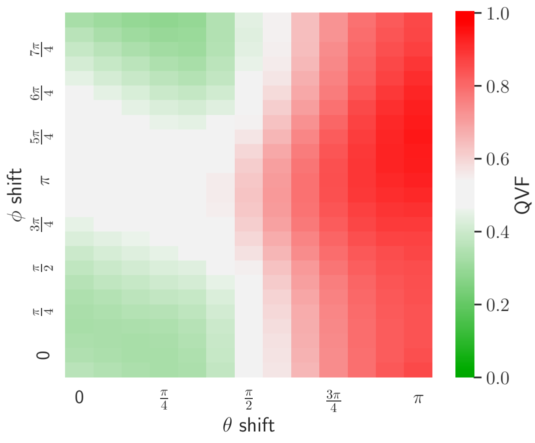
Figure 4.8: QNN: Circuit level QVF heatmaps.

a more noticeable increase in the QVF in the previously mentioned $[\frac{3\pi}{4}, \frac{5\pi}{4}]$ range, other than the overall shift towards the left of the white vertical uncertainty bar with values around 0.5.

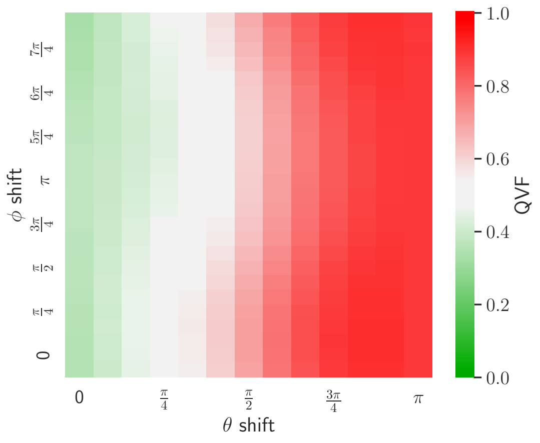
Specific qubit double fault heatmaps are displayed in Figure 4.9 and each of them shows a different behaviour. Qubit 0 reflects more closely the general circuit behaviour undergoing a double fault, with a clear dependence on the fault angle θ and a noticeable increase in the QVF in the ϕ range $[\frac{3\pi}{4}, \frac{5\pi}{4}]$. Qubit 1 shows on the left a very narrow vertical band of values higher than 0.4, whilst the uncertainty vertical bar is shifted to $\theta = \frac{\pi}{4}$: all other values are greater than 0.6. Qubit 2 has a similar behaviour to qubit 1, albeit a shifted vertical white bar at $\theta = \frac{\pi}{2}$. Qubit 3 is instead the more diverse of the four, with an horizontal high QVF bar spanning the whole heatmap with $\phi \in [\frac{3\pi}{4}, \frac{6\pi}{4}]$ with a slight downward direction from left to right.

This more dominant sensitivity to the fault of the azimuthal angle is a clear consequence of the amplitude embedding used in the qLayer. In fact, rotations about the polar angle are invariant with respect to the value encoded in the qubits, albeit such rotations may modify the effect of successive gates.

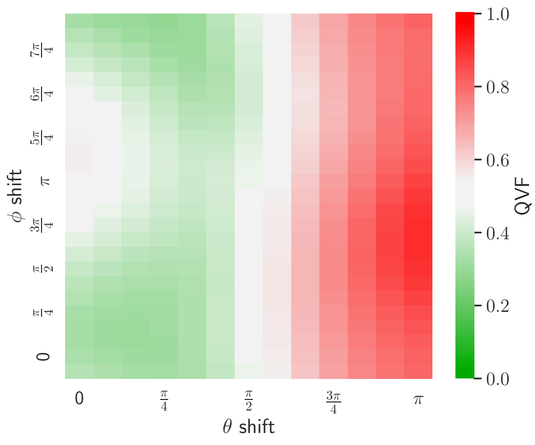
The impact of the double fault is summarized by both the delta QVF heatmap and the kernel distribution estimates in the histogram of the QVF values for single and double faults, shown in Figure 4.10. A general slight increase of the QVF values is noticeable by both the delta heatmap being almost uniformly made of values ranging between 0.1 and 0.15, reflected into a shift in the peaks of the KDEs towards the right. Numerically, it amounts to a 5% shift in the mean between the two distributions, as it can be seen in Table 4.2. The fact that the histograms show



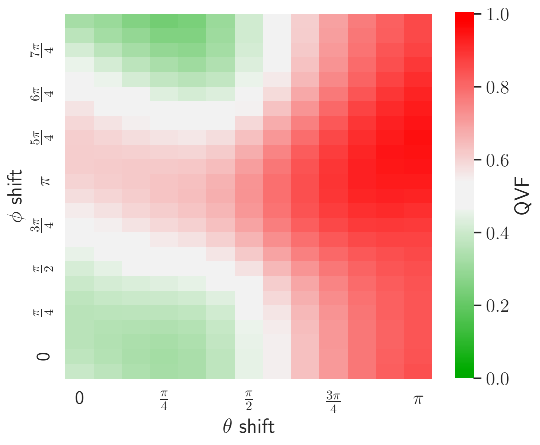
QVF_d on qubit 0.



QVF_d on qubit 1.



QVF_d on qubit 2.



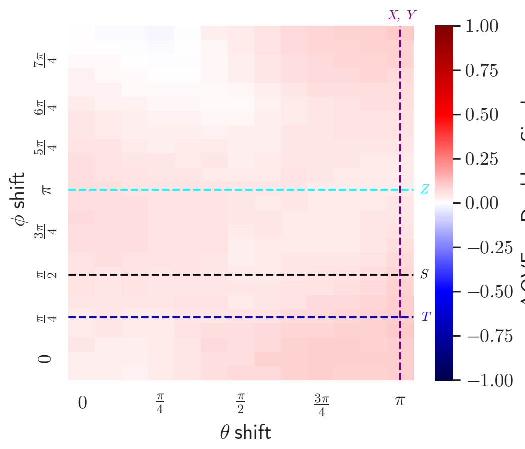
QVF_d on qubit 3.

Figure 4.9: QNN: Qubit level double fault QVF heatmaps.

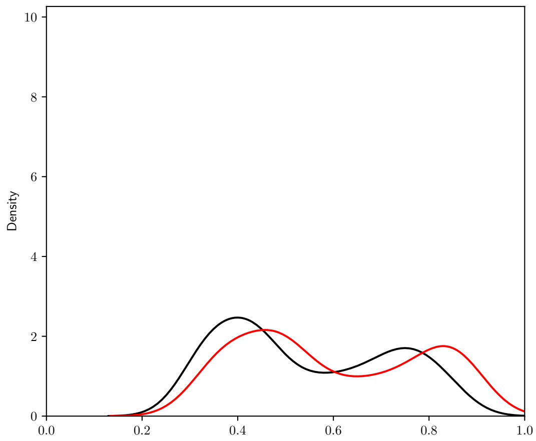
a bimodal distribution is a direct measure of the sensitivity of the quanvolutional layer to faults, as it either resists the negative effects very well or flat out performs extremely poorly, whilst never settling into a middle ground behaviour.

Metric Fault type	Mean	Stnd. dev
Single fault	0.55	0.17
Double fault	0.60	0.19

Table 4.2: QNN: histograms statistics.



ΔQVF=QVF$_d$−QVF$_s$.



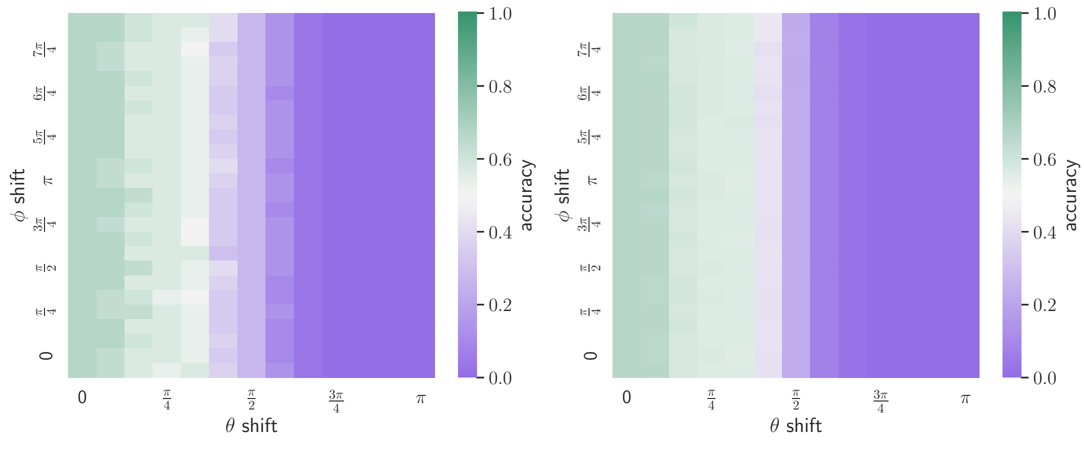
Single (black) and double (red) fault histograms.

Figure 4.10: QNN: Delta heatmap and histogram.

