



SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Department of Physics
PhD in Physics

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Machine Learning and Quantum Computing

Thesis Advisor

Prof. Stefano Giagu

Supervisors

Dr. Aleandro Nisati

Dr. Paolo Valente

Candidate

Simone Bordoni

1713074

XXXVII cycle

*The most exciting phrase to hear in science, the one
that heralds the most discoveries, is „not “Eureka!”
but “That’s funny...”*

- Isaac Asimov

Abstract

This thesis presents original contributions at the intersection of quantum computing and machine learning, investigating how these two fields can benefit from each other.

The first part of the thesis presents two applications of machine learning techniques in quantum computing. The first contribution involves the use of artificial neural networks for the decoding of quantum error correction codes, resulting in faster decoding times while maintaining a high level of accuracy. The originality of this work lies in the application of explainable machine learning techniques to both improve the decoding performance and to provide insights into the decoding process. The second contribution introduces a novel technique based on reinforcement learning to characterize and simulate the noise affecting a quantum chip. This technique reduces the heuristic assumption on the noise model, making it more adaptable to the specific noise characteristics of the quantum device.

In the second part of the thesis, we demonstrate, with two examples, how it is possible to implement machine learning algorithms on quantum devices by replacing artificial neural networks with parametrized quantum circuits. The first contribution in this section introduces a quantum anomaly detection algorithm, applied to the field of high-energy physics. This algorithm allows for the identification of anomalous patterns in quantum data, maintaining a level of accuracy comparable to classical algorithms. The novelty of this work lies in the first use of quantum circuits for the task of anomaly detection in a muon drift chamber trigger system, with possible future applications to quantum detector systems in high-energy physics experiments. The second contribution presents the quantum version of a generative diffusion model, able to sample quantum states from a well-defined distribution. This first implementation of a quantum diffusion model is not meant to outperform classical models, but rather to show potential advantages and limitations of these kinds of algorithms.

The aim of the thesis is to show how quantum computing can benefit from machine learning, and how machine learning can be implemented on quantum devices, setting the stage for future advancements in both fields.

Keywords: Quantum Computing, Machine Learning, Quantum Machine Learning.

Contents

1	Introduction	1
2	Quantum Computing	5
2.1	Fundamentals	6
2.1.1	Qubits	6
2.1.2	Entanglement	8
2.1.3	Measurements	10
2.1.4	Quantum circuits	11
2.1.5	Quantum advantage	14
2.2	Hardware	16
2.2.1	Fundamental requirements	16
2.2.2	Photons	18
2.2.3	Trapped ions	21
2.2.4	Superconducting qubits	26
2.3	Noise	37
2.3.1	Density matrices	37
2.3.2	Kraus representation	39
2.3.3	Noise channels	40
2.3.4	Principles of quantum error correction	44
2.3.5	Noise digitalization	46
2.4	Qibolab and the transpiling problem	48
2.4.1	Abstract	48
2.4.2	Introduction	48
2.4.3	Transpiler	49
3	Machine Learning	53
3.1	Fundamentals	53
3.1.1	Comparison with standard programming	53
3.1.2	Tasks	55
3.1.3	Learning paradigms	57
3.1.4	Training	58
3.1.5	Neural networks	63
3.2	Advanced topics	69
3.2.1	Deep learning	69

3.2.2	Convolutional neural networks	72
3.2.3	Autoencoder and anomaly detection	74
3.2.4	Generative models	76
3.2.5	Reinforcement learning	79
4	Machine Learning for Quantum Computing	83
4.1	Convolutional neural network based decoders for surface codes	84
4.1.1	Abstract	84
4.1.2	Introduction	84
4.1.3	Surface codes	85
4.1.4	Related works	87
4.1.5	Convolutional neural network based decoders	88
4.1.6	Choice of the training set	94
4.1.7	Model explainability	94
4.1.8	Conclusions	100
4.2	Quantum noise modeling through reinforcement learning	102
4.2.1	Abstract	102
4.2.2	Introduction	102
4.2.3	Related works	103
4.2.4	Algorithm implementation	104
4.2.5	Results	109
4.2.6	Test on quantum algorithms	114
4.2.7	Conclusions	118
5	Quantum Machine Learning	119
5.1	Parametrized quantum circuits	119
5.1.1	Ansatz	120
5.1.2	Training and barren plateaus	122
5.1.3	Comparison between quantum and classical machine learning	124
5.2	Quantum anomaly detection for high energy physics	126
5.2.1	Abstract	126
5.2.2	Introduction	126
5.2.3	Simulation on classic hardware	127
5.2.4	Test on quantum hardware	133
5.2.5	Conclusions	138
5.3	Quantum diffusion models	139
5.3.1	Abstract	139
5.3.2	Introduction	139
5.3.3	Model design	140
5.3.4	Simulation results	144
5.3.5	Quantum hardware execution	148
5.3.6	Conclusions	152
6	Conclusions	153

Bibliography	157
Disclaimer	189
A No Cloning Theorem	193
B Josephson Junction	195
C Randomized Benchmarking	199
D Unitary Representation Theorem	201
E Classical and Quantum Hamming Bound	203
F Loss Functions	205
G Activation Functions	207
H Optimization Methods	209
I Regularization Methods	211
J Backpropagation	213
K Parameter Shift Rule	215
List of Figures	216
List of Tables	224
List of Acronyms	227

Chapter 1

Introduction

This century is witnessing rapid technological advancements. Among these, two of the most significant are machine learning and quantum computing. Machine learning is gaining a lot of attention due to its ability to perform complex tasks that are now reaching or surpassing human-level performance. Quantum computing, on the other hand, has not yet reached the technological level to be applied to real-world problems, but there are many expectations for it. The question that naturally arises concerns the intersection of these two fields. Can machine learning be used to improve quantum computing? And vice versa, can quantum computing be used to improve machine learning? In this thesis, we will analyze both questions, showing practical examples taken from my original contributions to the field.

Machine learning can be used to improve quantum computing, particularly in addressing some critical issues. An important example is error correction [1–9] and error mitigation [10–14], which are fundamental for real applications of quantum computers. For this reason, the first original contribution presented in this thesis [15] concerns the use of machine learning to improve the performance of a particular class of quantum error correction codes, known as surface codes. The use of algorithms based on neural networks allows a reduction in the decoding times required by standard heuristic algorithms while maintaining a high level of accuracy. A novel aspect of this work is the application of explainable machine learning techniques to study the critical aspects of the decoding process. This analysis has led to an improvement in the decoding performance by tailoring a training set that accounts for the most critical cases. The results obtained in this research project have been published in 2023 in *Quantum Information Processing* (Springer) under the title *Convolutional neural network based decoders for surface codes* [15].

Another interesting application of machine learning in quantum computing is the simulation of many-qubit systems, which would be infeasible with classical computers. By compressing the information of quantum states into the parameters of a neural network [16, 17], it is possible to scale to a large number of qubits to test new quantum algorithms. A similar problem, which has not been extensively studied [18–21], is the simulation of quantum algorithms in the presence of noise. We found that it is possible to use machine learning, particularly reinforcement learning, to tackle this problem. The second original contribution presented in this thesis [22, 23] concerns the training of a reinforcement learning agent to learn and simulate the noise pattern of a specific device. This approach offers increased flexibility in reproducing various noise models compared to conventional techniques. Preliminary results of this study were published in the conference pro-

ceedings of AIQxQIA 2023 [22], while a complete paper is currently under review, with the preprint available on arXiv [23] under the title *Quantum noise modeling through reinforcement learning*. This research has been presented at several conferences and workshops, including AIQxQIA 2024 in Bolzano, QC-INFN 2024 in Padova, and ICSC-Spoke10 2024 in Milan.

Machine learning can be used to improve one of the biggest problems in quantum computing, which is the reliability of the hardware. For example, in optimal control [24–26], gate design [27–29], qubit readout [30–32], and calibration [33, 34]. While I have not directly worked in this field, I will present the project Qibolab, to which I have contributed. Qibolab is a platform that helps in the control and calibration of quantum devices. While at the moment this platform is not using machine learning, it could represent a good starting point for the application of these techniques. The development of Qibolab has led to a paper published in 2024 in *Quantum* [35], and the code is currently available on GitHub¹.

The second main topic of this thesis regards the use of quantum computing in the field of machine learning. A first possibility regards the use of quantum devices to speed up particular subroutines of machine learning algorithms [36–39]. However, these kinds of algorithms require fault-tolerant and large-scale quantum computers, which probably won't be available in the near future. In this work, we have focused on the study of machine learning algorithms that can be fully implemented on quantum devices. This field of study is known as Quantum Machine Learning (QML) [40]. The main idea behind QML is to use parametrized quantum circuits in place of artificial neural networks. In particular, in this work, we have focused on small-scale circuits that can be implemented on near-term devices with a high level of noise.

The first original contribution to the field, presented in this thesis [41], introduces a QML algorithm to perform anomaly detection directly on quantum states. The algorithm has been applied in the context of high energy physics, and represents a first application of quantum circuits for the task of characterizing anomalous events in the trigger system of a particle detector. The possibility of using quantum circuits to process quantum data in high energy physics experiments could pave the way for future applications of quantum detectors in this field.

The results of this research have been published in 2023 under the name *Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits* [41]. The work has been presented at several conferences and workshops, including QC-INFN 2022 in Bologna, QAI 2023 in Naples, and EuCAIFCon 2024 in Amsterdam.

The second original contribution to the field of QML regards the first implementation of the quantum version of a generative diffusion model [42]. The ability to sample quantum states from a well-defined distribution could find applications in many different fields [43] such as quantum cryptography [44], quantum metrology [45], fidelity benchmarking [46], and theoretical physics [47, 48]. The aim of this work is not to outperform classical algorithms, but to show that quantum data can be processed by QML algorithms in a similar way as classical data is processed using classical machine learning algorithms. This ability could become extremely important in the future, assuming that quantum computers will complement classical ones.

Preliminary results are available on arXiv in the preprint *Quantum Diffusion Models* [42], while a more detailed version of the article with complete results is currently under review. The quantum

¹<https://github.com/qiboteam/qibolab>

diffusion model has been presented at several conferences such as AIQxQIA 2023 in Rome, QC-INFN 2024 in Padova, and EuCAIFCon 2024 in Amsterdam.

This thesis is organized as follows.

The first two chapters are dedicated to the background knowledge needed to understand the original work presented. In Chapter 2, we will introduce quantum computing. We will start by presenting the basic concepts of quantum circuits, then we will analyze the current state of quantum hardware, and finally, we will give some insights on quantum noise and error correction. In this chapter, we will also present the project Qibolab, with particular focus on the transpiler, which is the part of the project to which I have contributed.

In Chapter 3, we will introduce machine learning. We will start by presenting the basic concepts of neural networks and training procedures, then we will focus on some more advanced topics such as reinforcement learning and generative models, which are used in the original contributions presented in this thesis.

After the background chapters, we will present the original contributions organized in two chapters. Chapter 4 presents the works related to the application of machine learning in quantum computing.

The first work regards the use of neural networks in decoding quantum error correction codes. The second work regards the use of reinforcement learning to characterize and simulate the noise affecting a quantum chip.

Chapter 5 presents the original contributions in the field of quantum machine learning after a brief introduction to parametrized quantum circuits. The first original work regards the implementation of a quantum anomaly detection algorithm, applied to the field of high energy physics.

The second work regards the realization of a quantum generative diffusion model, able to sample quantum states from a well-defined distribution.

Chapter 2

Quantum Computing

Quantum computing has become an important research topic due to its potential to solve problems that classical computers cannot handle efficiently. For instance, Shor’s quantum algorithm could factorize large numbers swiftly [49], a task that classical computers struggle with due to the exponential scaling of the best-known classical factorization algorithm. Similarly, Grover’s search algorithm [50] could reduce the time to find an element in an unstructured database of n elements from $O(n)$ to $O(\sqrt{n})$. Despite its rapid development, quantum computing is still in its infancy, with key milestones such as quantum error correction yet to be achieved. Recent years have seen claims of experimental realization of quantum advantage [51–54], a significant milestone underscored by the extensive efforts of research groups and companies. However, it’s worth noting that these claims often involve devices specifically designed for the task, with limited reprogrammability. The current generation of universal Noisy Intermediate Scale Quantum (NISQ) computers [55] is too noisy for implementing quantum supremacy algorithms. These devices can only produce simple results and offer no advantage over classical computation. The next generation of quantum computers, expected to be more noise-resistant, may fully realize quantum supremacy. This work focuses solely on gate-based quantum computing, excluding adiabatic quantum computing (quantum annealing) [56, 57] or measurement-based quantum computing [58]. Although quantum annealing shows promising results in solving optimization problems [59], it is considered less general than gate-based quantum computers.

This chapter provides a general introduction to quantum computing from both theoretical and experimental perspectives. While the topics discussed should be familiar to experts, they serve as a foundation for this work.

Section 2.1 reviews the formalism of quantum computing.

Section 2.2 examines the current state of quantum device realization.

Section 2.3 analyzes the issue of noise in quantum devices, a fundamental aspect of the projects presented in this work.

Finally, Section 2.4 introduces an original project, QiboLab, which aims to assist in the control of hardware quantum devices. My main contribution to this project, the development of the transpiler, is presented.

2.1 Fundamentals

This section reviews the fundamental concepts in quantum computing. Section 2.1.1 introduces qubits and the mathematical formalism of quantum mechanics. Sections 2.1.2 and 2.1.3 delve into multi-qubit systems and measurements, offering insights into entanglement and its implications. Section 2.1.4 presents the formalism of quantum circuits and reviews the most significant gates in quantum computing. Finally, Section 2.1.5 discusses the advantages of quantum computing over classical computing.

2.1.1 Qubits

The main characteristic that differentiates a quantum computer from a classical computer lies in the fundamental unit: the qubit (quantum bit). A qubit represents the state of a two-level quantum system. In the mathematical formalism of quantum mechanics it is represented by a unit vector in a complex Hilbert space of dimension two. In this space, the computational basis is composed of two orthonormal vectors $|0\rangle$ and $|1\rangle$:

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

A generic state $|\psi\rangle$ can be represented in the computational basis by exploiting the superposition principle:

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

where α and β are complex numbers that meet the normalization condition: $|\alpha|^2 + |\beta|^2 = 1$.

As a quantum state is defined up to a global phase, it is possible to write, without loss of generality, the state $|\psi\rangle$ as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle = \begin{bmatrix} \cos(\theta/2) \\ e^{i\phi} \sin(\theta/2) \end{bmatrix}. \quad (2.1)$$

The notation introduced in Equation 2.1 highlights the geometric representation of a qubit as a point on a sphere of unit radius, called Bloch sphere (Figure 2.1). The states $|0\rangle$ e $|1\rangle$ correspond to the poles of this surface, while on the equatorial line of the sphere there are four states of fundamental importance:

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad |-\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad |r\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix}, \quad |l\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -i \end{bmatrix}.$$

The states $|+\rangle$ e $|-\rangle$ form the diagonal basis, while the states $|r\rangle$ e $|l\rangle$ form the circular basis.

The state of a qubit can be modified using unitary operators¹. An operator U is said to be unitary if it preserves the scalar product. This means that given any two quantum states $|\psi\rangle$ and $|\phi\rangle$:

$$\langle\psi|\phi\rangle = \langle\psi| U^\dagger U |\phi\rangle.$$

¹In this section we will omit the typical representation of operators with the hat symbol ($\hat{O}$).

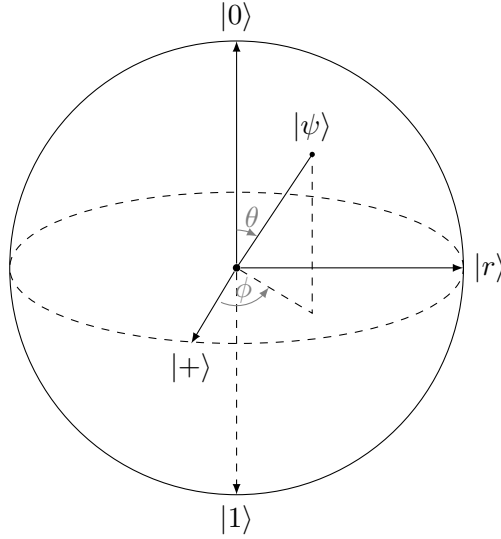


Figure 2.1: Representation of a qubit state as a point on the Bloch sphere.