Accuracy study

The following step was to study the impact of faults during inference from the accuracy point of view. As such, the trained network has been set up in a similar fashion to that of the previous QVF study, ranging over all the combinations of angles for $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. However, this time, a specific fault position has been chosen, rather than testing all possible combinations gathered from the coupling map of the transpiled circuit. This is due to a time and load requirement on the machine used to perform the computation. In fact, it has been estimated that attempting to extract the accuracy metrics over all fault positions on a computing node with 28 vitual cores at 2.22 GHz would have required upwards of 15 days of continuous computation. As such, the range variation has been preserved, but the position variation has not. The specific fault configuration is depicted as an orange operator in Figure 4.12.

The accuracy results over the whole testing dataset has been measured for each of those fault points and the average over all the fault points has been presented as the comparison metric for the following results. Figure 4.11 shows the heatmaps of both single and double fault injections, respectively on the left and on the right. Both of them show a clear dependence on the azimuthal fault angle θ, whilst being largely invariant with respect to the value of ϕ. The singe fault injection, being composed of a lower number of results for each tile, shows clearer low degree variations in the colour mapping, a characteristic which is instead absent in the double fault case thanks to the fact that the multiple fault configurations allowed for a wider range of values to average over, smoothing the final colour map. The accuracy study step allowed to prove that it is possible to confidently drop the variation of the polar angle ϕ in the successive analyses regarding the qLayer. From a computational standpoint this permitted to reduce computation times by $\sim 99.69\%$.



Single fault QNN accuracy. Double fault QNN accuracy.

Figure 4.11: QNN: Accuracy heatmaps, fault position 1.

At this point, the interest shifted towards testing specific fault cases in order to get a more granular view of the sensitivity of each qubit. To do this, four fault positions have been chosen, according to whether they are placed on the tuple $(qubit0, qubit1)$ or $(qubit2, qubit3)$ and whether the position of the faults precedes or succeeds to the two mid-circuit control gates. The names by which they will be referred to from now onwards are presented in Figure 4.12.

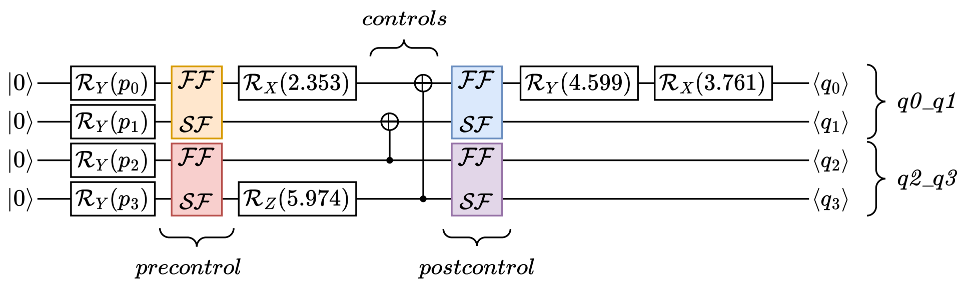
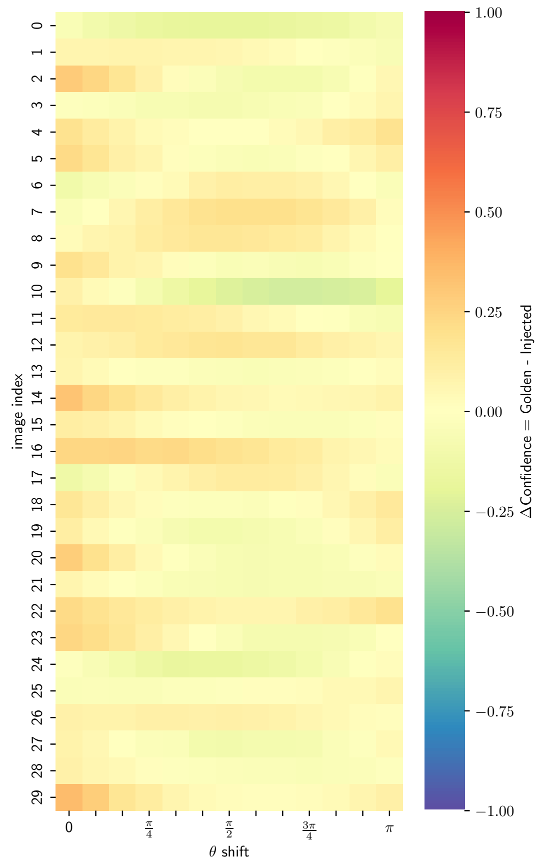


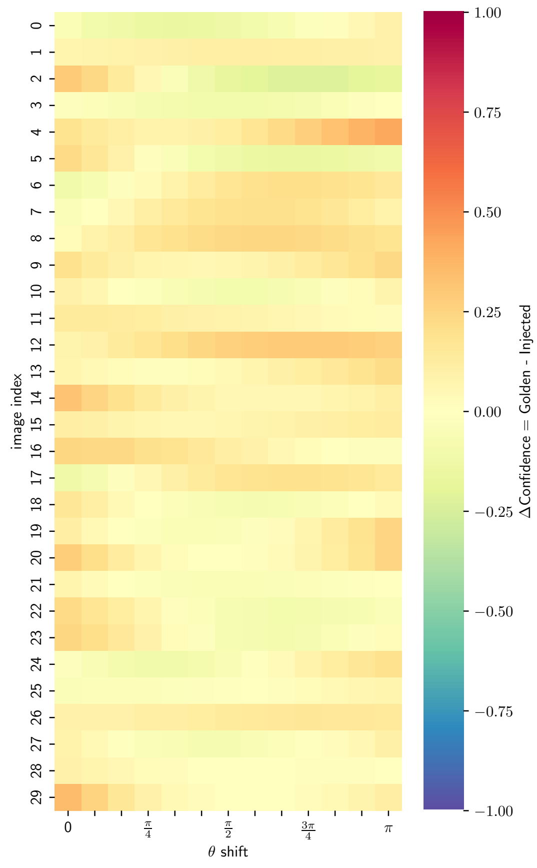
Figure 4.12: QNN: Case study fault positions. *FF* represents the first fault, *SF* the second one. The p parameters are the input pixel values.

The analysis has been repeated for all the remaining fault positions and a comparable independence with respect to the polar angle in injected faults has been found.

Qubit0 and Qubit1 precontrol



$\Delta Confidence_s$ = Golden - Single.



$\Delta Confidence_d$ = Golden - Double.

Figure 4.13: QNN, fault 1: Dataset azimuthal fault bound accuracy heatmaps.

This fault configuration refers to the orange position in Figure 4.12. The heatmaps presented in Figure 4.13 plot the $\Delta Confidence$ for the single and double fault cases with respect to the θ azimuthal angle for each image in the test partition. The $\Delta Confidence$ is computed as the difference between the softmax output of the predicted class in the golden execution of the QNN and the value in predicting that same class in the injection case.
Most of the images share a common pattern of behaviour, with values mostly ranging between -0.30 and 0.20, an effect which can be explained by the fact that the first two qubits are possibly less important in the context of producing the final prediction. The variations among the images are mainly explained by the noise in the circuit simulation, to which the injected faults share a similar impact. In

limited cases associated to images 0, 10 and 24 in the single fault and image 2 in the double fault, the confidence increases, however this effect is sporadic and justified by chance rather than systematic. The double fault increases the $\Delta \mathit{Confidence}$ towards the rightmost part of the second heatmap, as to be expected by the greater number of faults injected in those cases.

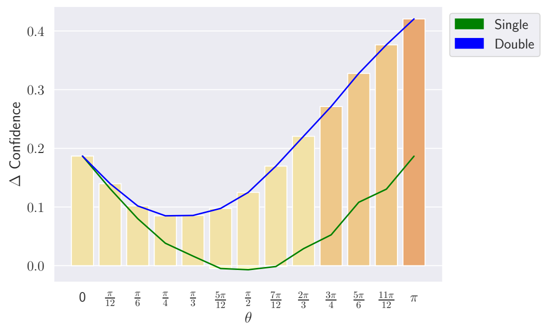


Image 4.

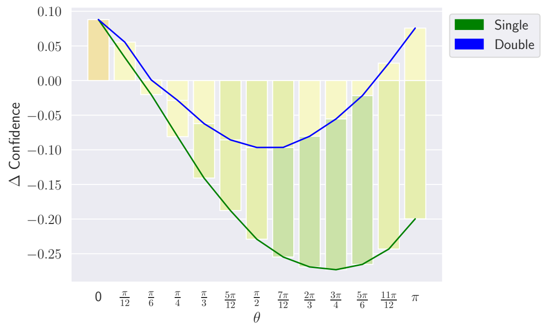


Image 10.

Figure 4.14: QNN, fault 1: Image 4 and 10 $\Delta \mathit{Confidence}$.

Figure 4.14 details the heatmaps of images 4 and 10, the two most extreme settings in the impact of the fault injection. In the first case there is a shared behaviour among single and double faults, with the latter providing a worsening in performance towards the higher values of θ, as the qubits are shifted away more from their original state. The rightmost bar chart shows the opposite behaviour, where the prediction performance improves almost across the whole θ range.

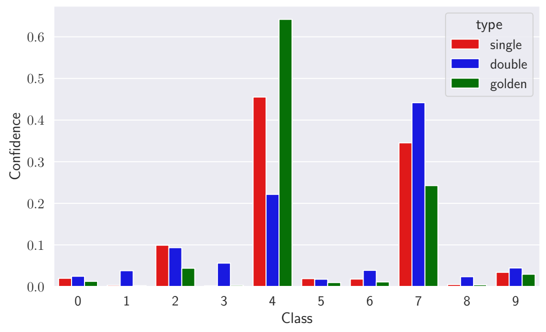


Image 4, $\theta = \pi$.

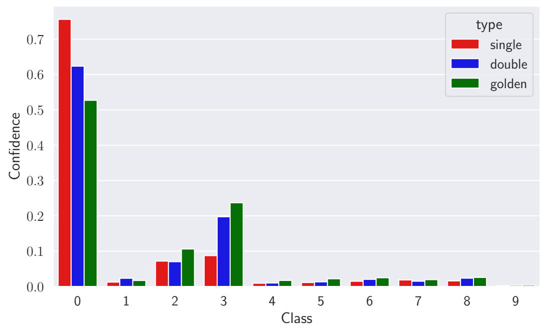
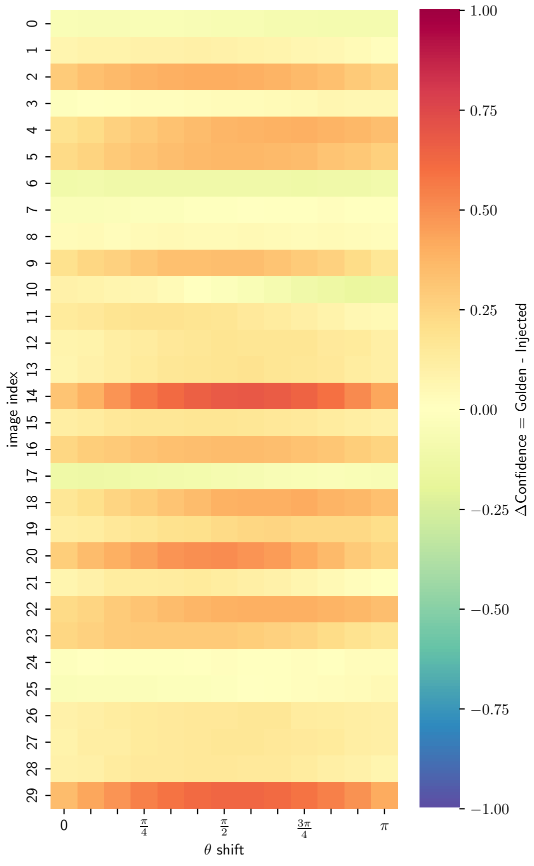


Image 10, $\theta = \frac{\pi}{2}$.

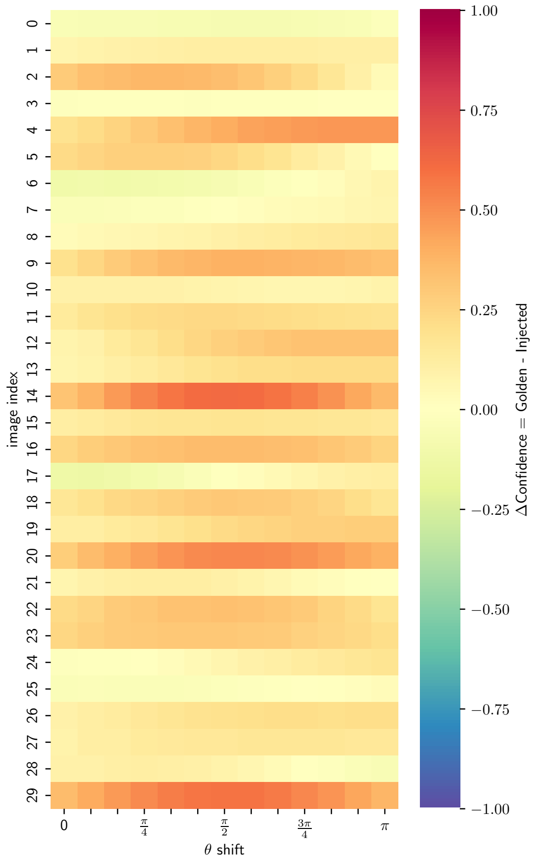
Figure 4.15: QNN, fault 1: Image 4 and 10 softmax outputs.

Figure 4.15 shows the softmax layer outputs of the network with respect to a specific θ angle value. The impact of the double fault on image 4 is the only case that brings the QNN to misclassify *class 4* as *class 7* in the whole dataset, whilst the single fault is still small enough to go unnoticed. Image 10 instead shows the opposite effect, where the accuracy increases with the single fault and drops slightly with the double fault. A part of the probability amplitudes of *classes 2* and *4* get converted to *class 0*, whilst all other amplitudes, already close to zero amplitude, are not affected.

Qubit0 and Qubit1 postcontrol



$\Delta Confidence_s$ = Golden - Single.



$\Delta Confidence_d$ = Golden - Double.

Figure 4.16: QNN, fault 2: Dataset azimuthal fault bound accuracy heatmaps.

This second case refers to the blue position in Figure 4.12. The dataset level heatmaps show a similar behaviour with respect to the pre-control case, with most of the input images undergoing smaller oscillations in the [0,0.25] range, more often than not worsening the performance by a negligible amount, thus not affecting the final predicted label. Again this lower impact shares the same reasoning as before, in that the network's output boasts a lower degree of dependence from the first two output channel of the quanvolution. Again, there is the presence of a limited amount of cases in which the performance improves by a marginal amount, however this is not to be considered a systematic improvement, but rather the byproduct of chance when associated with simulated circuit noise. The double fault generally increases the value of the $\Delta Confidence$, but still not to a point able to cause deviations in the classfication.

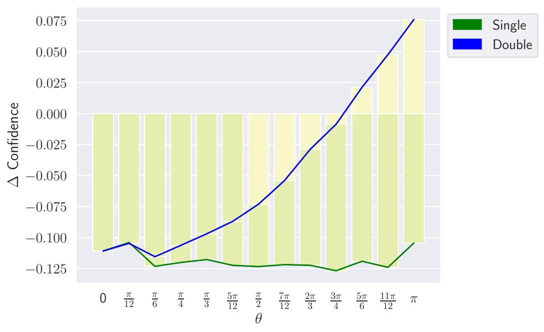


Image 6.

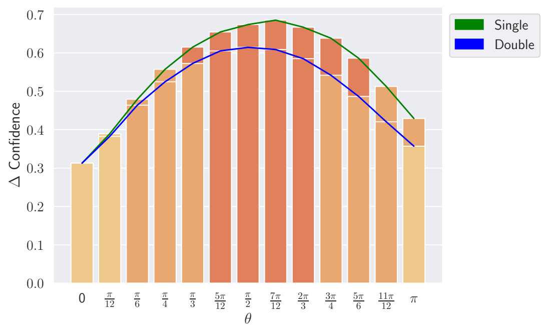


Image 14.

Figure 4.17: QNN, fault 2: Image 6 and 14 $\Delta Confidence$.

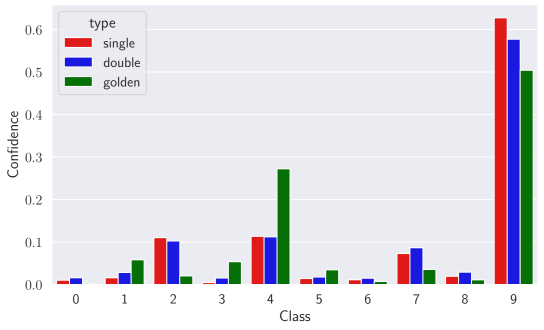


Image 6, $\theta = \frac{\pi}{2}$.

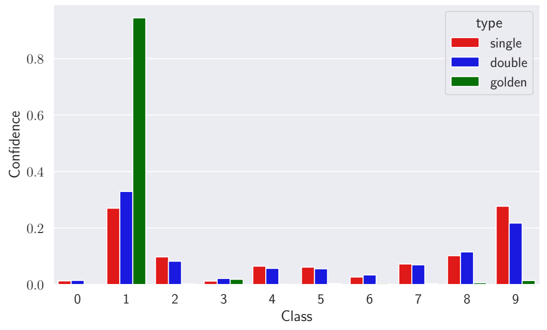


Image 14, $\theta = \frac{\pi}{2}$.