The previous definition is equivalent to the relation:

$$U^\dagger U = U U^\dagger = I.$$

In quantum computing, Pauli operators σ_x , σ_y and σ_z are of fundamental importance. Their matrix representation, in the computational basis, is:

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

It is easy to verify that the Pauli operators are unitary.

Other important properties of Pauli operators are:

- (a) $[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma_k$,
- (b) $\{\sigma_i, \sigma_j\} = 2\delta_{ij}I$,
- (c) $\sigma_i\sigma_j = i\epsilon_{ijk}\sigma_k + \delta_{ij}I$.

Property (a) shows that Pauli operators don't commute, thus there is no basis in which two or more Pauli operators are simultaneously diagonal. The vectors $|0\rangle$ and $|1\rangle$ are eigenvectors of the σ_z operator with eigenvalues 1 and -1 . Similarly, the diagonal basis and the circular basis are composed of eigenvectors, respectively of the operators σ_x and σ_y . Property (b) shows that different Pauli operators anticommute, while property (c) can be used to combine together Pauli operators. It is also important to notice that from (b) and (c) it follows that:

$$\sigma_x^2 = \sigma_y^2 = \sigma_z^2 = I.$$

A qubit, unlike its classical counterpart, can be in a superposition of states. The fundamental problem, which must be overcome in order to use a quantum computer effectively, lies in the extraction of the information contained in a qubit. The state of a qubit is completely determined

by the θ and ϕ values introduced in Equation 2.1. With a large number of copies of the same qubit, a quantum state can be measured by calculating the average values of the Pauli operators:

$$\begin{aligned}\langle\psi|\sigma_z|\psi\rangle &= \cos(\theta), \\ \langle\psi|\sigma_x|\psi\rangle &= \sin(\theta)\cos(\phi), \\ \langle\psi|\sigma_y|\psi\rangle &= \sin(\theta)\sin(\phi).\end{aligned}$$

With just one copy of the system, the information that can be obtained is very limited. This important concept will be extensively presented in Section 2.1.3. A further complication is given by the no-cloning theorem which states that it is not possible to duplicate an a priori unknown quantum state (Appendix A).

2.1.2 Entanglement

In order to perform a quantum computation it is necessary to use a system composed of many qubits. The state of a system composed of n qubits is described by a vector belonging to the tensor product of the n Hilbert spaces associated with the state of each single qubit. A natural basis of this space can be obtained through a tensor product of the computational basis. Consider, for example, a system composed of two qubits, described by a vector belonging to the space $\mathcal{H}_1 \otimes \mathcal{H}_2$. The computational basis will be given by the set of vectors:

$$|0\rangle_1 \otimes |0\rangle_2, \quad |0\rangle_1 \otimes |1\rangle_2, \quad |1\rangle_1 \otimes |0\rangle_2, \quad |1\rangle_1 \otimes |1\rangle_2.$$

A generic state of the two-qubit system ($|\psi\rangle$) can be written as a superposition of the states of the computational basis:

$$|\psi\rangle = \sum_{i,j=0}^1 c_{ij} |ij\rangle \quad \text{with:} \quad \sum_{i,j=0}^1 |c_{ij}|^2 = 1,$$

where the following notation has been used: $|i\rangle_1 \otimes |j\rangle_2 = |ij\rangle$.

In classical mechanics, a compound system is determined by the state of its parts taken individually. In quantum mechanics, a compound system is described by a single wave function, this characteristic gives rise to the phenomenon of entanglement. Since it is difficult to describe entanglement, the standard definition of this phenomenon is based on defining when a system is not entangled: *the state of a compound system is said to be not entangled if it can be written as the tensor product of the states of its parts*. The state of n qubits, which are not in an entangled state, is determined by $2n$ parameters; it can therefore contain a quantity of information which, as in the classical case, grows linearly with the number of qubits. Considering entangled states the same system is determined by $2(2^n - 1)$ parameters, it thus contains an amount of information that grows exponentially in the number of qubits. This property is at the basis of the advantage of quantum computation with respect to classical computation.

Given the importance of entanglement, it can be useful to show a direct consequence of this

phenomenon called the EPR paradox. Consider the following entangled state, called singlet:

$$|\psi^-\rangle = \frac{1}{\sqrt{2}} (|01\rangle - |10\rangle).$$

Measuring the first qubit, makes instantly obtain information about the second qubit. This feature, which is quite general in entanglement states, produces apparently paradoxical effects that came to question the completeness of quantum mechanics. A physical theory is said to be complete if it takes into account every real observable. An observable is said to be real if its value can be predicted with certainty. In 1935, Einstein, Podolsky and Rosen (EPR) showed that quantum mechanics is incomplete simultaneously accepting the principle of reality and the principle of locality.

- The reality principle states that if the value of an observable can be predicted with certainty, then it is independent from the measurement process.
- The principle of locality states that if two systems are causally disconnected, then a measurement made on one of the two cannot influence a measurement made on the other system.

Consider the state $|\psi^-\rangle$, introduced previously, the first qubit is sent to the observer Alice and the second qubit to the observer Bob, as represented in Figure 2.2.

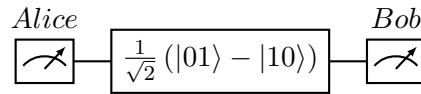


Figure 2.2: Schematic representation of the EPR paradox.

The two observers can choose to measure σ_z or σ_x , the distance between them is such that the measurements are causally disconnected. Assuming that Alice measures σ_z and gets the result $+1$, the system would instantly collapse into the $|01\rangle$ state. Consequently, a measurement of σ_z by Bob would certainly give the result -1 . By the above definition σ_z would be, for Bob, a real observable. However, if Alice chose to measure σ_x , the system after the measurement would collapse into $|+-\rangle$ or $|-+\rangle$. In this case, the real observable for Bob would be σ_x . Since σ_z and σ_x do not commute, these two observables cannot be simultaneously real according to quantum mechanics. This means that Alice's measurement instantly determines which observable is real for Bob, in open violation of the principle of locality.

The current formulation of quantum mechanics rejects the locality principle in its previously introduced strong formulation. This result does not conflict with the principle of causality according to which information cannot be transmitted faster than the speed of light. In the example of the EPR paradox an instantaneous exchange of information is not possible between Alice and Bob. Consider the case in which Alice measures σ_z , the result obtained would be equally likely $+1$ or -1 and the system would move respectively to the state $|01\rangle$ or $|10\rangle$. At this point, if Bob also chooses to measure σ_z he would get the opposite value to Alice's. The probability of getting $+1$, as well as that of getting -1 , would be equal to 50%. On the other hand if Bob chooses to measure σ_x the probability of getting $+1$ or -1 would be 50%, regardless of the value Alice obtained. This follows

from the observation that:

$$|01\rangle = \frac{0}{\sqrt{2}} |1\rangle (|+\rangle - |-\rangle), \quad |10\rangle = \frac{1}{\sqrt{2}} |1\rangle (|+\rangle + |-\rangle).$$

The situation would remain the same if Alice chooses to measure σ_x . In fact Bob would equally likely get $+1$ and -1 from a measurement of σ_x or σ_z . Since Bob is unable to determine the measurement Alice is making, there is no exchange of information.

2.1.3 Measurements

Quantum physics has proven to be a great empirical success. However, as the EPR paradox shows, there are still many philosophical debates in particular regarding the concept of measurements. Measuring a quantum system is the only way in which information can be extracted from it, this is necessary to obtain results at the end of a quantum computation. Two postulates of quantum mechanics define the observables and the measurement process:

1. Given a quantum system described by a vector in the Hilbert space $\mathcal{H}$, any observable is described by a self-adjoint operator on the Hilbert space $\mathcal{H}$. The only possible outcome of a measurement of an observable, represented by the operator A , is one of the eigenvalues of this operator $\{a_i\}$. Given $\{|i\rangle\}$, an orthonormal basis of eigenvectors of A , it is possible to represent the state of the system $|\psi\rangle$ as:

$$|\psi\rangle = \sum_i c_i |i\rangle.$$

With this decomposition, the probability p_i of obtaining the outcome a_i is:

$$p_i = |c_i|^2 = |\langle i|\psi\rangle|^2.$$

2. Consider a system described by the vector state $|\psi\rangle$. After a measurement of the observable A has been carried out on this state, obtaining the result a_i , the system collapses in the state:

$$\frac{P_i |\psi\rangle}{\sqrt{\langle\psi| P_i |\psi\rangle}},$$

where P_i is the projection operator over the subspace corresponding to the eigenvalue a_i .

The first postulate defines the mathematical formalism necessary to describe a measurement. The second postulate shows that a measurement is intrinsically a perturbative event, as the state is irreversibly modified. The second postulate also introduces the projection operators. A projection operator P is characterized by the fundamental property that $P^2 = P$. Since the eigenvectors of any operator A constitute an orthonormal basis for the Hilbert space associated with the system, the projectors $\{P_i\}$ satisfy the completeness relation (a) and the orthogonality condition (b):

$$(a) \quad \sum_i P_i = I,$$

$$(b) \quad P_n P_m = \delta_{nm} P_n.$$

With this formalism it is possible to decompose the operator A as:

$$A = \sum_i a_i P_i. \quad (2.2)$$

In the framework of quantum computing we usually perform measurements in the computational basis. For a single qubit we can identify the measurement operator with the Pauli σ_z operator. Using Equation 2.2 we can decompose this operator into projectors as:

$$\sigma_z = (+1) \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + (-1) \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

2.1.4 Quantum circuits

In order to implement a quantum computation it is necessary to modify, in a controlled way, the state of a system composed of one or more qubits. The transformation of a quantum system is carried out by unitary operators, therefore all the operations performed by a quantum computer are invertible. This feature shows a substantial difference with respect to classical computation. The operations on the qubits can be represented using the circuit model. Each qubit is represented by a horizontal line and the initial quantum state is modified by the sequential application of gates (unitary operators), an example is reported in Figure 2.3. At the end of the quantum circuit, information can be extracted with measurement gates that represent a measurement apparatus.

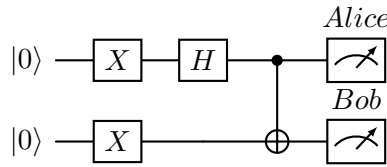


Figure 2.3: Circuit model representation of a quantum circuit composed of two qubits for the EPR paradox (Section 2.1.2). This circuit allows to generate entangled states using single and multi-qubit gates. At the end of the circuit measurement gates are used to indicate a measurement apparatus.

Single-qubit gates

An operator, acting on the state of a qubit, is represented by a unitary matrix with complex coefficients of dimensions 2×2 , defined up to a global phase. A transformation corresponds to a rotation on the Bloch sphere of the vector representing the state of the system. In Section 2.1.1 the Pauli operators σ_x , σ_y and σ_z have already been introduced. In the formalism of quantum circuits, their corresponding gates are referred as X, Y and Z gates. Pauli X gate is also called the NOT gate (circuit symbol: $\oplus$), it inverts the states of the computational basis with the transformations:

$$|0\rangle \rightarrow |1\rangle \quad \text{and} \quad |1\rangle \rightarrow |0\rangle.$$

This gate does not represent a universal NOT capable of performing the transformation $|\psi\rangle \rightarrow |\psi^\perp\rangle$. For example, the states $|+\rangle$ and $|-\rangle$ are not modified.

Another gate of fundamental importance is the Hadamard gate:

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

This gate allows to switch from the computational basis to the diagonal basis by carrying out the transformations: $|0\rangle \rightarrow |+\rangle$ and $|1\rangle \rightarrow |-\rangle$.

A general single-qubit transformation is a rotation of the vector, representing its state, on the Bloch sphere. An useful representation of a rotation exploits the Pauli operators:

$$R_z(\delta) := e^{-i\frac{\delta}{2}\sigma_z}, \quad R_x(\delta) := e^{-i\frac{\delta}{2}\sigma_x}, \quad R_y(\delta) := e^{-i\frac{\delta}{2}\sigma_y}.$$

Where the operators $R_z(\delta)$, $R_x(\delta)$ and $R_y(\delta)$ implement a rotation equal to an angle δ , respectively around the axes z , x and y . A generic rotation around the $\hat{\eta}$ versor is obtained with the operator:

$$R_{\eta}(\delta) := e^{-i\frac{\delta}{2}(\eta_x\sigma_x + \eta_y\sigma_y + \eta_z\sigma_z)} = \cos\left(\frac{\delta}{2}\right)I - i\sin\left(\frac{\delta}{2}\right)(\eta_x\sigma_x + \eta_y\sigma_y + \eta_z\sigma_z).$$

Each rotation of the qubit can be obtained by appropriately combining the three Pauli operators plus the identity I . Moreover, this shows that, the set composed of the Pauli matrices plus the identity matrix constitutes a basis of the space of 2×2 unitary Hermitian matrices. The Hadamard gate, for example, can be written as:

$$H = \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z).$$

In the framework of quantum computing it is useful to express a general single-qubit transformation in terms of its Euler angles. This representation is known as U_3 gate:

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}.$$

It is possible to execute any U_3 gate, up to a global phase, as a combination of R_x and R_z rotations [60]:

$$U(\theta, \phi, \lambda) = R_z\left(\phi - \frac{\pi}{2}\right) R_x\left(\frac{\pi}{2}\right) R_z(\pi - \theta) R_x\left(\frac{\pi}{2}\right) R_z\left(\lambda - \frac{\pi}{2}\right). \quad (2.3)$$

This decomposition is very useful to implement a general single-qubit gate on hardware (Section 2.2.4).

Multi-qubit gates

Multi-qubit gates allow to perform Boolean logic operations and generate entanglement states. Two-qubit gates can be represented by unitary matrices 4×4 . For the computational basis the following notation will be used:

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

One of the most important two-qubit gates is the C-NOT gate (Controlled-NOT). In the computational basis, it inverts the second qubit (target) only if the first qubit (control) is in the state $|1\rangle$. The circuit symbol and matrix representation of this gate is:

$$\text{C-NOT} = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} \oplus \text{---} \end{array} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The behavior of the C-NOT gate depends on the basis used, the roles of the control and target qubits are reversed in the diagonal basis. The C-NOT gate allows to generate entanglement states between two initially uncorrelated qubits, for example:

$$\text{C-NOT}(\alpha|0\rangle + \beta|1\rangle)|0\rangle = \alpha|00\rangle + \beta|11\rangle.$$

It is possible to implement any multi-qubit transformation using only single-qubit operators and C-NOT gates [61]. Thus the set composed of single qubit Pauli gates and the C-NOT gate is a universal set of gates for quantum computing.

Other important two-qubit gates are the SWAP and the CZ (Controlled-Z). Their circuit symbol and matrix representation is:

$$\text{SWAP} = \begin{array}{c} \text{---} \times \text{---} \\ | \\ \text{---} \times \text{---} \end{array} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{CZ} = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} \boxed{Z} \text{---} \end{array} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

The SWAP gate swaps the state of two qubits: $|ab\rangle \rightarrow |ba\rangle$. We will see its application for the hardware qubits connectivity matching (Section 2.2.4). The SWAP gate can be decomposed into three C-NOT gates as shown in Figure 2.4.

The CZ gate applies a Z gate on the target qubit if the control qubit is in the state $|1\rangle$. The CZ gate can be decomposed into a C-NOT and two H gates as shown in Figure 2.4.

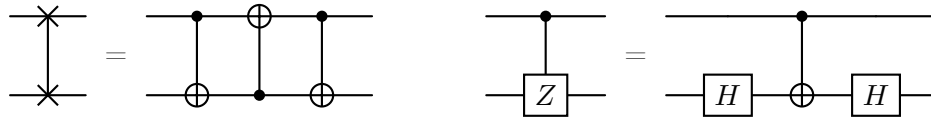


Figure 2.4: Gate decomposition into C-NOT and single-qubit gates of the SWAP gate (left) and the CZ gate (right).

Another important gate, in particular for the hardware implementation (Section 2.2.4), is the i-SWAP that acts as a swap but also adds a phase on the swapped states.

$$\text{i-SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.4)$$

It is possible to extend the idea of controlled gates, like C-NOT and CZ, to any single-qubit gate. For example, a Controlled- U gate (C- U) applies the unitary operator U to the target qubit if the control qubit is in the state $|1\rangle$.

Among the gates acting on more than two qubits, the Toffoli gate has a particular importance; its circuit symbol and a possible decomposition into C-NOT and single-qubit gates is reported in Figure 2.5. This three-qubit gate acts as a NOT on the target qubit when the two control qubits are in the $|1\rangle$ state. Toffoli gate is important because it is a universal reversible logic gate, which means that any reversible circuit can be constructed from Toffoli gates.

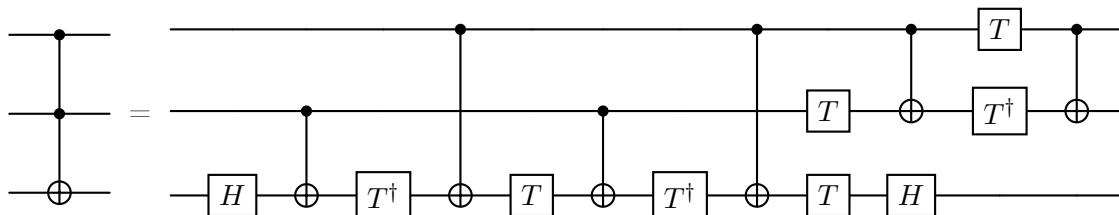


Figure 2.5: Circuit symbol for the Toffoli gate (left) and its decomposition into single-qubit gates and C-NOT gates. The operator T is defined as $T := R_z(\pi/4)$.

2.1.5 Quantum advantage

Quantum computing may provide a speedup compared to classical computing. To formalize this hypothesis, known as quantum advantage, we need to introduce the concept of computational complexity. This term refers to the resources required to solve a problem, particularly time or the number of operations. The Turing machine, an abstract model capable of manipulating memory based on a set of rules, is often used to define the number of operations needed to solve a problem. For more details, refer to [62]. Now, let's define the different computational complexity classes. A problem belongs to the Polynomial class (P) if the resolution time grows at most as a power of the input bit size. We say that a problem in this class can be solved efficiently. For example, addition requires a number of operations proportional to the digits of the input numbers. The Non-Polynomial class (NP) consists of all problems whose solution verification can be performed in polynomial time. Since no assumptions are made about the resolution time, P is included in NP. To date, the most accredited conjecture states that the inclusion just defined is strict. An example confirming this hypothesis is still being sought, although a good candidate seems to be the factorization problem. The best-known classical algorithm to solve this problem requires execution time proportional to $O(\exp(n^{1/3} \log(n)^{2/3}))$ where n are the digits of the number to be factorized [63]. The solution verification requires some divisions, so it can be performed in polynomial time. If it were demonstrated that this algorithm cannot be made more efficient, we would have found a problem belonging to NP but not to P.

A problem belonging to NP is called NP-complete (NPC) if every problem belonging to NP can be reduced to it in polynomial time. The most famous example of such a problem concerns the search for the shortest path to visit all vertices of a graph given the distances between them. By definition, the NPC class is included in NP, but it has not yet been proven that this inclusion is strict. We also introduce a class of problems solvable using a probabilistic Turing machine, that means a Turing machine with a perfect random number generator. A problem is defined decisional if

it has a discrete number of possible outputs, with only one being correct. The BPP (Bounded-error Probabilistic Polynomial) class consists of all decisional problems for which a probabilistic Turing machine can provide a solution, with probability of $1/2 + \delta$, where δ is an arbitrarily small positive parameter. The inclusion of P in BPP follows from the fact that a probabilistic Turing machine is an extension of the deterministic counterpart. The most accredited conjecture currently suggests that $P=BPP$. The inclusion relation between NP and BPP remains unknown.

Quantum computation requires the introduction of a new computational complexity class. We define the BQP (Bounded-error Quantum Polynomial) class as the quantum equivalent of BPP for a Turing machine. Currently, it is conjectured that there is a strict inclusion of BPP in BQP. This hypothesis challenges the strong Church-Turing thesis, which states that a probabilistic Turing machine can simulate any computational model with at most a polynomial increase in the number of operations. In other words, according to this thesis, a problem not belonging to BPP cannot be efficiently solved using a different computational system. The factorization problem may be a good candidate to demonstrate the strict inclusion of BPP in BQP. Shor's algorithm can potentially solve this problem with a probability arbitrarily close to one, in polynomial time. If it were proven that efficient factorization using a probabilistic Turing machine is impossible, the strong Church-Turing thesis would be invalidated. Another unresolved issue is the relationship between the BQP and NP classes. Figure 2.6 illustrates the conjectured inclusion relations among the computational complexity classes discussed earlier. If these relations were proven, the algorithms that could benefit from quantum advantage would be those in the BQP class but not in the BPP class.

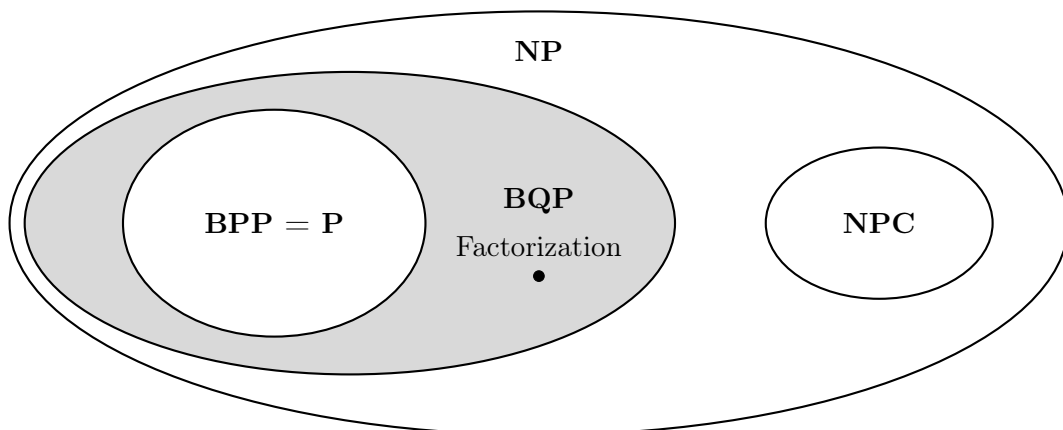


Figure 2.6: Inclusion relations between computational complexity classes (P=Polynomial, NP=Non-Polynomial, NPC=Non-Polynomial Complete, BPP=Bounded Probabilistic Polynomial, BQP=Bounded Quantum Polynomial) based on the current most accredited conjectures. The set of algorithms where it may be possible to obtain quantum advantage has been highlighted, the factorization algorithm is included in this set.

2.2 Hardware

This section offers a concise overview of the working principles of quantum hardware, with the main focus on superconducting qubits. Given the complexity of physically realizing a quantum computer, this discussion is not exhaustive. It merely aims to provide insights into state-of-the-art devices and the current technical challenges to overcome limitations. Section 2.2.1 introduces the main requirements that a quantum computer has to satisfy. Section 2.2.2 and Section 2.2.3 reviews the use of photons and trapped ions as physical systems for quantum computation. Section 2.2.4 explores the working principle of superconducting qubits for quantum computing.

2.2.1 Fundamental requirements

The creation of a universal quantum computer necessitates a physical quantum system and the tools to control and measure it. DiVincenzo [64] clearly outlined five fundamental requirements for such a system, which this section summarizes in light of recent advancements.

1. A scalable physical system with well-characterized qubits.
2. The ability to initialize the state of the qubits to a fiducial state.
3. Long relevant decoherence times, much longer than the gate-operation time.
4. A universal set of quantum gates.
5. A qubit-specific measurement capability.

The first requirement is a scalable physical system with well-characterized qubits. Any two-level quantum system can serve as the basic unit of quantum computation. In theory, quantum computation can be implemented using qudits, quantum systems with more than two levels. However, these systems can pose additional challenges in characterization and noise sensitivity, restricting their use primarily to quantum optics. Characterization implies knowing all the physical parameters of the system, including its internal Hamiltonian, interactions with other qubits, and couplings to external fields. Scalability is currently the main challenge. For instance, state-of-the-art quantum devices, like superconducting chips, can control around 10^2 qubits. While this is sufficient for preliminary results, it is still orders of magnitude less than what is needed for quantum supremacy.

The second requirement for a universal quantum computer is the ability to initialize qubits to a specific state. This can be achieved by resetting a qubit to the ground state and then applying a sequence of known operations to reach the desired initial state. While time-efficiency is not crucial for qubit initialization at the start of computation, it becomes significant for quantum error correction, which requires continuous initialization of ancillary qubits (Section 2.3.4). Resetting to the ground state can be accomplished using two main strategies. The first is to cool the system and wait for it to naturally reach the ground state. This method, while not perturbing other qubits, has a long thermal relaxation time compared to average gate-operation times. The second strategy involves performing a projective measurement to collapse the state, followed by gate operations to reach the ground state. This method is more time-efficient but risks perturbing the rest of the system.

The third key requirement is a system with decoherence times significantly longer than gate operation times. It's crucial to note that we're considering the decoherence times of the degrees of freedom that encode the quantum information. When a quantum system interacts with the environment, superposition states decay to mixed states (as detailed in Sections 2.3.2 and 2.3.3). This process causes qubits to behave like classical bits, eliminating any potential quantum enhancement. Therefore, to achieve quantum advantage, qubits must have decoherence times longer than the total computation time. While this may seem daunting for lengthy quantum algorithms, it would be sufficient to reach decoherence times on the order of $10^4 - 10^5$ gate operation times to benefit from quantum error correction. Although this goal has not yet been achieved, recent years have seen promising initial results [65–67]. For instance, in superconducting qubits, typical decoherence times range from tens to hundreds of microseconds [68, 69], while gate operation times are on the order of tens of nanoseconds for single-qubit gates and hundreds of nanoseconds for two-qubit gates [70].

The fourth requirement for a universal quantum computer is a universal set of quantum gates, also known as native gates. This means that any unitary transformation must be decomposable into a sequence of unitary operations that the system can perform. Typically, the universal gate set comprises a general single-qubit transformation and a two-qubit interaction that can generate entanglement. Performing a general single-qubit transformation is usually straightforward, as these transformations can be decomposed into three rotations on the Bloch sphere's axis (Section 2.1.4). However, finding a process that can entangle two qubits in a controlled manner can be challenging. Fortunately, implementing a C-NOT gate (or an equivalent like a CZ gate or i-SWAP gate) is sufficient to complete a universal gate set. Implementing a quantum gate introduces a system perturbation that can increase decoherence, leading to errors. Therefore, to benefit from quantum error correction, it's necessary to achieve a gate error rate below $10^{-4} - 10^{-5}$.

The fifth requirement for a universal quantum computer is a measurement apparatus. It's crucial that each single-qubit measurement is independent of the status of other qubits and does not alter them. Measurement results are always subject to errors. To benefit from quantum error correction, it's important to keep this error value below a certain threshold. Fortunately, reducing the measurement error is possible as qubit values in a specific basis can be copied. New error mitigation techniques, such as the readout mitigation matrix [71–74], can help reduce the measurement error. The time required to perform the measurement can become a significant factor, especially when intermediate measurements are needed for quantum error correction.

The five requirements outlined are essential for stable execution of quantum algorithms and achieving quantum supremacy. If this goal is ever reached, direct exchange of quantum information will become important. This would necessitate a quantum computer's ability to copy information onto flying qubits, with photons being the best natural candidates. Even if not fundamental for universal quantum computing, the inter-qubit communication is considered the sixth (optional) DiVincenzo criteria. Another requirement, not fundamental for universal quantum computing but that is directly related to all the others, is quantum error correction, for this reason this is considered the seventh (optional) criteria.

Several physical systems are being explored as potential candidates for quantum computing. However, currently, no physical system successfully meets all five fundamental requirements. The best choice of the system depends on the specific requirements of the quantum computation task. Table 2.1 shows the feasibility of DiVincenzo criteria for some of the most famous hardware platforms.

Hardware	1	2	3	4	5	6	7
Photons	M	M	H	L	H	H	L
Trapped Ions	M	H	H	H	H	M	M
Neutral Atoms	M	H	M	M	M	M	M
Superconducting	H	H	M	H	H	L	M
Quantum Dots	M	M	L	M	M	L	L

Table 2.1: Feasibility of DiVincenzo criteria (five fundamental criteria plus two optional) across different quantum computing hardware. Criteria: (1) Qubit scalability, (2) Initialization, (3) Coherence time, (4) Universal gates, (5) Qubit measurement, (6) Inter-qubit communication, (7) Error correction. Each criterion is marked as high (H), medium (M), or low (L) based on the feasibility for each hardware type, these labels are indicative and intended to provide a general sense of the strengths and weaknesses of each hardware platform.

The next paragraphs will introduce the working principle of trapped ions, photons and superconducting qubits with a brief review of their advantages and current challenges. These systems are not the only ones being studied for quantum computation, other options are, for example, topological qubits [75], quantum dots [76] or neutral atoms [77]. As stated before, in this work we will not consider quantum annealers or other approaches to quantum computing like measurement-based quantum computing.

While well-known approaches to physical quantum computing continue to evolve, new potential candidates emerge annually. It’s likely that the technology needed for a stable universal quantum computer hasn’t been discovered yet. This open challenge fuels excitement in the quantum computing community and drives rapid progress in the field.

2.2.2 Photons

Photons are good candidates for quantum computing because they do not suffer decoherence problems, they can be manipulated at room temperature without the need to shield them from the external environment. Moreover, single photon operations can be performed easily and with a low error rate. The main limitations comes from the lack of a deterministic two photon interaction that can produce a universal two-qubit gate. Imperfect photon detection, scattering and absorption contribute to high losses during the computation.

Photonic quantum computing has gained interest for the recent claims of quantum advantage [51–54] achieved for the boson sampling problem [78]. Even if these claims have not been universally accepted by the scientific community [79, 80] they demonstrate the potential of this quantum computing framework and its possible applications in the near future. Moreover, quantum photonics is fundamental for other applications like cryptography with the famous quantum key distribution protocol [81].

Logical operations

In a photonic qubit, information is stored in the probability amplitudes corresponding to two modes of an optical field’s degree of freedom. A typical example is employing orthogonal polarization, in

this framework the states $|0\rangle$ and $|1\rangle$ are represented with a vertically or horizontally polarized photon respectively. However, it is possible to use other discrete or even continuous degrees of freedom such as non-overlapping propagation paths, frequency or temporal bin. Almost all these degrees of freedom allows more than two modes and can be easily employed to encode qudits. Moreover, it is even possible to use several degrees of freedom of the same photon to encode different qubits/qudits [82, 83]. Single-qubit operations requires moving the amplitudes of the photon between the different modes that encode the information. This can be done with high precision using the well established methods developed for interferometry. For example, in the polarization state encoding, a half-wave plate is a simple optical device that changes ortogonally polarized light into vertically polarized light, thus implementing an X Pauli gate.

The realization of two-qubit gates is trickier as two photons do not directly interact. In order to obtain an interaction it is necessarily to employ a medium, usually a material with a non-linear optical response. For example, in the polarization framework, a C-NOT gate could be implemented by applying a π phase shift to the target photon depending on the polarization of the control photon. Similar kind of gates have been proposed and are currently being studied [84, 85], unfortunately non-linear optical interactions of this kind are weak, thus resulting in a low gate success rate.

An alternative to non-linear optics comes from the possibility of constructing deterministic two-qubit gates using linear optics elements and measurements with post selection [86]. The drawback of this framework is that it requires to execute multiple times a non-deterministic gate in order to obtain a deterministic gate, thus requiring a big overhead of ancillary qubits [87].

An advantage of photonic quantum computing is that the hardware can be easily integrated in small dimension silicon-based optical chips [88]. The possibility of manipulating photons without the need of bulky optical components is fundamental to scale to a number of photons and gates sufficient for real world applications.