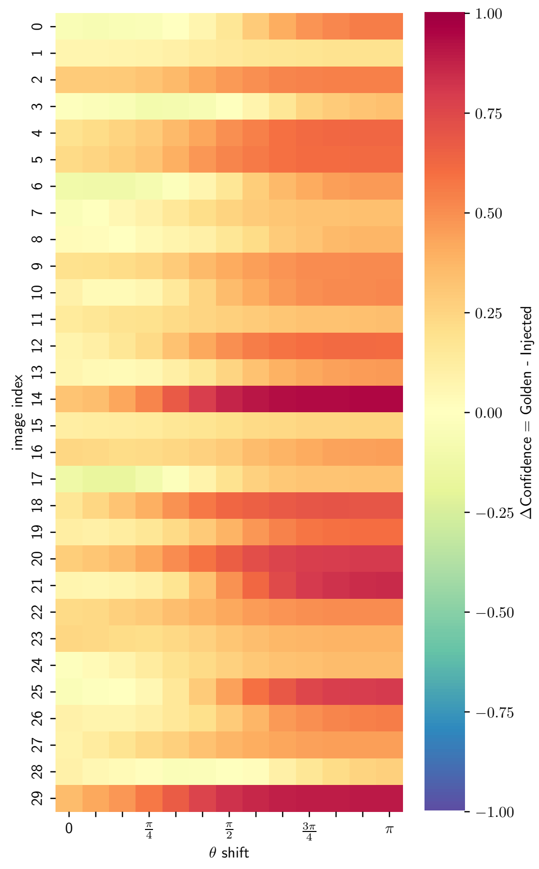
Figure 4.18: QNN, fault 2: Image 6 and 14 softmax outputs.

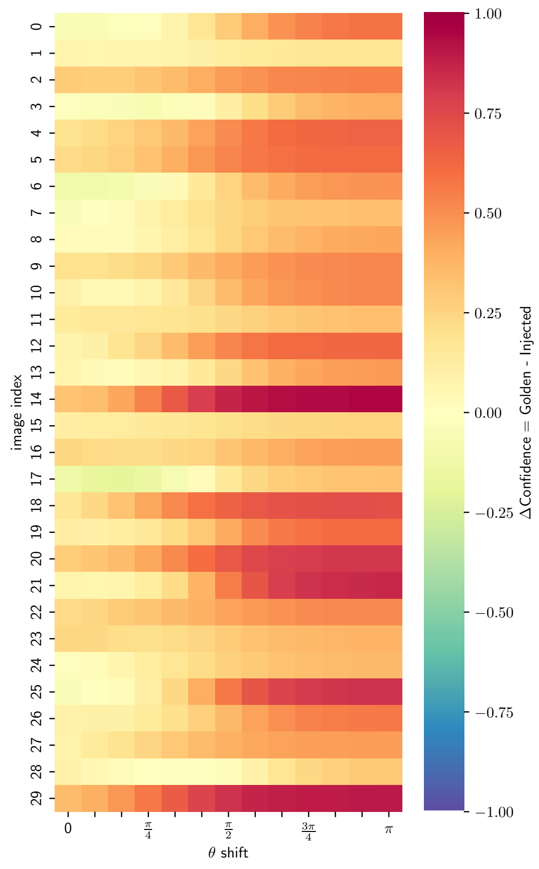
The two most extreme cases are detailed in Figure 4.17. On the left, the bar chart of image 6 clearly shows its independence with respect to the value of the single injection angle θ at the cost of a constant positive shift, as the line is almost flat and in the negative region of the bar chart, whilst with the introduction of the second fault, the previously induced shift is compensated, producing a small and

almost unnoticeable worsening towards the high end of the θ amplitudes. Image 14 is instead the worst performing image, where the faults consistently cause a drop in the confidence varying from 0.3 to up to ~ 0.65 in the middle of the θ range. In this second case, the effect of the double fault is less noticeable with respect to image 6, however the magnitude of the *Confidence* gains back upwards of 10% over the whole scale.
By going more in depth with image 6 at the level of the network's softmax layer output with a fault amplitude of $\frac{\pi}{2}$, in the leftmost bar chart of Figure 4.18, it is possible to see that the predicted *class 9* is still confidently classified as such, and that the second highest prediction, *class 4*, suffers the largest reduction in confidence, which gets evenly distributed among *classes 2*, *7* and *9*. In the opposite way, image 14 at a fault amplitude of $\frac{\pi}{2}$ sees a dramatic fall in the confidence associated with *class 1*, distributing its probability amplitude between almost all the other available classes, especially with *class 9*, causing a switch in the prediciton for the single fault case, but barely resisting it in the double fault one.

Qubit2 and Qubit3 precontrol

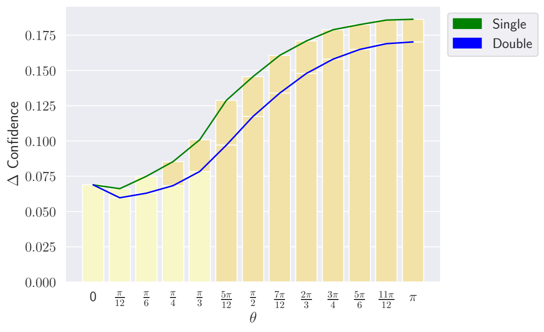
The third setting refers to the red position in Figure 4.12. It is noticeable how most of the images display a worse Δ*Confidence* with respect to both of the *q0_q1* cases. In fact, there is an overall increase which is proportional to the value of the fault angle θ. The single and double heatmaps share this common behaviour across the dataset, suggesting that qubit 2 has a higher incidence on the QNN's output and as such is more sensible to faults with respect to qubit 3, which however still manages to provide a negative effect on the more resilient images of the single fault case. Importantly, no image has an incidental positive gain in the Δ*Confidence*, with most values ranging between 0.0 and 0.95.
Figure 4.20 details the comparison between the single and double faults on the Δ*Confidence* of images 1 and 14, respectively the ones with the lowest and highest overall shifts. In the bar chart on the left, representing the behaviour of image 1, it is possible to see that a smooth increase in the measured metric with respect to the angle θ. The single and double fault cases both share a commonly shaped curve, however the latter causes a slight decrease in the Δ*Confidence* values of 0.025 across the whole chart. On the right figure, the impact of the fault is much more prominent, starting from a minimum value of ~ 0.30 when $\theta = 0$ and rising steadily up to ~ 0.90 and beyond when θ crosses the $\frac{\pi}{2}$ threshold.
The softmax output bar charts for images 1 and 14, presented in Figure 4.21, show more in depth the devastating effects of faults on these specific qubits. Both images have been considered at $\theta = \pi$, since it is the point of greatest divergence with respect to the golden execution. Image 1 shows a very low confidence in the golden case, as *class 1* and *class 6* have almost the same value. The single fault causes an





$\Delta \mathit{Confidence}_s$ = Golden - Single. $\Delta \mathit{Confidence}_d$ = Golden - Double.

Figure 4.19: QNN, fault 3: Dataset azimuthal fault bound accuracy heatmaps.



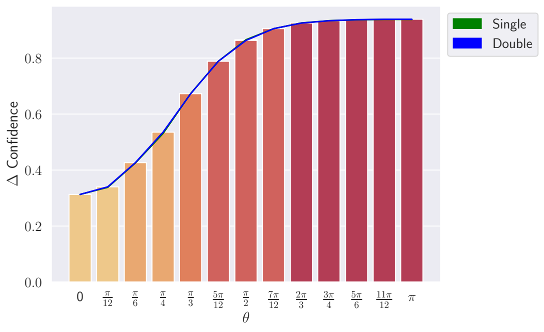


Image 1. Image 14.

Figure 4.20: QNN, fault 3: Image 1 and 14 $\Delta \mathit{Confidence}$.

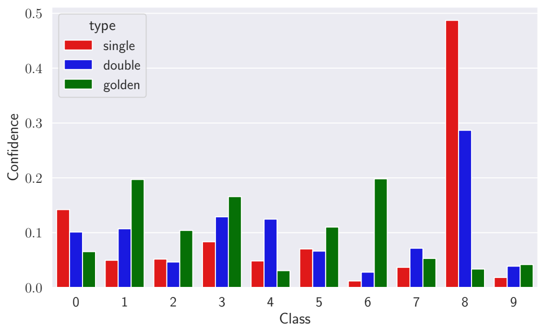


Image 1, $\theta = \pi$.