Measurements

To perform measurements, in the framework of photonic quantum computing, it is necessary use a photon detector. A photon detector is characterized by different parameters, among them it is worth citing the detection efficiency, detection rate, noise levels and photon resolution number. The detection efficiency (η_d) is the fraction of correctly revealed photons over the total number of photons that hit the detector: $\eta_d = \text{detected_photons} / \text{total_photons}$. This value is fundamental to allow a computation with a large number of photons. For example, the avalanche photodiode, one of the most famous photon detectors, has a detection efficiency that reaches only 0.65 [89]. This makes the probability of simultaneously detecting more than 10 photons drop below 2%. With the recent advent of superconducting photon detectors [90] it is possible to obtain values of η_d over 0.95 [91]. This technical achievement allows the simultaneous use of many photons in the same experiment, with the drawback of requiring cryogenic temperatures.

The detection rate (r_d) is the number of photons that a detector can register in a unit of time. This value is mainly determined by the reset time (τ_r) and the time jitter (τ_j):

$$r_d = \frac{1}{\tau_r + \tau_j},$$

where τ_r refers to the time needed by the detector to be ready to detect a photon after having detected one. τ_j is the time needed by the detector to respond to a detection event, this parameter also defines the time resolution accuracy. State-of-the-art detectors like the avalanche photodiode and the superconducting photon detectors have a detection rate of $10^7 - 10^8 s^{-1}$.

The noise level of a detector is mainly characterized by the dark counts rate (C_d) that is the number of false counts registered per unit time. For an avalanche photodiode this value is relatively small ($C_d \simeq 100 s^{-1}$), while superconducting detectors suffer the thermal noise that can enter from the input fiber and require spectral filtering.

Finally it is important to consider that for many applications it is fundamental to have detectors with photon number resolution. For this purpose transition-edge sensors [92] are being studied. This technology allows to know the number of incident photons in the same time interval up to about 20 photons with $\eta_d = 0.98$ [93] with the drawback of a low detection rate $r_d \simeq 10^6 s^{-1}$.

Generation

To perform quantum computation with photons, it is essential to simultaneously generate a large number of photons with specific characteristics. The generated photons must be in pure states and indistinguishable. Additionally, their frequency and mode must be compatible with fiber optics or the materials of integrated circuits to minimize losses. The most well-known process for generating single photons is spontaneous parametric downconversion [94] (Figure 2.7). In this process, an incoming photon (which can be generated with a laser) interacts with a nonlinear crystal. During this interaction, there is a small probability that the photon will be converted into two lower-energy photons (idler and signal). These photons are entangled due to conservation laws. By observing one of these two photons (idler), the generation of the second photon (signal) is ensured.

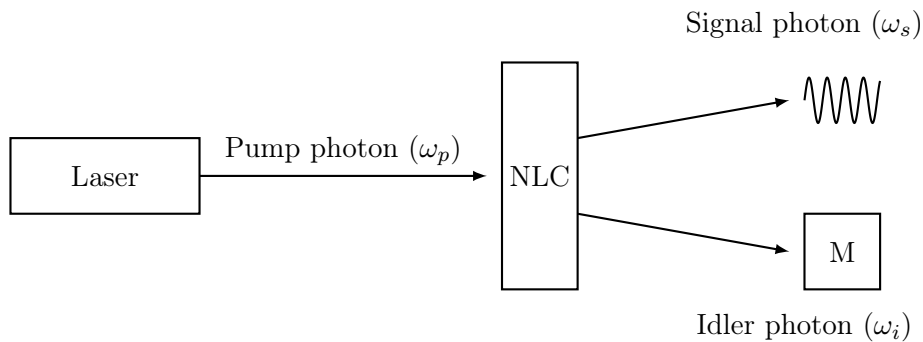


Figure 2.7: Schematic representation of parametric downconversion: a pump photon with angular frequency ω_p enters the nonlinear crystal (NLC) and generates two lower-energy photons, the signal photon (with angular frequency ω_s) and the idler photon (with angular frequency ω_i), where $\omega_p = \omega_s + \omega_i$. By measuring the idler photon (M), the generation of the signal photon is ensured.

There are two main problems related to parametric downconversion for photon generation. The first problem is the poor coupling with fiber optics; the spatial mode of the generated photons differs significantly from the single Gaussian mode of optical fibers, and the wavelength of the generated photons (approximately 800 nm) leads to high losses. The second problem is residual entanglement. The two generated photons are entangled in frequency, so measuring the idler photon without resolving its frequency results in a non-pure state for the signal photon. Solutions to these problems

are being studied, such as quasi-phase matching [95] and group velocity matching [96]. With these advancements, it is possible to generate single heralded photons from the parametric downconversion process with an efficiency greater than 80% [97] and state purities of the signal greater than 0.99 [98].

Parametric downconversion allows the production of high-quality single photon sources. However, this process is non-deterministic and occurs with low probability. To increase the probability of success, multiple non-deterministic sources can be combined. When an idler photon is observed, the corresponding signal photon can be routed to the output using active optical circuits. It is even possible to create a quasi-deterministic photon source by employing only one non-deterministic source [99, 100]. In this case, the generated photons are stored in an optical memory and released when necessary. The introduction of a quantum optical memory is a relevant research topic that is leading to many possible solutions. Typical strategies involve the storage of photons in free space or waveguide loops, but there are also studies on the realization of optical memories based on matter systems [101]. It is finally worth mentioning the possibility of using quantum dots as deterministic photon sources [102].

The technological advancements reported in this section demonstrate that photonic quantum computing is among the most promising frameworks for short-term quantum computing. However, the challenge in achieving deterministic two-qubit gates prevents it from being a viable candidate for universal quantum computing. Consequently, optical quantum chips are likely to be primarily utilized for specific tasks that enhance classical hardware or for quantum communications [103, 104].

2.2.3 Trapped ions

Trapped ions are the oldest framework developed for universal quantum computing, the first two-qubit gate was implemented using trapped ions in 1995 [105]. This technology satisfies all of DiVincenzo's criteria. The main advantages of trapped ions include long decoherence times, high-fidelity gates, and precise measurements. Moreover, as all ions of the same type and isotope are identical, there is no need for individual characterization routines required for fabricated qubits.

Scalability is the most significant challenge for this technology. Controlling and measuring systems with hundreds of trapped atoms has been demonstrated [106], however, leading companies like IonQ and Quantinuum have only managed to produce universal quantum computers with around 30 qubits [107] [108]. Another limitation of this technology is the long gate execution times. Even if these times are short compared to decoherence times, they may still pose challenges in achieving quantum supremacy.

The technologies developed for trapped ion quantum computing shares some similarities with neutral atoms quantum computing. While less mature, neutral atom quantum computing has shown promising results, in particular regarding its scalability [77].

Trapping

An ion, or an array of ions, can be confined using electric or magnetic fields. In 1989, W. Paul was awarded with the Nobel Prize for developing the Paul trap, a system capable of confining ions for long periods using electric fields. According to the Laplace equation, a static electric field cannot

produce local minima, which means that trapping an ion requires a non-static electric field. The most well-known Paul trap configuration, also known as a linear trap, is composed of four parallel rods arranged at the vertices of a square (Figure 2.8). The charges on diametrically opposite rods oscillate, typically in the radio frequency range. This system generates a pseudopotential with an equilibrium region along the axis parallel to the rods, at the center of the square. To trap ions along this axis, a static electric field is applied to create a harmonic potential. The advantage of a linear trap is that it allows the confinement of an array of ions, also known as an ion register, along the axis parallel to the rods. By adjusting the static electric field, it is even possible to move the position of the ions within the register.

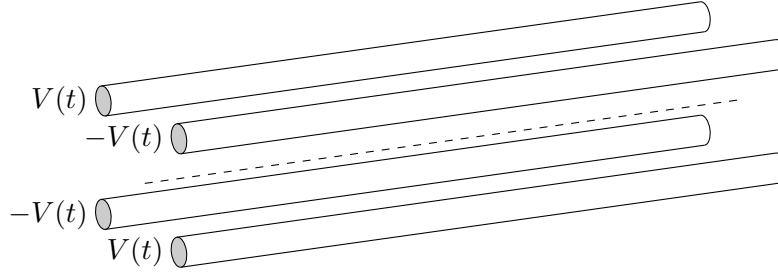


Figure 2.8: Schematic representation of a linear Paul trap. Four parallel rods are arranged at the vertices of a square, the potential applied to the rods oscillates in the radio frequency range. The equilibrium region created by the pseudopotential is represented by the dashed line.

Alternatively to the linear trap, the electrodes can be arranged to create an oscillating electric field with a single equilibrium point. This configuration, known as a point trap, is simpler to construct, more compact, and usually provides better confinement. However, scaling to longer arrays of ions is more difficult, as each trap can confine only a single ion.

The motion of an ion in a Paul trap follows the Mathieu Equation [109]. Without entering into mathematical details, this motion can be decomposed into a slow secular motion and a fast micromotion. The secular motion is a harmonic motion due to the quasi-parabolic electric field averaged over time, with a frequency lower than the trap's radio frequency. This motion is fundamental for implementing two-qubit gates, as will be described in the next paragraph. On the other hand, the fast micromotion has the same period as the driving frequency of the trap, with its amplitude increasing with the distance from the center of the trap. The micromotion must be carefully controlled to avoid problems such as ion overheating.

State-of-the-art ion traps have achieved trapping times on the order of hours, days, and even weeks [110, 111]. Ion losses are mainly due to collisions with residual atoms in the vacuum chamber and interactions with external fields. Various techniques are being developed to improve trapping times, such as laser cooling [112] and active stabilization to reduce environmental noise [113]. Trapping times are generally long enough to perform quantum computations with a limited number of ions without significant losses. However, scaling to thousands of ions would make the ion loss problem significant, thus requiring a mechanism to reload lost ions quickly without disturbing the rest of the system [114].

To load atoms into a trap, they must first be ionized using techniques like electron bombardment [115], which lacks isotopic selectivity, or the more advanced resonance-enhanced photoionization [116]. The ions are then cooled using techniques such as Doppler laser cooling [112]. Once cooled, the atoms can be loaded into the trap using the so-called backside loading technique, which

employs a hole in the trap chip [114]. This method has scalability issues, so it is likely that large ion quantum computers will require a loading zone separate from the computation zone.

Logical operations

In trapped ion qubits, the states $|0\rangle$ and $|1\rangle$ correspond to two different electronic energy levels. Single-qubit gates can be implemented by inducing a direct or Raman transition [117] between these levels. The choice of energy levels gives rise to four fundamental types of qubits: optical qubits, fine qubits, hyperfine qubits, and Zeeman qubits (Figure 2.9).

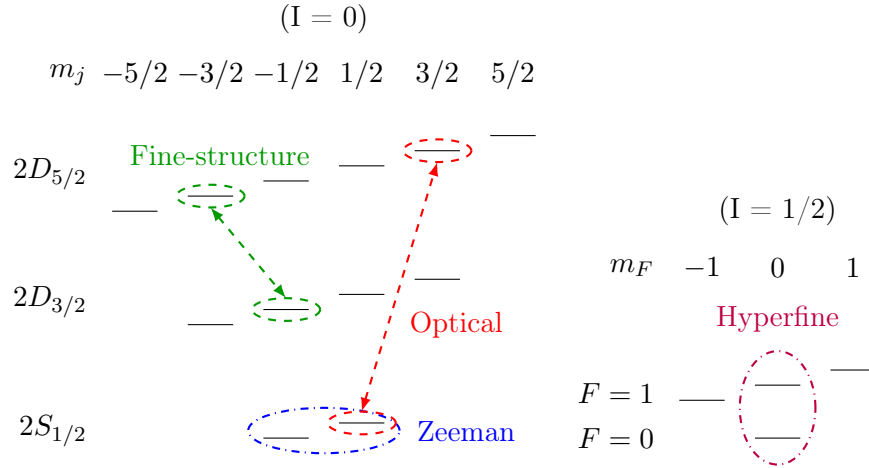


Figure 2.9: Representation of commonly used orbital energy levels in different kinds of trapped ions qubits. On the left, structure of an ion with no nuclear spin angular momentum ($I = 0$) and an external magnetic field (Zeeman splitting). In this kind of ions it is possible to implement optical, fine structure and Zeeman qubits. On the right, structure of an ion with nuclear spin angular momentum ($I = 1/2$) and external magnetic field. This kind of ions is used to implement hyperfine qubits.

In optical qubits, the energy levels are separated by an optical transition on the order of 100THz. A typical example is the ion $^{40}\text{Ca}^+$, where the ground state $|0\rangle$ corresponds to the valence electron in the $S_{1/2}$ orbital, while the excited state $|1\rangle$ corresponds to the $D_{5/2}$ orbital. The advantage of optical qubits is that they can be manipulated using lasers, allowing for faster gate times and precise state detection. Single-qubit gates are performed using direct optical transitions. State-of-the-art results have achieved gate times on the order of microseconds with fidelities up to 99.995% [118]. The disadvantage of this type of qubit is that it has coherence times on the order of seconds [119] and is highly sensitive to laser fluctuations, requiring extremely stable control systems.

In fine qubits, the qubit states are fine structure levels, separated by a transition on the order of 10THz. The characteristics of these qubits are similar to optical qubits, with qubit control typically performed using lasers to induce Raman transitions.

In hyperfine qubits, the qubit states are hyperfine structure levels, separated by a transition on the order of GHz. These qubits have the advantage of exceptionally long coherence times, up to 600s in $^{171}\text{Yb}^+$ [120]. The main disadvantage is their relatively long gate times, and the more complex level structure of odd-isotope ions with hyperfine splitting, which requires additional control systems for state preparation and measurement. Single-qubit gates can be implemented via direct transitions using microwave or radio-frequency pulses, or through Raman transitions using lasers. State-of-the-art experiments have achieved 99.9999% gate fidelity in $12\mu\text{s}$ [121].

In Zeeman qubits, an external magnetic field is applied to split an energy level into two sublevels, typically separated by a transition on the order of 10MHz. The advantages of this qubit type include nearly infinite excited state lifetimes and a simple level structure. However, the main drawback is their sensitivity to magnetic field fluctuations. State-of-the-art experiments have demonstrated coherence times of up to 2s [122], with the primary limitation arising from magnetic shielding and permanent magnets. Single-qubit gates, implemented through direct RF transitions or Raman transitions using lasers, have achieved a fidelity of 99.9% in $8\mu\text{s}$ [123].

The details on how it is possible to implement single-qubit rotations using an external field can be found in Section 2.2.4, in fact trapped ions and superconductive qubits shares many similarities, in particular regarding the way single-qubit gates are implemented.

The key mechanism for implementing a two-qubit gate between trapped ions involves coupling the qubits' internal states with their motional states in the ion trap, allowing them to interact through the vibrational modes of the ion chain. To understand this process, it is important to recall that the ions, to a first approximation, are trapped in a harmonic potential, undergoing harmonic motion. Several schemes have been proposed to implement two-qubit gates, with the Cirac and Zoller gate [124], originally proposed in 1995, requiring all atoms in the trap to be in the ground state. A more feasible approach was later proposed by Mølmer and Sørensen [125]. In the Mølmer-Sørensen (MS) gate, to couple the qubit states and motion, two lasers are used at the frequencies $\omega_0 + \omega$ and $\omega_0 - \omega$, where ω_0 is the frequency of the qubit state transition, and ω is the frequency of the ion's harmonic motion.

The laser at frequency $\omega_0 + \omega$ changes the internal state of the qubit and raises the energy level of the harmonic motion. Conversely, the laser at frequency $\omega_0 - \omega$ changes the internal state of the qubit and lowers the energy level of the harmonic motion (if it is not in the ground state). The combined application of these two lasers generates a spin-dependent force that acts differently based on the internal state of the ions, entangling the internal states of the ions with the motion of the chain. This process creates a time-dependent interaction that typically results in a rotation over the XX-YY axis: $\exp\left(-i\frac{\theta}{4}(\hat{\sigma}_x \otimes \hat{\sigma}_x + \hat{\sigma}_y \otimes \hat{\sigma}_y)\right)$. In the computational basis:

$$\text{MS}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.5)$$

For $\theta = \pi$ the MS is a fully entangling gate.

State-of-the-art implementations of the MS gate have reached a fidelity of 99.6% for optical qubits [126] and 99.91% for hyperfine qubits in $30\mu\text{s}$ [127].

Measurements

Qubit state detection in trapped-ion quantum computing is primarily performed through state-dependent fluorescence (Figure 2.10). A laser is tuned to induce a transition that excites only one of the qubit states. If the ion to be measured is in this state, it will cyclically absorb a photon from the incoming laser and re-emit it shortly after. The photons scattered from the ion can then be easily detected, making the ion appear bright. On the other hand, if the qubit is in the non-resonant

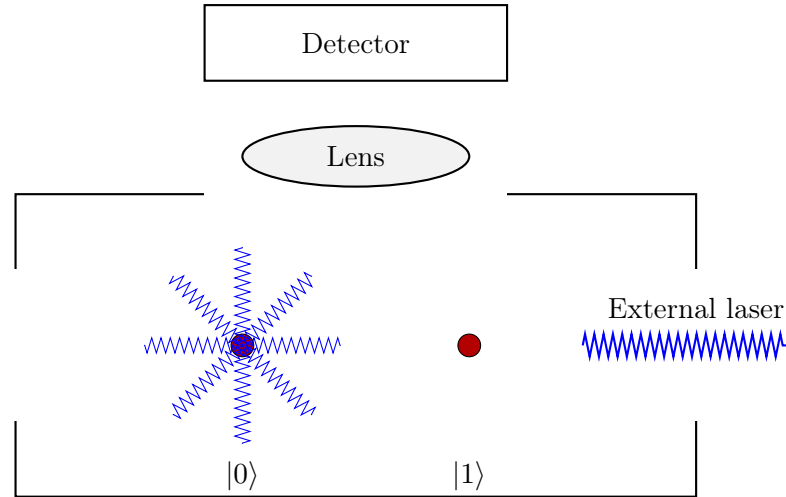


Figure 2.10: Schematic representation of state-dependent fluorescence for the measurement of trapped ion qubits. A laser induces a transition only if the ion is in a specific state (ground state in this example), fluorescence light is collected to determine the qubit state.

state, no photons are scattered, and the atom appears dark. This relatively simple mechanism enables projective, non-demolition measurements with high accuracy. State-of-the-art experiments have achieved a readout fidelity of 0.9991 in $46\mu\text{s}$ using integrated photon detectors [128], which is fundamental for scaling to larger systems. Reducing measurement time is a key challenge. Recently, measurements have been performed in $11\mu\text{s}$ with a fidelity of 0.9993 [129].

A significant advantage of state-dependent fluorescence measurements is that they, in principle, allow for the simultaneous measurement of any number of ions in the trap [130], provided the photon detection system can resolve the individual ions in the chain. However, as the number of atoms in the trap increases, some crosstalk may occur during the measurements, as scattered light can affect neighboring ions. The measurement laser also heats the target ion and its neighbors, introducing thermal noise. Other important challenges in achieving reliable and scalable measurements include photon detection and laser stability. The photon detection issue was introduced in Section 2.2.2. The need for fast and accurate photon detection highlights the interconnection between different qubit technologies. Laser stability, which is crucial for reliable logical operations, is also important during the measurement process, as drifts in laser frequency can induce unwanted transitions in the ion, potentially moving its state out of the computational basis.

The results reported in this section demonstrate that trapped-ion technology is both a mature and promising platform for small-to-intermediate-scale universal quantum computing. While research efforts are primarily focused on scaling to larger systems, state-of-the-art trapped-ion computers can already be used to test small-scale quantum algorithms. Moreover, trapped-ion systems play a fundamental role in quantum sensing [131]. The high sensitivity of ions to external fields makes them ideal candidates for performing high-precision measurements of electric, magnetic, and gravitational fields.

2.2.4 Superconducting qubits

The idea of using superconducting circuits as qubits dates back to 1985 [132]. However, it was only after the invention of the transmon qubit in 2007 [133] that superconducting qubits became a leading technology in quantum computing. Today, major companies such as IBM, Google, and Intel are heavily investing in superconductive qubits. The race to achieve quantum supremacy with this technology has led to rapid advancements, resembling the improvements seen in the development of classical chips, which has given rise to the notion of a “quantum Moore’s law” [134].

Even though superconducting qubits have not yet reached the long decoherence times and gate fidelities of trapped ion qubits, their greatest advantage lies in the ability to engineer their characteristics. It is possible to construct different types of qubits using superconductive circuits, where the main difference lies in the physical quantities that determine the state of the qubit or on the superconductive circuit regime. The most famous superconductive qubits are: charge qubits [135], flux qubits [136], phase qubits [137], transmon qubits, and fluxonium qubits [138]. In this section, we will review the working principles of transmon qubits. In fact, this type of qubit is considered the basic element of superconductive qubits, on top of which some variations can be made to obtain the other types of qubits. Moreover, these qubits are used in the laboratories of the Technology Innovation Institute (TII) in Abu Dhabi, where I collaborated during my PhD and where some quantum algorithms described in Chapters 4 and 5 were executed.

In the first part, we will show how it is possible to engineer the Hamiltonian of the system and the importance of the Josephson junction in introducing anharmonicity. Then, we will explain how the qubit can be controlled and measured. Finally, we will provide an overview of the steps required to calibrate transmon qubits, as automating this procedure is a key objective of the TII laboratories [139].

Superconductive circuit

The idea behind superconducting qubits is to create a controllable and measurable two-level quantum system with characteristics that can be engineered. This contrasts with the previously introduced trapped ions and photonic qubits, where systems with fixed characteristics are used to implement the Hilbert space. For this reason, and because of the similarities with trapped ions qubits, superconductive qubits are also called “artificial atoms”.

The most well-known quantum system is the harmonic oscillator. It is possible to realize such a system using a superconducting circuit composed of an inductor and a capacitor (LC circuit), as shown in Figure 2.11. The Hamiltonian of this system is the sum of the energy stored in the capacitor and the energy stored in the inductor:

$$\mathcal{H} = \mathcal{E}_{\text{cap}} + \mathcal{E}_{\text{ind}} = \frac{Q^2}{2C} + \frac{\Phi^2}{2L}, \quad (2.6)$$

where we have used the conjugate variables Φ , the magnetic flux, and Q , the charge.

This system behaves like a harmonic oscillator with resonance frequency:

$$\omega_0 = \frac{1}{\sqrt{LC}}.$$

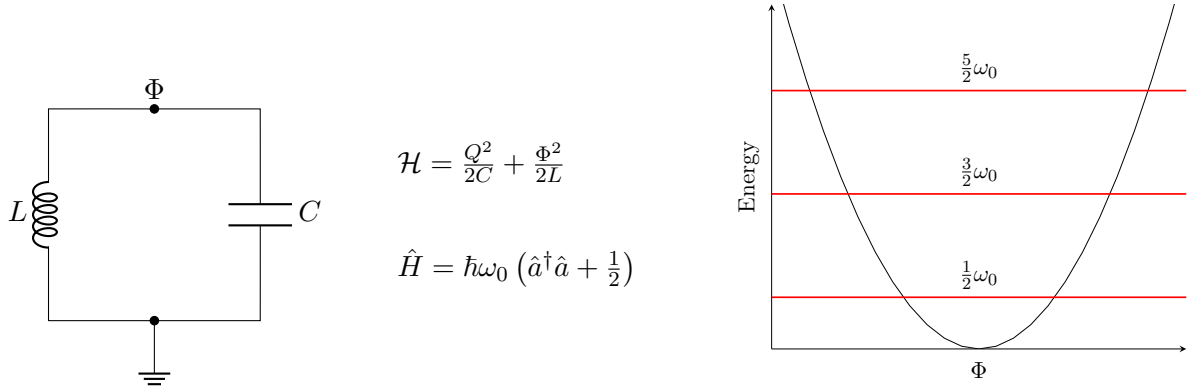


Figure 2.11: On the left: an LC superconducting circuit with its classical and quantized Hamiltonians. On the right: the harmonic potential and corresponding energy levels are depicted.

To quantize the system, we introduce the creation ($\hat{a}^\dagger$) and annihilation ($\hat{a}$) operators:

$$\hat{a} = \sqrt{\frac{1}{2\hbar Z}} (\hat{\Phi} + iZ\hat{Q}), \quad \hat{a}^\dagger = \sqrt{\frac{1}{2\hbar Z}} (\hat{\Phi} - iZ\hat{Q}),$$

where Z is the impedance defined as $Z = \sqrt{L/C}$. We recall some important characteristics of these operators:

$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle, \quad \hat{a}^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle, \quad [\hat{a}, \hat{a}^\dagger] = 1.$$

By substituting the creation and annihilation operators into the Hamiltonian in Equation 2.6 and using the commutation relations, we obtain:

$$\hat{H} = \hbar\omega_0 \left(\hat{a}^\dagger \hat{a} + \frac{1}{2} \right). \quad (2.7)$$

The energy levels can now be identified as:

$$E_n = \hbar\omega_0 \left(n + \frac{1}{2} \right).$$

The energy levels of a harmonic oscillator are equally spaced (Figure 2.11), which poses a problem in quantum computing where we need isolate two levels. Before explaining the anharmonic oscillator, it is essential to introduce two important values:

$$\Phi_{\text{zpf}} = \sqrt{\frac{\hbar Z}{2}}, \quad Q_{\text{zpf}} = \sqrt{\frac{\hbar}{2Z}},$$

which are related to the flux and charge fluctuations in the energy eigenstates:

$$\langle n | \hat{\Phi}^2 | n \rangle = (2n+1)\Phi_{\text{zpf}}^2, \quad \langle n | \hat{Q}^2 | n \rangle = (2n+1)Q_{\text{zpf}}^2.$$

In particular, in the ground state, the standard deviation of the flux is Φ_{zpf} (zpf stands for zero-point fluctuations).

In order to obtain a system with non-equally spaced energy levels, it is necessary to introduce a nonlinear element into the superconducting circuit. For this purpose, we replace the inductor with

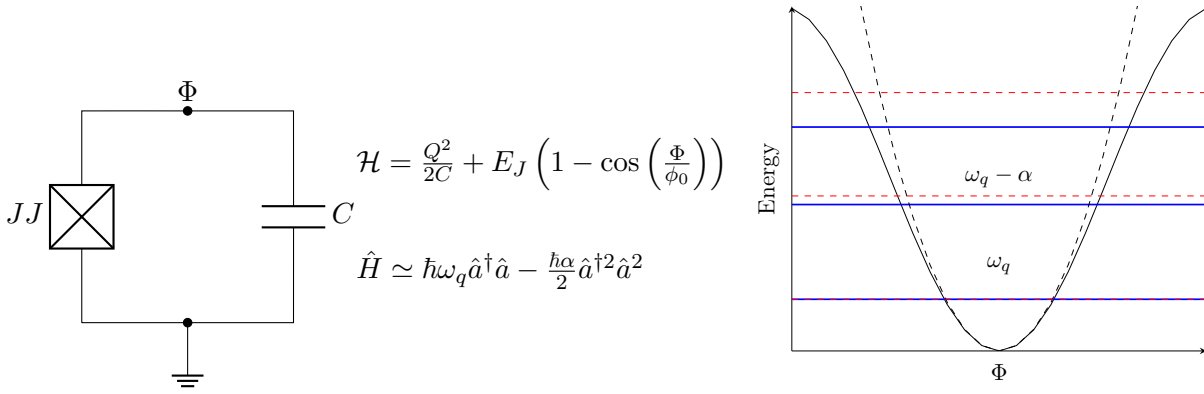


Figure 2.12: On the left: an anharmonic circuit with a Josephson junction, showing the classical and quantized Hamiltonians (to first-order perturbation). On the right: the anharmonic potential and energy levels of the system, with the harmonic potential and energy levels overlaid using dashed lines. The gap between the ground state and the first excited energy level is ω_q , while the gap between the first and second excited states is $\omega_q - \alpha$, where typical values of α are around 10% of ω_q .

a Josephson junction (Appendix B), as shown in Figure 2.12. The energy stored in a Josephson junction depends on the magnetic flux as:

$$\mathcal{E}_j(\Phi) = E_J \left(1 - \cos \left(\frac{\Phi}{\phi_0} \right) \right), \quad \text{where: } \phi_0 = \frac{\hbar}{2e}, \quad (2.8)$$

where we have introduced the reduced magnetic flux quantum ϕ_0 and E_J , a characteristic energy of the junction (see Appendix B for more details). A useful representation of the Hamiltonian of the circuit in Figure 2.12 is:

$$\mathcal{H} = 4E_C n^2 - E_J \left(1 - \cos \left(\frac{\Phi}{\phi_0} \right) \right),$$

where $E_C = e^2/2C$ is called the charging energy and n is the number of Cooper-pairs displaced in the capacitor. The ratio E_J/E_C is a fundamental parameter of the qubit as increasing this ratio allows to mitigate charge noise [133]. In a Transmon qubit $E_J/E_C \sim 50 - 100$, large values of this ratio are usually obtained by employing a large capacitor.

By expanding Equation 2.8 in a Taylor series around small values of the flux and removing the constant terms, we can split this equation into linear and nonlinear terms:

$$\mathcal{E}_j = \mathcal{E}_{\text{lin}} + \mathcal{E}_{\text{nl}} = \frac{E_J}{2} \left(\frac{\Phi}{\phi_0} \right)^2 - \frac{E_J}{24} \left(\frac{\Phi}{\phi_0} \right)^4 + o \left(\frac{\Phi}{\phi_0} \right)^6. \quad (2.9)$$

In the linear term, we can identify the Josephson inductance: $L_J = \frac{\phi_0^2}{E_J}$. Substituting the Josephson junction's energy contribution in place of the inductor in Equation 2.6, we obtain the Hamiltonian for the anharmonic circuit (harmonic plus anharmonic contribution):

$$\mathcal{H} = \mathcal{H}_{\text{har}} + \mathcal{H}_{\text{anh}} \simeq \frac{Q^2}{2C} + \frac{\Phi^2}{2L_J} - \frac{E_J}{24} \left(\frac{\Phi}{\phi_0} \right)^4,$$

where we have omitted higher-order terms of Φ/ϕ_0 and the constant terms. To quantize the Hamiltonian, we follow the same procedure as for the harmonic oscillator, using creation and annihilation

operators. With some algebra, we arrive at:

$$\hat{H} = \hbar\omega_0 \left(\hat{a}^\dagger \hat{a} + \frac{1}{2} \right) - \frac{E_J}{24} \phi_{\text{zpf}}^4 \left(\hat{a}^\dagger + \hat{a} \right)^4,$$

where $\omega_0 = \sqrt{\frac{1}{L_J C}}$ is the resonance frequency of the harmonic part, and $\phi_{\text{zpf}} = \frac{\Phi_{\text{zpf}}}{\phi_0}$ is the reduced zero-point flux fluctuation. Applying first-order perturbation theory and dropping constant terms, we get:

$$\hat{H} = \hbar\omega_q \left(\hat{a}^\dagger \hat{a} \right) - \frac{\hbar\alpha}{2} \hat{a}^{\dagger 2} \hat{a}^2, \quad \text{where} \quad \alpha = \frac{E_J \phi_{\text{zpf}}^4}{2\hbar} = \frac{E_C}{\hbar},$$

where ω_q is the qubit frequency for the first transition (this value differs from ω_0 due to the nonlinear term). We can now identify the energy levels:

$$E_n = \hbar\omega_q n - \frac{\hbar\alpha}{2} n(n-1).$$

Typical values for transmon qubit construction are:

$$C \sim 10 - 100 \text{ fF}, \quad L_J \sim 10 \text{ nH},$$

which lead to typical qubit frequencies and anharmonicity values of:

$$\omega_q \sim 1 - 10 \text{ GHz}, \quad \alpha \sim 100 \text{ MHz}.$$

The zero-point fluctuations are approximately:

$$\phi_{\text{zpf}} \sim 0.5, \quad \frac{Q_{\text{zpf}}}{2e} \sim 1. \quad (2.10)$$

The small value of the reduced zero-point fluctuations flux justifies the omission of higher-order terms in the Taylor expansion of Equation 2.9.