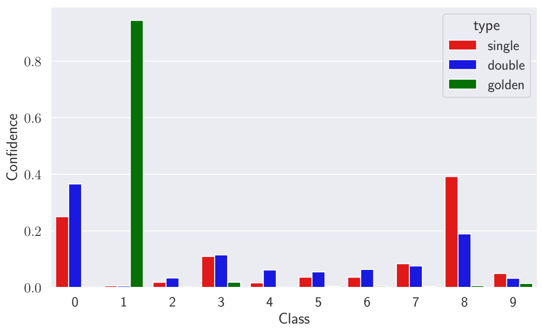


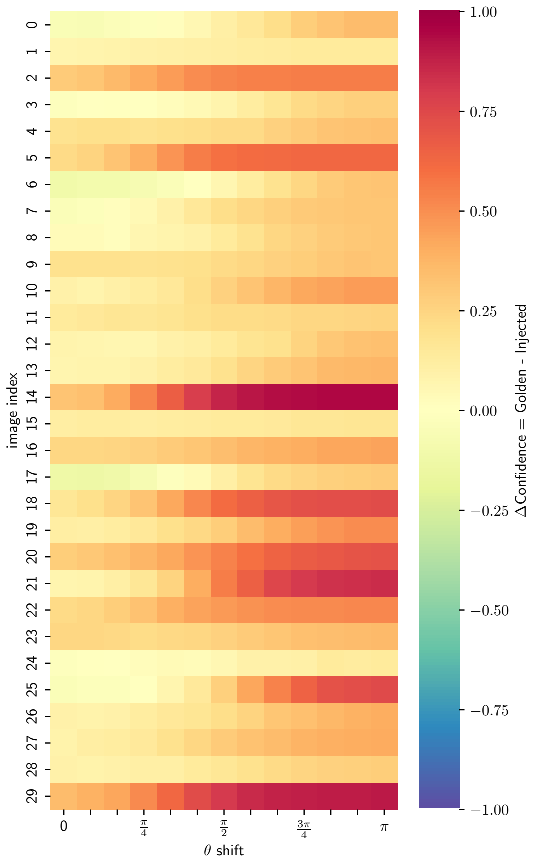
Image 14, $\theta = \pi$.

Figure 4.21: QNN, fault 3: Image 1 and 14 softmax outputs.

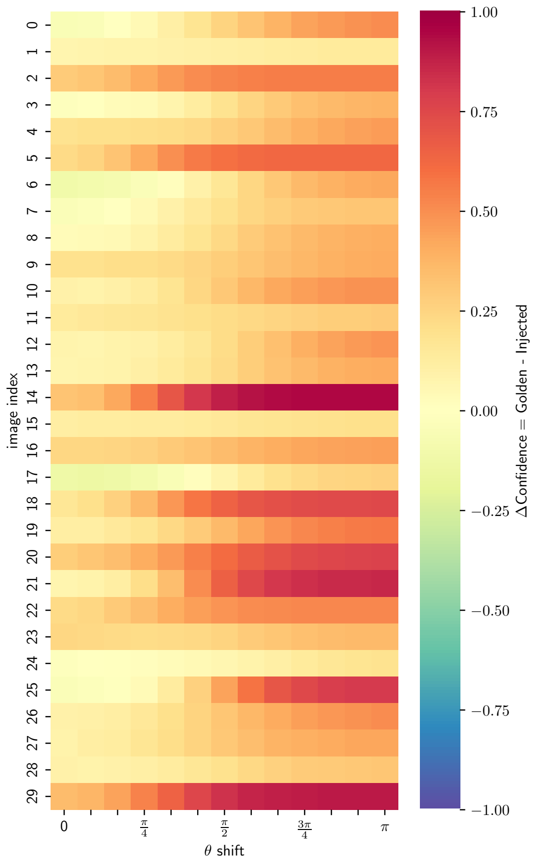
abrupt shift in the output towards *class 8*, which is among the lowest scoring classes in the golden execution. The introduction of the double fault compensates for this initial rise and spreads the probability distribution between the other classes. In both cases, the QNN is forced to produce a classification which is different from the one in the golden setting. Image 14 is the worst performing case of this setup, with the golden prediction of *class 1* being inverted to become the lowest confidence predicted class, whilst *class 0* and *class 8* get elevated to the predicted class for the double and single fault cases.

Qubit2 and Qubit3 postcontrol

This last setting refers to the purple fault position in Figure 4.12. The dataset heatmaps are strikingly similar with the ones from the *q2_q3_precontrol* fault configuration, with a few exceptions in images 17 and 24. Similarly, most of the $\Delta Confidence$ ranges in [0.0,0.95]. The relative effect of the fault occurring after the controlled *Pauli-X* gates is thus negligible, as in the case of the *q0_q1_postcontrol* when compared with the *precontrol* configuration. The double fault presents a low impact on the overall confidence, generally providing a slight increase towards the higher values of θ.
Once again, as seen in figure 4.23, the best and worst performing images have been plotted in detail in the single-double comparison bar chart. It is important to note that for both of them, the $\Delta Confidence$ doesn't improve, apart from a very minor and almost unnoticeable shift for $\theta = 0$ in images 6 and 24, once again proving the previously introduced explaination for this phenomenon with random noise. On the left, image 14 shows a constantly rising $\Delta Confidence$ up to $\theta = \frac{\pi}{2}$, where it stabilizes around a value of ~ 0.95. The double fault provides an extremely marginal increase in the values located in the middle of the figure. Image 24, on



$\Delta Confidence_s$ = Golden - Single.



$\Delta Confidence_d$ = Golden - Double.

Figure 4.22: QNN, fault 4: Dataset azimuthal fault bound accuracy heatmaps.

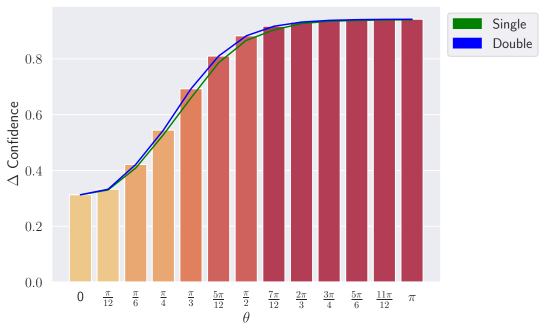


Image 14.

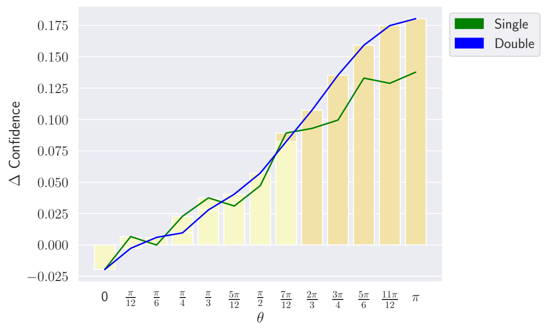


Image 24.

Figure 4.23: QNN, fault 4: Image 14 and 24 $\Delta Confidence$.

the right, features a similar constantly rising bar chart, maxing out at $\theta = \pi$, which however ranges over much smaller values of the considered metric. The double fault is responsible for a more noticeable shift with respect to the single fault one.

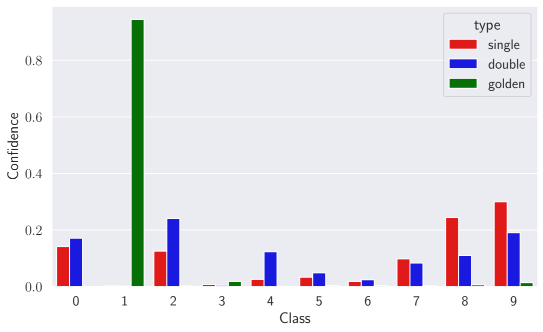


Image 14, $\theta = \pi$.

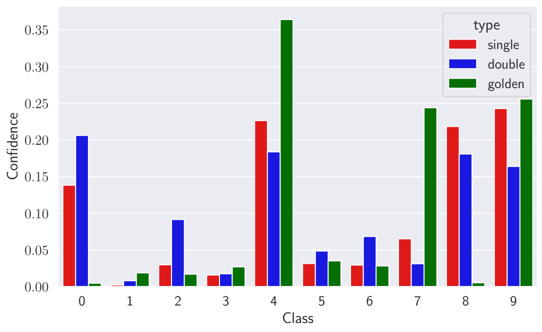


Image 24, $\theta = \pi$.

Figure 4.24: QNN, fault 4: Image 14 and 24 softmax outputs.

The softmax output charts in Figure 4.24 show the variations in the confidence caused by the faults at a numerical level, both considered at $\theta = \pi$. Image 14 undergoes the most radical change, with the golden predicted *class 1* dropping consistently to ~ 0.0 in both the single and double fault cases, to the gain of *classes 8* and *9* with the single fault and *classes 2* and *9* in the double fault. In both cases, the QNN mislabels the input. Image 24, despite being more resilient to both kinds of faults, can not manage to preserve the correct output, and once again lowers the golden *class 4*'s confidence in favour of *classes 8* and *9* in the single fault case and mainly to *class 0* in the double fault one.

Final analysis

The overall behaviour of the quanvolutional layer with respect to faults is not yet fully characterized, however, given the information provided by the previously presented exploratory analysis it is possible to make some assumptions.
The qLayer's resilience seems to be very much dependent on which qubit gets affected. In fact, both the *q2_q3* fault positions have shown to be more prone to hard faults rather than the *q0_q1* ones, as suggested by the single qubit QVF heatmaps provided in Figure 4.9.
The depth of the fault has instead proven to be a less impactful factor, often amounting to small variations which range around the same order of magnitude as those caused by quantum noise.
As such, a more thorough analysis could be conducted by considering as fault positions all the combinations of the four qubits, right after the encoding layer,

extracting a higher level behaviour, and dropping any additional combination with the depth position in the circuit. It will be however left to future research.