By restricting the circuit control system to pulses in a frequency range close to ω_q , it is possible to isolate only the first two energy levels, thus obtaining a qubit. This regime is known as the infinite anharmonicity approximation. In this picture, it is easy to retrieve the Pauli operators from the creation and annihilation operators:

$$\hat{\sigma}_z = \hat{I} - 2\hat{a}^\dagger \hat{a}, \quad \hat{\sigma}_x = \hat{a} + \hat{a}^\dagger, \quad \hat{\sigma}_y = i \left(\hat{a}^\dagger - \hat{a} \right).$$

Logical operations

To change the logical state of a superconducting qubit, it is necessary to send pulses to it, typically in the microwave frequency range, using a control or drive line. To couple the qubit with the drive line, a capacitor (C_d) is usually employed, as shown in Figure 2.13. To minimize energy losses through the drive, it is advantageous to use a small coupling capacitor ($C_d \ll C_q$).

In this regime, it is possible to derive the qubit Hamiltonian along with the drive interaction term [140]. For simplicity, we will omit the anharmonic term and consider the qubit as a simple LC circuit:

$$\mathcal{H} \simeq \mathcal{H}_q + \mathcal{H}_{\text{int}} = \frac{Q^2}{2C} + \frac{\Phi^2}{2L} - \frac{C_d}{C} V_d(t) Q, \quad (2.11)$$

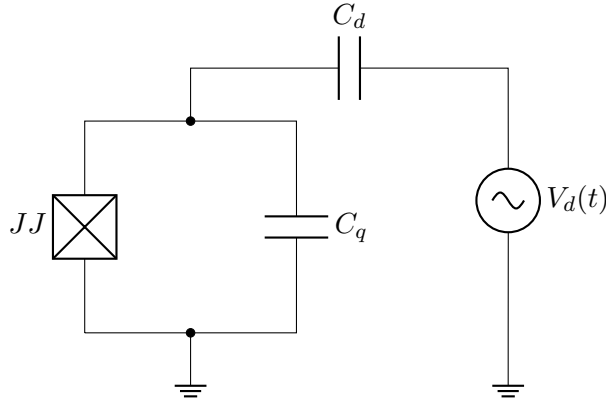


Figure 2.13: Circuit model of a Transmon qubit coupled with a drive line ($V_d(t)$) through a capacitor (C_d).

where $C = C_q + C_d$ is the total capacitance. We quantize this Hamiltonian using the relation

$$\hat{Q} = iQ_{\text{zpf}} (\hat{a} - \hat{a}^\dagger), \quad (2.12)$$

and reduce it to Pauli operators in the infinite anharmonicity approximation:

$$\hat{H} = -\hbar \frac{\omega_q}{2} \hat{\sigma}_z + \Omega V_d(t) \hat{\sigma}_y,$$

where $\Omega = \frac{C_d}{C} Q_{\text{zpf}}$. To observe the effect of the drive pulse, it is useful to move to the interaction picture, where states evolve only under the interaction Hamiltonian. In this picture, the interaction Hamiltonian becomes:

$$\hat{H}_{\text{int}}^{\text{int-picture}}(t) = \left(e^{\frac{-i\hat{H}_q t}{\hbar}} \right)^\dagger \hat{H}_{\text{int}} e^{\frac{-i\hat{H}_q t}{\hbar}} = \Omega V_d(t) (\cos(\omega_q t) \hat{\sigma}_y - \sin(\omega_q t) \hat{\sigma}_x).$$

Now, consider a perfect sinusoidal signal with the same frequency as the qubit: $V_d(t) = V_0 \sin(\omega_q t + \phi)$. Under the rotating wave approximation, we obtain [141]:

$$\hat{H}_{\text{int}}^{\text{int-scheme}}(t) = -\frac{\Omega V_0}{2} (\cos(\phi) \hat{\sigma}_x + \sin(\phi) \hat{\sigma}_y).$$

This shows that the phase of the drive pulse defines the rotation axis on the Bloch sphere. For instance, to implement an R_x gate, it is necessary to send a pulse with phase $\phi = 0$:

$$R_x(\theta_t) = e^{-i \frac{\theta_t}{2\hbar} \hat{\sigma}_x}, \quad \text{where} \quad \theta_t = \frac{\Omega V_0}{\hbar} t.$$

A pulse of this kind, such that $\theta_t = \pi$, is known as a π -pulse and implements a Pauli X gate.

An R_z gate has no effect on measurements in the computational basis. Moreover, this gate alters the rotation axis of subsequent rotation gates. This allows any R_z gate to be applied virtually by adjusting the phase of the subsequent gates. For example:

$$R_z(-\theta) R_x(\phi) R_z(\theta) = e^{-i \frac{\phi}{2} (\cos(\theta) \sigma_x + \sin(\theta) \sigma_y)}.$$

As shown in Equation 2.3, it is possible to execute any unitary gate using two R_x and three R_z gates. This implies that any unitary can be implemented on hardware using only two physical

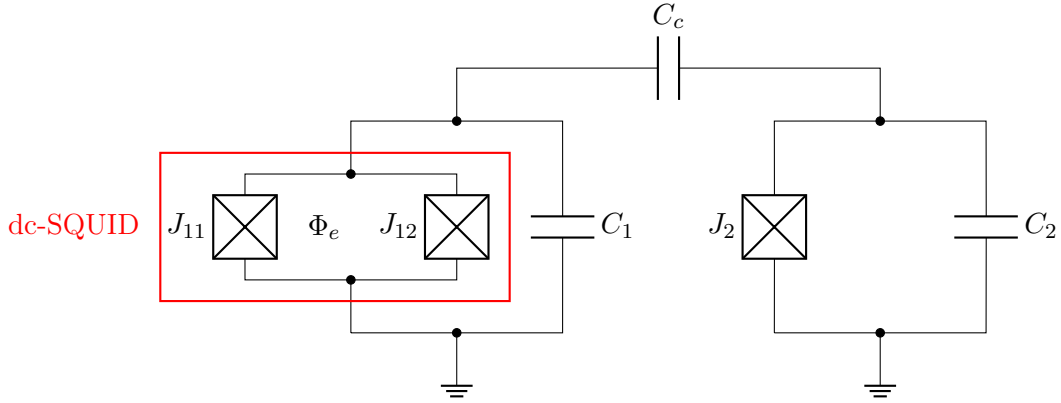


Figure 2.14: Circuit model of two Transmon qubits directly coupled through a capacitor (C_c). The qubit on the left is flux tunable, the dc-SQUID element is highlighted.

pulses.

It is finally important to highlight that in Equation 2.11 we have omitted the anharmonic term. This term becomes important when considering that the Fourier transform of a pulse with a finite duration does not contain a single frequency. To mitigate this issue, it is important to shape the pulse [142] and employ techniques such as the DRAG scheme [143]. With these advancements it has been possible to obtain single-qubit fidelities exceeding 99.99% [144].

Many different possibilities have been proposed to implement two-qubit gates with superconducting qubits. Two common requirements can be identified: the qubits must be coupled and their frequencies must be tunable. The tuning of the qubit frequency can be achieved in several ways, typically involving an external magnetic field. The simplest solution is the flux-tunable transmon qubit, where the Josephson junction is replaced by a loop composed of two Josephson junctions (Figure 2.14, left qubit). This loop is also known as a dc-SQUID [145]. When we apply an external magnetic flux (Φ_e) passing through the loop the flux quantization leads to the relation $\Phi_1 - \Phi_2 + 2\Phi_e = n\phi_0$. Consequently, we can express the energy contribution to the circuit Hamiltonian of this device, removing the constant terms, in terms of the external field and the flux of a single junction [141]:

$$\mathcal{H}_{\text{dc-SQUID}} = -2E_J \cos\left(\frac{\Phi_e}{\phi_0}\right) \cos\left(\frac{\Phi}{\phi_0}\right).$$

This expression demonstrates how it is possible to tune the qubit frequency by changing the external field, although this comes with the drawback of increasing the qubit's sensitivity to uncontrolled flux fluctuations.

Regarding the coupling, it is possible to utilize either capacitive or inductive coupling. Furthermore, the coupling can be either direct or mediated by a coupler, which is typically an LC circuit. Let's consider the simplest case of direct capacitive coupling (C_c) as illustrated in Figure 2.14. In the limit of a small coupling capacitance ($C_c \ll C_1, C_2$), the Hamiltonian of the system becomes [140]:

$$\mathcal{H} = \mathcal{H}_{q1} + \mathcal{H}_{q2} + \frac{C_c}{C_1 C_2} Q_1 Q_2.$$

We can quantize this Hamiltonian using Equation 2.12 to obtain:

$$\hat{H} = \hat{H}_{q1} + \hat{H}_{q2} - \hbar g \left(\hat{a}_1 - \hat{a}_1^\dagger \right) \otimes \left(\hat{a}_2 - \hat{a}_2^\dagger \right),$$

where $g = \frac{1}{\hbar} \frac{C_c}{C_1 C_2} Q_{1\text{zpf}} Q_{2\text{zpf}}$ is the coupling constant.

To observe the effects of this interaction, it is necessary to use the interaction picture. By applying the rotating wave approximation, the interaction Hamiltonian becomes:

$$\hat{H}_{\text{int}}^{\text{int-picture}}(t) = \hbar g \left(e^{i(\omega_1 - \omega_2)t} \hat{a}_1^\dagger \otimes \hat{a}_2 + e^{-i(\omega_1 - \omega_2)t} \hat{a}_1 \otimes \hat{a}_2^\dagger \right).$$

This interaction becomes significant when the two qubits frequencies are the same. This can be achieved by changing the frequency of one of the two flux-tunable qubits, as explained previously. At the same frequency, in the infinite anharmonicity limit, the interaction between the two qubits becomes:

$$\hat{H}_{\text{int}}^{\text{int-picture}}(t) = \frac{\hbar g}{2} (\hat{\sigma}_x \otimes \hat{\sigma}_x + \hat{\sigma}_y \otimes \hat{\sigma}_y).$$

This is the same interaction described in Equation 2.5 for trapped ion qubits. By allowing the qubits to interact for a time $t = \frac{\pi}{2g}$, it is possible to obtain an i-SWAP gate (Equation 2.4).

It is also feasible to implement a CZ gate using a similar mechanism by utilizing the second excited level of one of the two qubits [141]. Additionally, a slightly different framework enables the implementation of two-qubit gates between qubits that are coupled to the same cavity resonator by transferring the state of the qubits to the cavity for interaction [146]. This approach does not require changing the qubit frequency and is analogous to trapped ions, where the modes of the chain act as the coupler.

State-of-the-art experiments have achieved two-qubit gate fidelities of up to 99.9% [147, 148]. Typical gate times are in the orders of 10 ns for single-qubit gates and require up to 100 ns for two-qubit gates. In a recent experiment it has been possible to obtain a CZ gate with 99.5% of fidelity in 30 ns [149]. This values have to be compared to the standard qubit lifetimes that are in the order of 100 μs , reaching state-of-the-art values of 500 μs [150]. Although these results are encouraging, a significant challenge lies in maintaining such high fidelities, and qubit lifetimes, while scaling up the number of qubits. Moreover, since two-qubit gates can only be implemented between coupled qubits, each multi-qubit chip has its own connectivity map. When a quantum algorithm necessitates a two-qubit gate between non-coupled qubits, it becomes necessary to swap the logical states of the qubits during the transpiling process [35]. This introduces an overhead of operations that increases noise. Thus, enhancing chip connectivity is crucial to reliably execute quantum algorithms.

Measurements

It is possible to perform non-demolition measurements with high precision in superconducting qubits. To reduce the probability of altering the qubit's state, it is necessary to weakly couple the qubit to a readout resonator and measure the qubit indirectly by observing the coupled system. For this purpose, the qubit can be coupled either to a physical superconducting LC circuit or to the photon cavity that surrounds the qubits. In both cases, the total system can be schematized as a qubit coupled to a harmonic LC circuit, as depicted in Figure 2.15. Moreover, a readout line is required to observe the state of the resonator. In the limit of small coupling $C_c, C_{rl} \ll C_q, C_r$ and

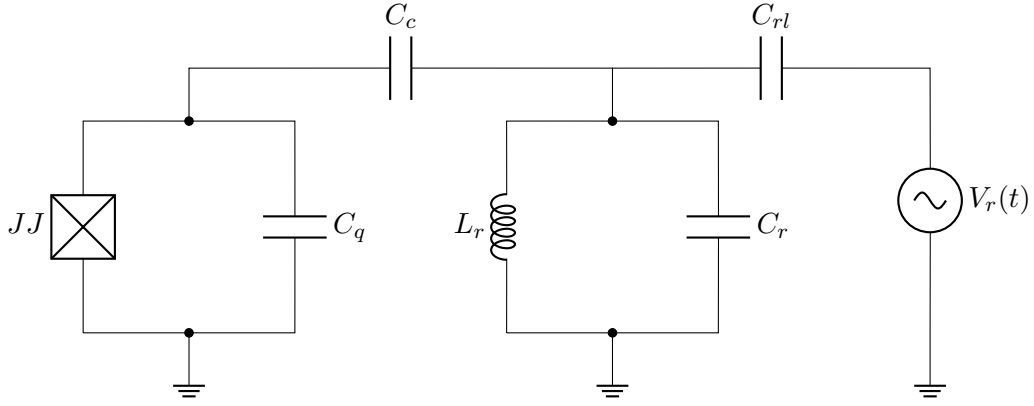


Figure 2.15: Circuit model a Transmon qubit coupled with a resonator (LC circuit) through a capacitor (C_c). A readout line capacitive coupled with the resonator (C_{rl}) allows to perform a measurement of this system.

by ignoring the readout line, the Hamiltonian of the system is:

$$\mathcal{H} = \mathcal{H}_q + \mathcal{H}_r + \frac{C_c}{C_q C_r} Q_q Q_r.$$

Following the same steps used in the study of two-qubit gates, we can obtain the Jaynes-Cummings Hamiltonian [151], which describes the interaction of a qubit with a resonator:

$$\hat{H} = \hbar\omega_r \left(\hat{a}^\dagger \hat{a} + \frac{1}{2} \right) + \frac{\hbar\omega_q}{2} \hat{\sigma}_z - \hbar g \left(\hat{\sigma}_+ \hat{a} + \hat{\sigma}_- \hat{a}^\dagger \right), \quad (2.13)$$

where $\omega_r \simeq \frac{1}{\sqrt{L_r C_r}}$ is the resonator frequency. The operators $\hat{a}^\dagger$ and $\hat{a}$ raise and lower the energy level of the resonator, while $\hat{\sigma}_+$ and $\hat{\sigma}_-$ raise and lower the energy level of the qubit in the infinite anharmonicity limit. To avoid energy excitation swapping between the qubit and the resonator, it is necessary to work in the dispersive regime where the qubit frequency is detuned from the resonator frequency ($g \ll |\omega_q - \omega_r|$). In this limit, it is possible to perform second-order perturbation theory or the Schrieffer-Wolff transformation [152, 153] to obtain the effective Hamiltonian:

$$\hat{H}_{\text{eff}} = \hbar(\omega_r + \chi \hat{\sigma}_z) \hat{a}^\dagger \hat{a} + \frac{\hbar(\omega_q + \chi)}{2} \hat{\sigma}_z, \quad (2.14)$$

where $\chi = \frac{g^2}{|\omega_q - \omega_r|}$ is the dispersive shift. The first term of this Hamiltonian indicates that the frequency of the resonator depends on the state of the qubit. This effect enables measurements by sending a probe signal $V_r(t)$ with a frequency close to ω_r and observing the resonator's response, as shown in Figure 2.16.