Chapter 5

Conclusions

5.1 Digression on the thesis's work

Towards the end of my Bachelor's degree I had followed a course in Quantum Computing, carried out by one of my supervisors, professor Montrucchio, and by professor Carbone. It opened my mind and I enjoyed it thoroughly, so I set out to learn more about the subject. However, when starting the Master's degree, I focused on other different topics involving computer graphics and machine learning, partly losing track of this initial drive. When I set out to choose my thesis' argument in December of last year, I considered multiple options revolving around the same fields of my specialization, until I found a proposal in Quantum Computing. It was literally the only one containing the word *quantum* among the almost 200 options available. I knew very well from my past experience that it is a dauntingly complex subject, but I couldn't resist my curiosity and thus I set out for it.
Right at the start of the research work, there were endless possibilities for merging quantum computing and fault injection. Literature on the topic at the time was, and still is to some extent, quite novel, with open issues such as Error Detection and Correction. Thus, I devoted the first month of work in getting myself up to speed with everything quantum, following online lectures by Qiskit, Maria Schuld and Peter Wittek. I read the whole introductory textbook by Thomas Wong *Introduction to Classical and Quantum Computing* in one week. I integrated some additional knowledge from Nielsen and Young's *Introduction to Quantum Computing* and from the 1998 *Lecture Notes for Physics 229: Quantum Information and Computation* by John Preskill (I found extremely funny that those notes are as ~~old~~ young as me). Lastly, I started reading research papers in the field and to my utmost surprise, I even partly began to understand what they talking about.
It was time to start working. I knew that the research group I was in contact with at my university had some ground already in quantum fault injection, but

since it is a wide topic that can be tackled in multiple ways, under the suggestion of my supervisors, I jotted down some possible themes of research for expanding onto their work. Despite discussing a lot with all of them about it, I discovered that there was really no better or worse option, so I got lucky enough to decide by myself. Nobody told me what to do, and there was no preconception of what to do. I adapted the research group's code of QuFI from the Qiskit backend to Pennylane and I tried to replicate the results from their papers. Meanwhile, I thought of a way to merge the subject of machine learning into my work. The natural consequence was adapting the newly written fault injector to run QML subroutines. A first test has been conducted with the QSVM, as proving grounds for what came next. It was promising, as such I tried to investigate a more complex architecture, a QNN. The model I used is an extremely simplified one, with a reduced version of the MNIST handwritten digits dataset. Despite cutting those corners, I was not able to provide a full, top down analysis of this second neural network due to hardware limitations for quantum simulation, but I have been able to provide some in depth insight on the inner workings of the quanvolutional layer, spotting out some of the more sensible qubits in the quantum circuit. Nonetheless, a modular method of evaluation has been implemented and tested, hopefully paving the way for further research in the same topic, possibly on other architectures.
All in all, this experience has been extremely complex, stressful, rewarding and most of all enlightening. I have had the pleasure to work with an exceptional team of people, always available for sharing ideas with me and striving for understanding. I have learnt so much, yet there is so much more to learn.
I am unsure whether this work will actually ever become the foundation of something of greater purpose for humanity, or if it will simply be relegated among the thousands other documents, scribbles and notes of a distant past nobody will ever look back to. However, this is not important. What I do know is that, one way or another, Quantum Computing is going to revolutionise our concept of information and reality as a whole. Having had the chance to be part of such a great feat of research is a reward in and of itself.

5.2 Future works

This Thesis left many open ends, more than the ones I thought it would when I began. They will be listed in no particular order in the following section, as a remainder and inspiration for further research in the years to come.

- Optimising QuFI, by adding fault gates as parametrizable objects, rather than generating a new QNode object in memory for each injection. Provide a better interface for load optimization in the simulations, which is currently governed by the device's core count only. Add support for specifying a subset of injection targets, instead of using all of the available combinations. Investigating further parallelization capabilities by exploiting the JAX library.

- Providing a full top down analysis of the accuracy of the QNN with respect to single and double faults.
- Adding a more complex fault model in QuFI, which takes into consideration additional factors such as transmons' physical proximity and possibly circuit level faults.
- Testing the current fault model onto other backends available with Pennylane, such as Rigetti Forest, Xanadu Strawberry Fields, Amazon AWS Bracket, IonQ and Honewell.
- Testing other, possibly more advanced, QML architectures.

Bibliography

[1] Paul Benioff. «Quantum mechanical hamiltonian models of turing machines». In: *Journal of Statistical Physics* 29.3 (Nov. 1982), pp. 515–546. ISSN: 1572-9613. DOI: 10.1007/BF01342185. URL: https://doi.org/10.1007/BF01342185 (cit. on p. 1).

[2] Paul A. Benioff. «Quantum Mechanical Hamiltonian Models of Discrete Processes That Erase Their Own Histories: Application to Turing Machines». In: *International Journal of Theoretical Physics* 21.3-4 (Apr. 1982), pp. 177–201. DOI: 10.1007/BF01857725 (cit. on p. 1).

[3] Richard P. Feynman. «Quantum mechanical computers». In: *Foundations of Physics* 16.6 (June 1986), pp. 507–531. ISSN: 1572-9516. DOI: 10.1007/BF01886518. URL: https://doi.org/10.1007/BF01886518 (cit. on pp. 1, 14).

[4] D. Deutsch. «Quantum theory, the Church-Turing principle and the universal quantum computer». In: *Proceedings of the Royal Society of London Series A* 400.1818 (July 1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070 (cit. on pp. 1, 13).

[5] Peter W. Shor. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137%5C%2Fs0097539795293172 (cit. on pp. 1, 15).

[6] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. DOI: 10.48550/ARXIV.QUANT-PH/9605043. URL: https://arxiv.org/abs/quant-ph/9605043 (cit. on p. 1).

[7] Isaac L. Chuang, Neil Gershenfeld, and Mark Kubinec. «Experimental Implementation of Fast Quantum Searching». In: *Physical Review Letters* 80.15 (Apr. 1998), pp. 3408–3411. DOI: 10.1103/PhysRevLett.80.3408 (cit. on p. 1).

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (cit. on pp. 1, 6, 9, 13).

[9] Stanley P. Gudder. «Book Review: Quantum Computation and Quantum Information. By Michael A. Nielsen and Isaac L. Chuang. Cambridge University Press, Cambridge, United Kingdom, 2000, i–xxv+676 pp., $42.00 (hardcover)». In: *Foundations of Physics* 31.11 (Nov. 2001), pp. 1665–1667. ISSN: 1572-9516. DOI: 10.1023/A:1012603118140. URL: https://doi.org/10.1023/A:1012603118140 (cit. on p. 1).

[10] Manuel Vogel. «Quantum Computation and Quantum Information, by M.A. Nielsen and I.L. Chuang. Scope: textbook. Level: advanced undergraduate and above». In: *Contemporary Physics* 52.6 (Nov. 2011), pp. 604–605. DOI: 10.1080/00107514.2011.587535 (cit. on p. 1).

[11] Zhi Zhao, Yu-Ao Chen, An-Ning Zhang, Tao Yang, Hans J. Briegel, and Jian-Wei Pan. «Experimental demonstration of five-photon entanglement and open-destination teleportation». In: *Nature* 430.6995 (July 2004), pp. 54–58. DOI: 10.1038/nature02643. arXiv: quant-ph/0402096 [quant-ph] (cit. on p. 1).

[12] R. Hanson, L. P. Kouwenhoven, J. R. Petta, S. Tarucha, and L. M. K. Vandersypen. «Spins in few-electron quantum dots». In: *Rev. Mod. Phys.* 79 (4 Oct. 2007), pp. 1217–1265. DOI: 10.1103/RevModPhys.79.1217. URL: https://link.aps.org/doi/10.1103/RevModPhys.79.1217 (cit. on p. 1).

[13] J. H. Plantenberg, P. C. de Groot, C. J. P. M. Harmans, and J. E. Mooij. «Demonstration of controlled-NOT quantum gates on a pair of superconducting quantum bits». In: *Nature* 447.7146 (June 2007), pp. 836–839. DOI: 10.1038/nature05896 (cit. on p. 1).

[14] P. C. Maurer et al. «Room-Temperature Quantum Bit Memory Exceeding One Second». In: *Science* 336.6086 (June 2012), p. 1283. DOI: 10.1126/science.1220513 (cit. on p. 2).

[15] Kamyar Saeedi et al. «Room-Temperature Quantum Bit Storage Exceeding 39 Minutes Using Ionized Donors in Silicon-28». In: *Science* 342.6160 (2013), pp. 830–833. DOI: 10.1126/science.1239584. eprint: https://www.science.org/doi/pdf/10.1126/science.1239584. URL: https://www.science.org/doi/abs/10.1126/science.1239584 (cit. on p. 2).

[16] Manjin Zhong, Morgan P. Hedges, Rose L. Ahlefeldt, John G. Bartholomew, Sarah E. Beavan, Sven M. Wittig, Jevon J. Longdell, and Matthew J. Sellars. «Optically addressable nuclear spins in a solid with a six-hour coherence time». In: *Nature* 517.7533 (Jan. 2015), pp. 177–180. DOI: 10.1038/nature14025 (cit. on p. 2).