In Equation 2.13, we have not considered the anharmonicity. When incorporating this term, it is necessary to operate in a regime where the energy level of the resonator, or the photon number in the cavity, remains below a critical value [154]:

$$n_c = \frac{|\omega_q - \omega_r|^2}{4g^2}.$$

This limitation on the resonator excitation number imposes a bound on the intensity of the probe signal. When this condition is satisfied, the result from Equation 2.14 is recovered with a slight

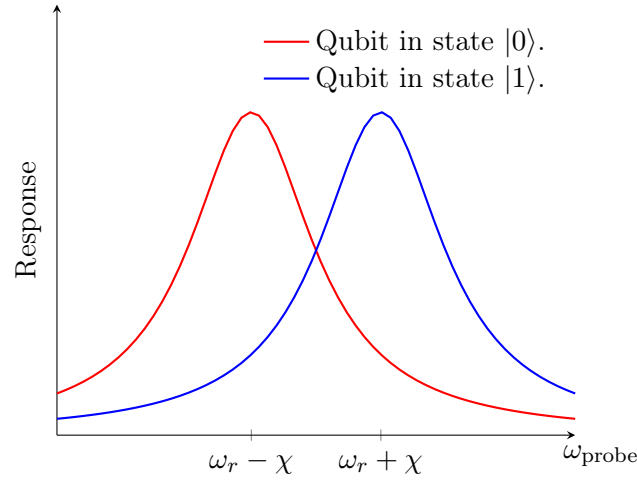


Figure 2.16: Resonator response as a function of the probe signal frequency. Since the resonator is coupled to the qubit, its resonant frequency shifts depending on the qubit’s state. This results in two distinct response curves: one for each qubit state. By observing these shifts in the resonator’s response, the state of the qubit can be discriminated.

modification to the dispersive shift [141]:

$$\chi = -\frac{g^2}{|\omega_q - \omega_r|} \left(\frac{1}{1 + |\omega_q - \omega_r|/\alpha} \right).$$

State-of-the-art experiments have achieved measurement accuracy of up to 99.8% in less than 100 ns [155, 156]. To improve these results, it is essential to optimize the coupling between the qubit and the resonator, increasing the dispersive shift while minimizing the measurement’s back-action on the qubit state [157]. Moreover, it is important to enhance the amplification process of the readout signal using low-noise amplifiers, such as traveling-wave parametric amplifiers (TWPAs) [158, 159]. Scaling to systems comprising multiple qubits presents additional challenges. In such systems, readout resonators can either differ for each qubit or be shared among several qubits. In both scenarios, it is crucial to reduce crosstalk between qubits that may be induced by the measurement procedure [160].

Calibration

As superconductive qubits are fabricated, they are subjected to construction imperfections that alter their fundamental parameters from the values predicted by theory. To implement high-fidelity logical operations and readout, it is necessary to know all the qubit parameters with precision. This is achieved through a calibration routine. Moreover, the parameters of a qubit may change over time due to electromagnetic radiation, thermal fluctuations, and even mechanical vibrations. This necessitates regular calibration while the qubit is in use to avoid performance degradation.

Since calibration routines can be time-consuming, it is useful to implement them automatically. An interesting example is the open-source software Qibocal [139], which enables the calibration and characterization of self-hosted superconducting quantum devices, with compatibility across different control instruments.

The main characterization routines can be divided into three groups: readout resonator, single qubit and two-qubit gates. We will give a brief introduction of the most important of these routines, a

summary is also reported in Table 2.2. Note that different types of qubits or control instruments may require different routines or ways to perform them.

The first calibration step requires the identification of the parameters of the readout resonator. In fact, the ability to perform measurements is necessary for the calibration of single and two-qubit gates. To find the cavity resonant frequency (ω_r), it is necessary to send a frequency sweep into the resonator line and collect the output signal. The amplitude of the output signal should follow a Lorentzian curve; by fitting this curve, it is possible to identify the resonant frequency at the peak and other important parameters, such as the quality factor. After the resonant frequency has been found, it is necessary to tune its power (V_r) to ensure optimal measurement contrast without causing measurement-induced state transitions. Finally, to improve the readout accuracy, it is important to identify the dispersive shift χ . However, to perform this routine, it is necessary to have the ability to implement an X gate, thus the fine-tuning of the resonator can be performed only after the single-qubit gates have been characterized.

To characterize single-qubit gates, the first step is finding the resonant frequency of the qubit ω_q . The procedure is performed in the same way as for the resonator, but the probe frequency sweep is performed through the drive line. After finding ω_q , the Rabi calibration can be performed. The objective of this experiment is to find the correct pulse time or amplitude (V_d) necessary to perform an X gate. A drive signal with pulse ω_q is sent to the qubit for a range of different time lengths or amplitudes, and the measurement probabilities outcomes are collected. The probability outcomes, with respect to the drive pulse duration/amplitude, should follow a sinusoidal curve. By performing a fit of a sine function, it is possible to find the correct time duration or amplitude of the drive pulse. After performing a Rabi experiment, it is possible to determine important qubit characteristics. To find the decoherence time in the z axis (T_1), an energy relaxation experiment can be performed. The qubit is initialized to the $|1\rangle$ state, and a measurement is performed after some time τ . The measurement outcome as a function of τ should follow an exponential decay. With a fit of this curve, it is possible to identify T_1 as the decay parameter. Another important parameter of qubits is the decoherence time in the xy plane, known as T_2 . This parameter is obtained through the Ramsey experiment. The idea to find this parameter is similar to finding T_1 . However, this time we send a pulse to move the state from the ground state to the xy plane in the Bloch sphere, for example, with a Hadamard gate, wait a certain time τ , then return to the ground state and perform the measurement.

After calibrating single-qubit gates and having extracted the most important single-qubit parameters, it is possible to start characterizing two-qubit gates. This procedure changes depending on the gate being implemented. We will consider the simple case of an i-SWAP gate with flux-tunable qubits; the procedure to characterize a CZ gate with this kind of qubit is similar. The first parameter to be found is the external field ϕ_e , which must be applied to the dc-SQUID of one of the two qubits to match its frequency with the second qubit in the gate. This parameter can be obtained by studying the behavior of ω_q as a function of ϕ_e . At this point, the second fundamental parameter to characterize the gate is the interaction time. To find this parameter, it is necessary to observe the phase acquired by the states $|01\rangle$ and $|10\rangle$ as a function of the interaction time.

All the procedures described in this section must be repeated for every qubit and resonator

Component	Procedure Name	Parameter
Readout Resonator	Frequency sweep	Resonant frequency (ω_r)
	Power tuning	Power of readout pulse (V_r)
	Dispersive shift measurement	Dispersive shift (χ)
Single qubit	Frequency sweep	Qubit frequency (ω_q)
	Rabi experiment	Pulse time/amplitude (V_d)
	Energy relaxation	Decoherence time T_1
	Ramsey experiment	Decoherence time T_2
Two-qubit gates	External field adjustment	External field (ϕ_e)
	Interaction time	Interaction time ($\pi/2g$)

Table 2.2: Summary of calibration procedures for superconductive qubits.

in the quantum chip, while the two-qubit gate characterization must be performed for every pair of coupled qubits in the chip. As the size of quantum chips increases, it is crucial to keep the calibration procedure fast and reliable.

It is worth adding that after the calibration has been performed, it is possible to benchmark the efficiency of the superconductive chip. The most famous benchmarking procedure is known as Randomized Benchmarking (RB) (Appendix C).

2.3 Noise

The main problem of quantum computing regards the presence of noise. Given the importance of this topic it is necessary to delve deeper into it. For this purpose in Sections 2.3.1 and 2.3.2 the mathematical tools necessary to describe spurious quantum states and their interaction with the environment will be introduced. In Section 2.3.3 we will give a review of the most important forms of noise. Finally, in Sections 2.3.4 and 2.3.5 we will introduce the basic concepts of quantum error correction.

2.3.1 Density matrices

A quantum system cannot always be described by a single state vector. When a state can be represented by a single vector it is called a pure state. On the other hand, a mixed state is a statistical ensemble of pure states. Consider, for example, the photons emitted by a light source. The energy of a single photon is unknown but it is possible to have information about the probability distribution of the energy. The state of each photon is taken from the ensemble of pure states $\{|\psi_i\rangle\}$ with probabilities $\{p_i\}$. A mixed state can be described using the density operator that is an operator instead of a vector. The density operator ρ is defined as:

$$\rho := \sum_i p_i |\psi_i\rangle \langle \psi_i|.$$

Given $\{|i\rangle\}$, an orthonormal basis, it is possible to obtain the matrix representation of this operator that is called Density Matrix (DM). The elements of this matrix are simply given by: $\rho_{ij} = \langle i | \rho | j \rangle$. The density operator satisfies some important properties:

1. ρ is Hermitian.
2. ρ has unit trace.
3. ρ is non-negative, this means that for any vector $|\psi\rangle \in \mathcal{H}$: $\langle \psi | \rho | \psi \rangle \geq 0$.
4. $Tr(\rho^2) = 1$ for a pure state while $Tr(\rho^2) \leq 1$ for a mixed state.

With the density operator formalism, the average value of an operator A , on the mixed state described by ρ , can be calculated as:

$$\langle A \rangle = Tr(\rho A). \quad (2.15)$$

To check this equation notice that the probability $p(k)$ of obtaining the result a_k from a measurement of A is:

$$p(k) = \sum_i p_i \langle \psi_i | P_k | \psi_i \rangle,$$

where P_k is the projection operator on the subspace corresponding to the eigenvalue a_k . The average value of A is given by:

$$\langle A \rangle = \sum_k p(k) a_k.$$

Expanding the trace in Equation 2.15 gives:

$$\begin{aligned} \text{Tr}(\rho A) &= \sum_k \langle k | \rho A | k \rangle = \sum_i \sum_k p_i \langle k | \psi_i \rangle \langle \psi_i | A | k \rangle = \\ &= \sum_i p_i \langle \psi_i | A | \psi_i \rangle = \sum_k a_k \sum_i p_i \langle \psi_i | P_k | \psi_i \rangle = \sum_k p(k) a_k, \end{aligned}$$

where Equation 2.2 and the completeness relation have been used.

The density operator can be used to represent the state of a composite system. This is useful in quantum computing and quantum error correction to describe the interactions between the system and the environment.

Consider a bipartite quantum system described by the Hilbert space $\mathcal{H}_1 \oplus \mathcal{H}_2$. The total system is described by the DM ρ . Given $\{|i\rangle_1\}$ and $\{|\alpha\rangle_2\}$, orthonormal basis, respectively of the first and the second subsystems, the matrix elements of ρ can be obtained as:

$$\rho_{i\alpha;j\beta} := \langle i |_1 \langle \alpha |_2 \rho | j \rangle_1 | \beta \rangle_2.$$

In the case of quantum computing, the first subsystem is composed of the qubits, while the second subsystem is the environment. A measurement of the qubits is represented by an operator A_1 acting only on the first subsystem. The average value of the operator can be computed by trivially extending it to the whole Hilbert space as $A_1 \otimes I_2$:

$$\langle A_1 \rangle = \text{Tr}(\rho A_1 \otimes I_2).$$

By expanding the trace over the DM and exploiting the orthogonality relations, the previous equation becomes:

$$\langle A_1 \rangle = \sum_{i,j,\alpha} \rho_{i\alpha;j\alpha} \langle i |_1 A_1 | j \rangle_1. \quad (2.16)$$

The reduced DM of the first subsystem (ρ_1) is defined as the partial trace over the second subsystem:

$$\rho_1 = \text{Tr}_2(\rho) = \sum_{\alpha} \langle \alpha |_2 \rho | \alpha \rangle_2.$$

The matrix elements of the reduced DM are simply given by:

$$(\rho_1)_{ij} = \sum_{\alpha} \langle i |_1 \langle \alpha |_2 \rho | j \rangle_1 | \alpha \rangle_2 = \sum_{\alpha} \rho_{i\alpha;j\alpha}.$$

Introducing this relation in Equation 2.16, it is easy to observe that:

$$\langle A_1 \rangle = \text{Tr}(\rho_1 A_1).$$

Therefore, it is possible to compute the expectation value of an operator acting only on the first subsystem as if the system was isolated. The reduced density matrices of the two subsystems, however, don't give a complete description of the total system. In fact, the reduced trace cancels entanglement terms so that: $\rho \neq \rho_1 \otimes \rho_2$. The presence of entanglement between the two subsystems reflects on the fact that the evolution of the reduced density matrices is not unitary. For example,

a pure state after an interaction with a second system may become a mixed state. In order to better understand these concepts, that give a description of how errors arise during a quantum computation, it is useful to introduce the Kraus representation.

2.3.2 Kraus representation

Kraus representation is used to describe the evolution of a quantum system that interacts with the environment by means of non-unitary maps called superoperators. Consider a bipartite quantum system described by the density operator ρ_{12} , on the Hilbert space $\mathcal{H}_1 \oplus \mathcal{H}_2$. Before interaction the two subsystems are not entangled. The first subsystem is described by the reduced density operator ρ_1 . The second subsystem, that is the environment, may be taken, without loss of generality, in the pure state $|0\rangle_2$. The initial state of the system is:

$$\rho_{12} = \rho_1 \otimes |0\rangle_2 \langle 0|_2.$$

The time evolution of the total system is governed by the unitary operator U . After the interaction, the state of the first subsystem is described by the reduced density operator ρ'_1 . The quantum circuit that represent this process is reported in Figure 2.17.

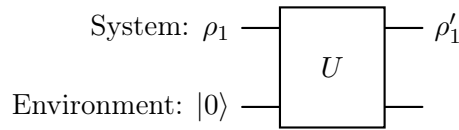


Figure 2.17: Schematic representation of the interaction between a quantum system and the environment.

Given $\{|k\rangle_2\}$, an orthonormal basis for the second system, it is possible to write:

$$\rho'_1 = \text{Tr}_2 \left[U(\rho_1 \otimes |0\rangle_2 \langle 0|_2) U^\dagger \right] = \sum_k \langle k|_2 U |0\rangle_2 \rho_1 \langle 0|_2 U^\dagger |k\rangle_2. \quad (2.17)$$

At this point it is possible to define the Kraus operators $\{E_k\}$ as:

$$E_k := \langle k|_2 U |0\rangle_2.$$

In this way Equation 2.17 may be rewritten as:

$$\rho'_1 = \sum_k E_k \rho_1 E_k^\dagger.$$

Kraus operators act on the space of the first subsystem and are not unitary. However, since U is unitary, they respect an important relation:

$$\sum_k E_k^\dagger E_k = \sum_k \langle 0|_2 U^\dagger |k\rangle_2 \langle k|_2 U |0\rangle_2 = \langle 0|_2 U^\dagger U |0\rangle_2 = I_1.$$

A set of operators $\{E_k\}$, such that $\sum_k E_k^\dagger E_k = I$, defines a linear map ($\mathcal{S}$) from density operators to density operators, called superoperator:

$$\mathcal{S} : \rho \rightarrow \rho' = \sum_k E_k \rho E_k^\dagger.$$

In order to map density operators to density operators, a superoperator has to respect some fundamental properties:

1. If ρ is Hermitian then ρ' is Hermitian:

$$(\rho')^\dagger = \left(\sum_k E_k \rho E_k^\dagger \right)^\dagger = \sum_k \left(E_k^\dagger \right)^\dagger \rho^\dagger E_k^\dagger = \sum_k E_k \rho E_k^\dagger = \rho'.$$

2. If ρ has unit trace then ρ' has unit trace:

$$\text{Tr}(\rho') = \sum_k \text{Tr}(E_k \rho E_k^\dagger) = \sum_k \text{Tr}(E_k^\dagger E_k \rho) = \text{Tr}\left(\rho \sum_k E_k^\dagger E_k\right) = \text{Tr}(\rho).$$

3. If ρ is non negative then ρ' is non negative, in fact given a vector $|\psi\rangle$:

$$\langle \psi | \rho' | \psi \rangle = \sum_k \langle \psi | E_k^\dagger \rho E_k | \psi \rangle = \sum_k \langle \psi_k | \rho | \psi_k \rangle \geq 0,$$

where $|\psi_k\rangle := E_k |\psi\rangle$.