[17] W. Pfaff et al. «Unconditional quantum teleportation between distant solid-state quantum bits». In: *Science* 345.6196 (Aug. 2014), pp. 532–535. DOI: `10.1126/science.1253512`. URL: `https://doi.org/10.1126%5C%2Fscience.1253512` (cit. on p. 2).

[18] Simon J. Devitt. «Performing quantum computing experiments in the cloud». In: *Physical Review A* 94.3 (Sept. 2016). DOI: `10.1103/physreva.94.032329`. URL: `https://doi.org/10.1103%5C%2Fphysreva.94.032329` (cit. on p. 2).

[19] Daniel Alsina and José Ignacio Latorre. «Experimental test of Mermin inequalities on a five-qubit quantum computer». In: *Physical Review A* 94.1 (July 2016). DOI: `10.1103/physreva.94.012314`. URL: `https://doi.org/10.1103%2Fphysreva.94.012314` (cit. on p. 2).

[20] John Preskill. «Quantum Computing in the NISQ era and beyond». In: *Quantum* 2 (Aug. 2018), p. 79. DOI: `10.22331/q-2018-08-06-79`. URL: `https://doi.org/10.22331%5C%2Fq-2018-08-06-79` (cit. on pp. 2, 19).

[21] N. Abramson. *Information Theory and Coding*. Electronic science series. McGraw-Hill, 1963. ISBN: 9780070001459. URL: `https://books.google.it/books?id=x4hQAAAAMAAJ` (cit. on p. 2).

[22] Benjamin Schumacher. «Quantum coding». In: *Physical Review A* 51.4 (Apr. 1995), pp. 2738–2747. DOI: `10.1103/PhysRevA.51.2738` (cit. on p. 2).

[23] Alexander S. Holevo. «Bounds for the quantity of information transmitted by a quantum communication channel». In: 1973 (cit. on p. 2).

[24] R.S. Ingarden. *Quantum Information Theory*. Preprint - Instytut Fizyki Uniwersytetu Mikołaja Kopernika. PWN, 1975. URL: `https://books.google.it/books?id=7CYhtwAACAAJ` (cit. on p. 2).

[25] Maria Schuld and Francesco Petruccione. «Introduction». In: *Supervised Learning with Quantum Computers*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-96424-9. DOI: `10.1007/978-3-319-96424-9_1`. URL: `https://doi.org/10.1007/978-3-319-96424-9_1` (cit. on pp. 3, 41).

[26] P. A. M. Dirac. «A new notation for quantum mechanics». In: *Proceedings of the Cambridge Philosophical Society* 35.3 (Jan. 1939), p. 416. DOI: `10.1017/S0305004100021162` (cit. on p. 3).

[27] F. Bloch. «Nuclear Induction». In: *Physical Review* 70.7-8 (Oct. 1946), pp. 460–474. DOI: `10.1103/PhysRev.70.460` (cit. on p. 4).

[28] F. T. Arecchi, Eric Courtens, Robert Gilmore, and Harry Thomas. «Atomic Coherent States in Quantum Optics». In: *Phys. Rev. A* 6 (6 Dec. 1972), pp. 2211–2237. DOI: `10.1103/PhysRevA.6.2211`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.6.2211` (cit. on p. 4).

[29] Wikipedia contributors. *Observer effect (physics) — Wikipedia, The Free Encyclopedia.* 2022. URL: https://en.wikipedia.org/w/index.php?title=Observer_effect_(physics)&oldid=1077850922 (cit. on p. 7).

[30] Kevin C Young. *C191-Lectures.* 2014. URL: https://inst.eecs.berkeley.edu/~cs191/fa14/lectures/lecture89.pdf (cit. on p. 7).

[31] Max Born. «On the Quantum Mechanics of Collision Processes». In: *Zeitschrift für Physik* 37 (June 1926). The German title of this Max Born article is: Die Quantenmechanik der Stoßvorgänge. T. Bollinger translated the article to English on 2021-08-28., pp. 863–868. DOI: https://doi.org/10.48034/20210828. URL: https://tarxiv.org/tao.2021-08-28.pdf (cit. on p. 8).

[32] T. G. Wong. *Introduction to Classical and Quantum Computing.* Rooted Grove, Jan. 2022. URL: http://www.thomaswong.net/introduction-to-classical-and-quantum-computing-1e2p.pdf (cit. on pp. 9, 13).

[33] A. Einstein, B. Podolsky, and N. Rosen. «Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?» In: *Physical Review* 47.10 (May 1935), pp. 777–780. DOI: 10.1103/PhysRev.47.777 (cit. on p. 9).

[34] A. Einstein. *Relativity, the special and the general theory; a popular exposition.* 1920 (cit. on p. 9).

[35] David P. DiVincenzo. «The Physical Implementation of Quantum Computation». In: *Fortschritte der Physik* 48.9-11 (Sept. 2000), pp. 771–783. DOI: 10.1002/1521-3978(200009)48:9/11<771::aid-prop771>3.0.co;2-e. URL: https://doi.org/10.1002%5C%2F1521-3978%5C%28200009%5C%2948%5C%3A9%5C%2F11%5C%3C771%5C%3A%5C%3Aaid-prop771%5C%3E3.0.co%5C%3B2-e (cit. on p. 13).

[36] Wikipedia contributors. *Cosmic ray — Wikipedia, The Free Encyclopedia.* 2022. URL: https://en.wikipedia.org/w/index.php?title=Cosmic_ray&oldid=1084857304 (cit. on p. 24).

[37] J. F. Ziegler. «Terrestrial cosmic rays». In: *IBM Journal of Research and Development* 40.1 (1996), pp. 19–39. DOI: 10.1147/rd.401.0019 (cit. on p. 23).

[38] D. J. Bird et al. «Detection of a cosmic ray with measured energy well beyond the expected spectral cutoff due to cosmic microwave radiation». In: *The Astrophysical Journal* 441 (Mar. 1995), p. 144. DOI: 10.1086/175344. URL: https://doi.org/10.1086%5C%2F175344 (cit. on p. 23).

[39] R. Baumann. «Soft errors in advanced computer systems». In: *IEEE Design Test of Computers* 22.3 (2005), pp. 258–266. DOI: 10.1109/MDT.2005.69 (cit. on p. 23).

[40] J.V. Carreira, D. Costa, and J.G. Silva. «Fault injection spot-checks computer system dependability». In: *IEEE Spectrum* 36.8 (1999), pp. 50–55. DOI: 10.1109/6.780999 (cit. on p. 23).

[41] A. R. Calderbank and Peter W. Shor. «Good quantum error-correcting codes exist». In: *Physical Review A* 54.2 (Aug. 1996), pp. 1098–1105. DOI: 10.1103/physreva.54.1098. URL: https://doi.org/10.1103%5C%2Fphysreva.54.1098 (cit. on p. 24).

[42] A. M. Steane. «Simple quantum error-correcting codes». In: *Physical Review A* 54.6 (Dec. 1996), pp. 4741–4751. DOI: 10.1103/physreva.54.4741. URL: https://doi.org/10.1103%5C%2Fphysreva.54.4741 (cit. on p. 24).

[43] Jiafeng Cui, A J Rasmusson, Marissa D'Onofrio, Yuanheng Xie, Evangeline Wolanski, and Philip Richerme. «Susceptibility of trapped-ion qubits to low-dose radiation sources». In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 54.13 (July 2021), 13LT01. DOI: 10.1088/1361-6455/ac076c. URL: https://doi.org/10.1088%5C%2F1361-6455%5C%2Fac076c (cit. on pp. 24, 25).

[44] P. Foulliat. «Instruments for mev irradiation from ev to mev». In: (2004). URL: http://sons.uniroma2.it/ericeneutronschool/wp-content/uploads/2017/11/Senesi_MeV_n_ERICE2018_v8.pdf (cit. on p. 25).

[45] Lukas Grünhaupt et al. «Loss Mechanisms and Quasiparticle Dynamics in Superconducting Microwave Resonators Made of Thin-Film Granular Aluminum». In: *Physical Review Letters* 121 (Sept. 2018). DOI: 10.1103/PhysRevLett.121.117001 (cit. on p. 25).

[46] R. Barends et al. «Minimizing quasiparticle generation from stray infrared light in superconducting quantum circuits». In: *Applied Physics Letters - APPL PHYS LETT* 99 (Sept. 2011). DOI: 10.1063/1.3638063 (cit. on p. 25).

[47] Antti P. Vepsäläinen et al. «Impact of ionizing radiation on superconducting qubit coherence». In: *Nature* 584.7822 (Aug. 2020), pp. 551–556. DOI: 10.1038/s41586-020-2619-8. URL: https://doi.org/10.1038%5C%2Fs41586-020-2619-8 (cit. on pp. 25, 26).

[48] C. D. Wilen et al. «Correlated charge noise and relaxation errors in superconducting qubits». In: *Nature* 594.7863 (2021), pp. 369–373. DOI: 10.1038/s41586-021-03557-5. URL: https://doi.org/10.1038/s41586-021-03557-5 (cit. on pp. 25, 26).

[49] Daniel Oliveira, Edoardo Giusto, Betis Baheri, Qiang Guan, Bartolomeo Montrucchio, and Paolo Rech. «A Systematic Methodology to Compute the Quantum Vulnerability Factors for Quantum Circuits». In: (2021). DOI: `10.48550/ARXIV.2111.07085`. URL: `https://arxiv.org/abs/2111.07085` (cit. on pp. 26, 28, 56).

[50] G. Catelani, R. J. Schoelkopf, M. H. Devoret, and L. I. Glazman. «Relaxation and frequency shifts induced by quasiparticles in superconducting qubits». In: *Phys. Rev. B* 84 (6 Aug. 2011), p. 064517. DOI: `10.1103/PhysRevB.84.064517`. URL: `https://link.aps.org/doi/10.1103/PhysRevB.84.064517` (cit. on p. 26).

[51] Benjamin R. Archer, Thomas R. Fewell, Burton J. Conway, and Philip W. Quinn. «Attenuation properties of diagnostic x-ray shielding materials». In: *Medical Physics* 21.9 (1994), pp. 1499–1507. DOI: `https://doi.org/10.1118/1.597408`. eprint: `https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.597408`. URL: `https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.597408` (cit. on p. 26).

[52] Poulami Das, Swamit Tannu, Prashant Nair, and Moinuddin Qureshi. «A Case for Multi-Programming Quantum Computers». In: Oct. 2019, pp. 291–303. ISBN: 978-1-4503-6938-1. DOI: `10.1145/3352460.3358287` (cit. on p. 27).

[53] Lei Liu and Xinglei Dou. «QuCloud: A New Qubit Mapping Mechanism for Multi-programming Quantum Computing in Cloud Environment». In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 2021, pp. 167–178. DOI: `10.1109/HPCA51647.2021.00024` (cit. on p. 27).

[54] Lei Liu and Xinglei Dou. «QuCloud: A New Qubit Mapping Mechanism for Multi-programming Quantum Computing in Cloud Environment». In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 2021, pp. 167–178. DOI: `10.1109/HPCA51647.2021.00024` (cit. on p. 27).

[55] Swamit S. Tannu and Moinuddin K. Qureshi. «Not All Qubits Are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers». In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '19. Providence, RI, USA: Association for Computing Machinery, 2019, pp. 987–999. ISBN: 9781450362405. DOI: `10.1145/3297858.3304007`. URL: `https://doi.org/10.1145/3297858.3304007` (cit. on p. 27).

[56] Heljä Kukkonen, Jyrki Rovamo, Kaisa Tiippana, and Risto Näsänen. «Michelson contrast, RMS contrast and energy of various spatial stimuli at threshold». In: *Vision Research* 33.10 (1993), pp. 1431–1436. ISSN: 0042-6989. DOI: `https://doi.org/10.1016/0042-6989(93)90049-3`. URL: `https://www.sciencedirect.com/science/article/pii/0042698993900493` (cit. on p. 28).

[57] Daniel Oliveira, Edoardo Giusto, Emanuele Dri, Nadir Casciola, Betis Baheri, Qiang Guan, Bartolomeo Montrucchio, and Paolo Rech. «QuFI: a Quantum Fault Injector to Measure the Reliability of Qubits and Quantum Circuits». In: (2022). DOI: `10.48550/ARXIV.2203.07183`. URL: `https://arxiv.org/abs/2203.07183` (cit. on p. 28).

[58] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing.* 2021. DOI: `10.5281/zenodo.2573505` (cit. on p. 28).

[59] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations.* 2018. DOI: `10.48550/ARXIV.1811.04968`. URL: `https://arxiv.org/abs/1811.04968` (cit. on p. 28).

[60] Tom M. Mitchell. *Machine learning, International Edition.* McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN: 978-0-07-042807-2. URL: `https://www.worldcat.org/oclc/61321007` (cit. on p. 33).

[61] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: `10.1007/BF00994018`. URL: `https://doi.org/10.1007/BF00994018` (cit. on p. 37).

[62] Asa Ben-Hur, David Horn, Hava Siegelmann, and Vladimir Vapnik. «Support Vector Clustering». In: *Journal of Machine Learning Research* 2 (Nov. 2001), pp. 125–137. DOI: `10.1162/15324430260185565` (cit. on p. 37).

[63] Shuang Wu, Guanrui Wang, Pei Tang, Feng Chen, and Luping Shi. *Convolution with even-sized kernels and symmetric padding.* 2019. DOI: `10.48550/ARXIV.1903.08385`. URL: `https://arxiv.org/abs/1903.08385` (cit. on p. 39).

[64] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: `10.1109/5.726791` (cit. on p. 40).

[65] Jean-Baptiste Michel et al. «Quantitative Analysis of Culture Using Millions of Digitized Books». In: *Science* 331.6014 (2011), pp. 176–182. DOI: `10.1126/science.1199644`. eprint: `https://www.science.org/doi/pdf/10.1126/science.1199644`. URL: `https://www.science.org/doi/abs/10.1126/science.1199644` (cit. on p. 41).

[66] Vedran Dunjko and Peter Wittek. «A non-review of Quantum Machine Learning: trends and explorations». In: *Quantum Views* 4 (Mar. 2020), p. 32. DOI: 10.22331/qv-2020-03-17-32. URL: https://doi.org/10.22331/qv-2020-03-17-32 (cit. on p. 41).

[67] Ewin Tang. «A Quantum-Inspired Classical Algorithm for Recommendation Systems». In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 217–228. ISBN: 9781450367059. DOI: 10.1145/3313276.3316310. URL: https://doi.org/10.1145/3313276.3316310 (cit. on p. 41).

[68] Gexiang Zhang. «Quantum-inspired evolutionary algorithms: a survey and empirical study». In: *Journal of Heuristics* 17.3 (June 2011), pp. 303–351. ISSN: 1572-9397. DOI: 10.1007/s10732-010-9136-0. URL: https://doi.org/10.1007/s10732-010-9136-0 (cit. on p. 41).

[69] Maria Schuld and Nathan Killoran. «Quantum Machine Learning in Feature Hilbert Spaces». In: *Physical Review Letters* 122.4 (Feb. 2019). DOI: 10.1103/physrevlett.122.040504. URL: https://doi.org/10.1103%5C%2Fphysrevlett.122.040504 (cit. on p. 43).

[70] Vojtech Havlicek, Antonio D. Corcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. «Supervised learning with quantum-enhanced feature spaces». In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. DOI: 10.1038/s41586-019-0980-2. URL: https://doi.org/10.1038%5C%2Fs41586-019-0980-2 (cit. on pp. 43, 50, 54).

[71] Ryan LaRose and Brian Coyle. «Robust data encodings for quantum classifiers». In: *Physical Review A* 102.3 (Sept. 2020). DOI: 10.1103/physreva.102.032420. URL: https://doi.org/10.1103%5C%2Fphysreva.102.032420 (cit. on p. 44).

[72] Phuc Q. Le, Fangyan Dong, and Kaoru Hirota. «A flexible representation of quantum images for polynomial preparation, image compression, and processing operations». In: *Quantum Information Processing* 10.1 (Feb. 2011), pp. 63–84. ISSN: 1573-1332. DOI: 10.1007/s11128-010-0177-y. URL: https://doi.org/10.1007/s11128-010-0177-y (cit. on p. 44).

[73] Yi Zhang, Kai Lu, Yinghui Gao, and Mo Wang. «NEQR: a novel enhanced quantum representation of digital images». In: *Quantum Information Processing* 12.8 (Aug. 2013), pp. 2833–2860. ISSN: 1573-1332. DOI: 10.1007/s11128-013-0567-z. URL: https://doi.org/10.1007/s11128-013-0567-z (cit. on p. 45).

[74] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. «Evaluating analytic gradients on quantum hardware». In: *Phys. Rev. A* 99 (3 Mar. 2019), p. 032331. DOI: `10.1103/PhysRevA.99.032331`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.99.032331` (cit. on p. 45).

[75] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. «The quest for a Quantum Neural Network». In: *Quantum Information Processing* 13.11 (Nov. 2014), pp. 2567–2586. ISSN: 1573-1332. DOI: `10.1007/s11128-014-0809-8`. URL: `https://doi.org/10.1007/s11128-014-0809-8` (cit. on p. 47).

[76] Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. «Continuous-variable quantum neural networks». In: *Phys. Rev. Research* 1 (3 Oct. 2019), p. 033063. DOI: `10.1103/PhysRevResearch.1.033063`. URL: `https://link.aps.org/doi/10.1103/PhysRevResearch.1.033063` (cit. on p. 47).

[77] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. «Quantum Support Vector Machine for Big Data Classification». In: *Phys. Rev. Lett.* 113 (13 Sept. 2014), p. 130503. DOI: `10.1103/PhysRevLett.113.130503`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.113.130503` (cit. on p. 49).

[78] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. *Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits.* 2019. DOI: `10.48550/ARXIV.1904.04767`. URL: `https://arxiv.org/abs/1904.04767` (cit. on pp. 50, 59).

[79] Maria Schuld. «Quantum machine learning models are kernel methods». In: (Jan. 2021) (cit. on p. 52).