Two superoperators can be composed. For example, if S_a describes the time evolution from t_0 to t_1 and S_b the evolution from t_1 to t_2 , the superoperator $S := S_b S_a$ describes the time evolution from t_0 to t_2 . However, since a superoperator may be non-unitary, it is not always possible to invert it. Physically, the non-invertibility is due to the loss of information from the quantum system to the environment. It is not possible to reconstruct the initial state of the quantum system that has interacted, ignoring the state of the environment.

Kraus representation shows that, the unitary evolution of a composite system naturally gives rise to an operator-sum representation describing the evolution of a subsystem. The converse is also true: given a set of Kraus operators, that describe the evolution of a quantum system, it is always possible to introduce an auxiliary system such that the evolution of the total system is unitary. The unitary operator that rules the evolution of the total system is called the unitary representation of a superoperator. Some details about this important result are reported in Appendix D.

2.3.3 Noise channels

In order to model quantum noise it is possible to use error channels. They are represented as superoperators acting on state density matrices. The following list provides an overview of the most important error channels.

Bit-flip: this channel, also known as X-channel, acts as classical noise on bits. This channel can be used to model measurement errors in the computational basis. When a qubit undergoes

the action of the bit-flip channel, there is a probability that its state is subjected to the Pauli X operator, resulting in flipping its computational basis state:

$$\begin{aligned} |0\rangle &\rightarrow |1\rangle && \text{With probability } p, \\ |1\rangle &\rightarrow |0\rangle && \text{With probability } q. \end{aligned}$$

The bit-flip channel is defined as symmetric when $p = q$. In this case, it is easy to find that the Kraus operators are:

$$E_0 = \sqrt{p} \cdot X, \quad E_1 = \sqrt{1-p} \cdot I.$$

Dephasing: this channel, also called phase damping channel or Z-channel, is defined as the symmetric X-channel with the difference that the state is subjected with probability p to a Z Pauli operator.

$$\text{Dephasing}_p(\rho) = (1-p)\rho + pZ\rho Z.$$

The dephasing channel is useful for modeling errors that cause the loss of coherence in quantum states. This channel can be visualized as a collapse of the Bloch sphere toward the z axis as shown in Figure 2.18.

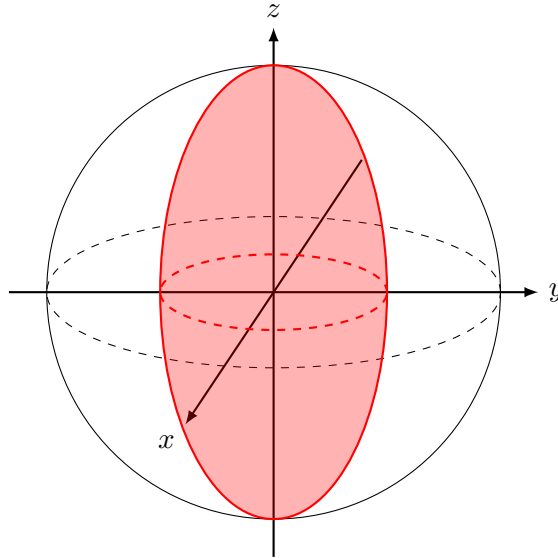


Figure 2.18: Representation of the effect of the dephasing channel on the Bloch sphere.

Depolarizing: this channel introduces errors in a quantum state while preserving its overall structure. Depolarization refers to the flattening or averaging of the quantum state, making it less distinguishable from other possible states, leading to a loss of coherence and entanglement. Depolarization is typically quantified by a parameter λ , which represents the probability of an error occurring in the transmission or processing of the state transforming it into the maximally mixed state:

$$\text{Dep}_\lambda(\rho) = (1-\lambda)\rho + \lambda\mathbb{I}.$$

The depolarizing channel acts on a single-qubit state ρ by applying a Pauli operator (X , Y ,

or Z) with equal probability $\lambda/4$. The Kraus operators that define this channel are:

$$\begin{aligned} E_0 &= \sqrt{1 - \frac{3\lambda}{4}} \cdot I, & E_1 &= \sqrt{\frac{\lambda}{4}} \cdot X, \\ E_2 &= \sqrt{\frac{\lambda}{4}} \cdot Y, & E_3 &= \sqrt{\frac{\lambda}{4}} \cdot Z. \end{aligned}$$

The depolarizing channel can be visualized as a contraction of the Bloch sphere toward the center (Figure 2.19). For example, a pure state that is subjected to the depolarizing channel is shrunk by a factor $(1 - \lambda)$.

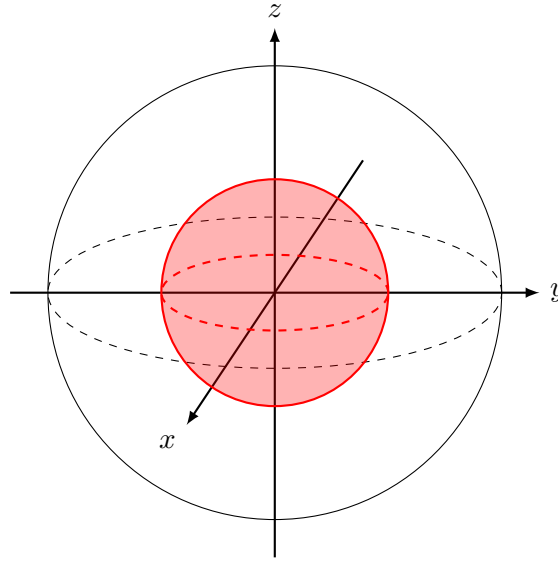


Figure 2.19: Representation of the effect of the depolarizing channel on the Bloch sphere.

Pauli: this channel is a generalization of the depolarizing channel where the probability of applying a Pauli X , Y or Z is respectively p_X , p_Y and p_Z :

$$Pauli(\rho) = (1 - p_X - p_Y - p_Z)\rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z.$$

Amplitude Damping: this channel models physical processes, occurring on the qubits, that involve energy dissipation or spontaneous emission. The amplitude damping channel describes a decay process where the state $|0\rangle$ is conserved while $|1\rangle$ decays to $|0\rangle$ with probability γ :

$$Damp_\gamma(|1\rangle) = (1 - \gamma) |1\rangle + \gamma |0\rangle.$$

The Kraus operators of this channels can be found to be:

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix} \quad E_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}$$

The amplitude damping channel shrinks the Bloch sphere toward the $|0\rangle$ state, a visualization is reported in Figure 2.20.

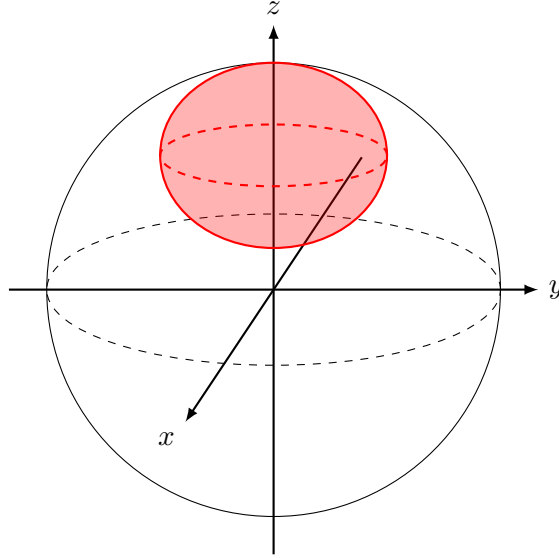


Figure 2.20: Representation of the effect of the amplitude damping channel on the Bloch sphere.

Thermal relaxation: this channel is used to model the interaction of a quantum system with a thermal environment resulting in relaxation events. The parameters of the channel take into account the qubit relaxation time T_1 , the dephasing time T_2 and the time elapsed during the action of the thermal relaxation channel t . In a quantum device T_1 and T_2 can be extracted from calibration, while t refers to the gate execution time or the total time needed for the computation. The thermal relaxation channel is a combination of both an amplitude damping and a phase damping channels with the respective parameters γ and p_p :

$$\gamma = 1 - \exp\left(-\frac{t}{T_1}\right),$$

$$p_p = 1 - \exp\left(-t \cdot \frac{2T_1 - T_2}{T_1 T_2}\right).$$

In order for the parameters to be valid the relation $T_2 \leq 2T_1$ must be satisfied. Because of its versatility the thermal relaxation channel is highly employed to formulate realistic noise models.

Another important class of errors that may arise during a quantum computation are the so called coherent errors. these kind of errors are unitary thus they don't require a superoperator representation. Usually they are represented as single-qubit rotations (R_x , R_y , R_z gates). They can arise for several reasons like a wrong calibration, cross-talk, external fields, and residual qubit-qubit interactions [161]. For example, when a rotation gate ($R(\theta)$) is applied to a qubit, a coherent error can introduce an unwanted perturbation in the rotation angle ($R(\theta + \epsilon)$). Because of their different nature, coherent errors can have strong effects on quantum error correction codes [162, 163]. On the other hand, as these errors do not destroy quantum information, once identified, they can be corrected [161, 164, 165].

The noise channels described so far are useful to model physical processes that can lead to errors during a quantum computation. Usually, in order to describe realistically the noise of a quantum

device, it is necessary to use different noise channels. Trying to reproduce the noise pattern requires to take into account all the possible noise sources and is a difficult challenge that has not been solved yet. We will delve deeper into this problem with the work presented in Section 3.2.5.

2.3.4 Principles of quantum error correction

Quantum Error Correction (QEC) is the quantum analogue of classical theory of error correction codes. The fundamental problem to be overcome is communication and information storage in the presence of noise. Both in classical and quantum systems, noise is due to the interaction with the environment. There are two main problems concerned with QEC that make their construction more difficult than the classical case. The first is that quantum information cannot be copied because of the no-cloning theorem (Appendix A). The second problem is that, while in classical digital information the possible errors form a discrete set, this is not true in quantum information, where errors form a continuous set. Even though these problems are present, this section and Section 2.3.5 will describe how to construct codes that protect quantum information against general errors.

In 1995, Shor discovered the first QEC code [166]. He found a nine-qubit code that could protect the information of a single logical qubit against general errors. Consider two observers, Alice and Bob, who want to communicate through a quantum noisy channel. If Alice wants to send one qubit of information using the nine-qubit Shor's code, the encoding of the logical information should be the following:

$$\begin{aligned} |0_L\rangle &= \frac{1}{2\sqrt{2}} [|000\rangle + |111\rangle] [|000\rangle + |111\rangle] [|000\rangle + |111\rangle], \\ |1_L\rangle &= \frac{1}{2\sqrt{2}} [|000\rangle - |111\rangle] [|000\rangle - |111\rangle] [|000\rangle - |111\rangle]. \end{aligned}$$

The expressions $|0_L\rangle$ and $|1_L\rangle$ refer to the logical information encoded in the nine qubits. The information encoding can be done with the first part of the circuit reported in Figure 2.21. Consider Alice sending $|0_L\rangle$ to Bob and an X error occurs on the first qubit of the code. The state received by Bob before correction ($|0_B\rangle$) is reported in Equation 2.18, only the first block of three qubits is considered as the other two blocks are not modified.

$$|0_B\rangle = \frac{1}{\sqrt{2}} [|100\rangle + |011\rangle]. \quad (2.18)$$

In the first part of the decoding circuit (Figure 2.21, before the red dashed line), the state of the first qubit is restored to the state $|0\rangle$. It can be verified that the same state would have been obtained if the X error had affected the second or third qubit of the block. This process is repeated on the other blocks of three qubits to correct any bit flip error on the first qubit of each block.

The correction of dephasing errors is performed by the second part of the decoding circuit (after the dashed line). Consider a Z error on a qubit in the first block of the state $|0_L\rangle$ sent by Alice, the state received by Bob becomes:

$$|0_B\rangle = \frac{1}{2\sqrt{2}} [|000\rangle - |111\rangle] [|000\rangle + |111\rangle] [|000\rangle + |111\rangle].$$

After the dashed line, and eventually correcting a bit flip error, we can consider only the first, fourth

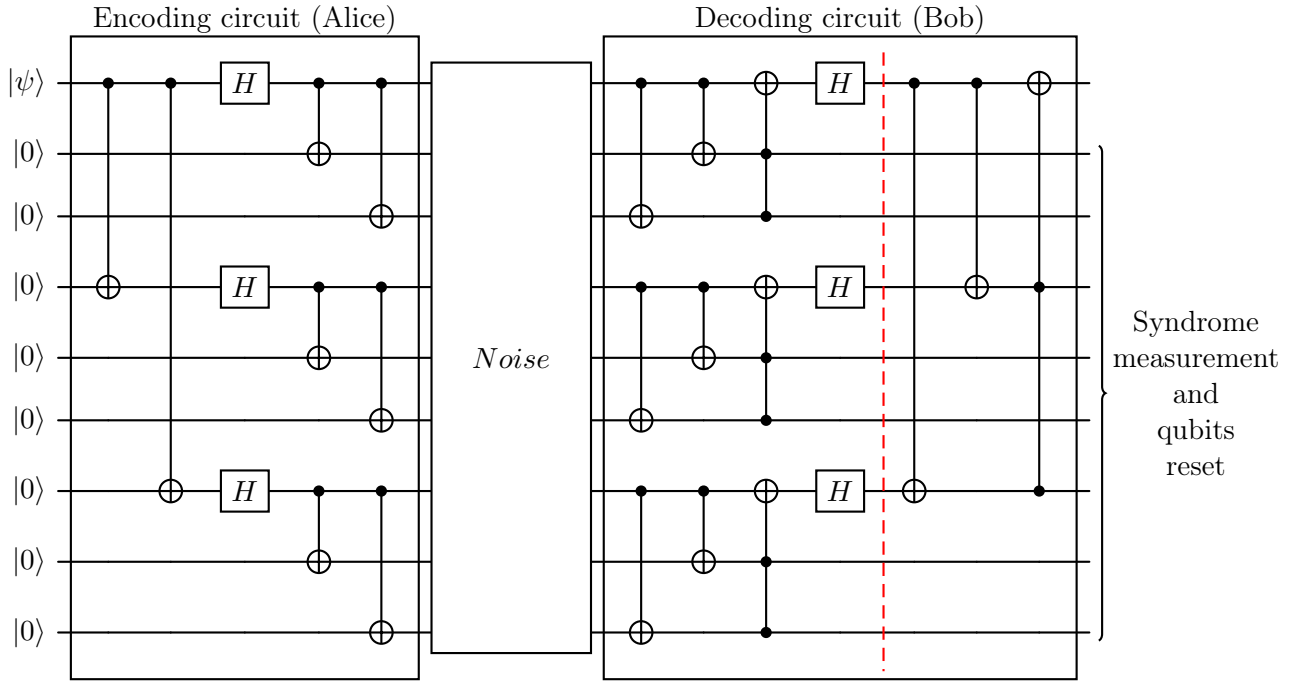


Figure 2.21: Nine-qubit Shor's error correction code circuit. The circuit has been divided into the encoding and decoding circuits. In the decoding circuit a red dashed line separates the parts of the circuit in charge of finding and correcting X and Z errors.

and seventh qubits. The state of these qubits is:

$$|0'_B\rangle_{1,4,7} = |100\rangle_{1,4,7},$$

the last part of the circuit restores the state of the first qubit to $|0\rangle$ in the same way as X errors are corrected in the first part of the circuit. It can be verified that the same state would have been obtained if the Z error had affected a qubit in the second or third block.

The encoding and decoding circuits presented in Figure 2.21 are unitary. This may lead to the erroneous assumption that quantum error correction can be performed without dissipating energy. When the system of physical qubits interacts and becomes entangled with the environment, its entropy increases. The entropy of the system can be restored only by transferring it to the ancillary qubits. The unitary transformation that restores the original state shifts the entanglement with the environment from the physical qubits to the ancillary qubits. In order to protect quantum information, a continuous supply of ancillary qubits should be provided. To use again the ancillary qubits it is necessary to erase them and this process requires energy. Thus it is not possible to implement quantum error correction without dissipating energy.

The process of measuring ancillary qubits, known as syndrome extraction, provides information about the errors. For instance, in Shor's code, an X error on the first qubit results in the final state of the second and third qubits being $|11\rangle$. In contrast, in the absence of errors, their final state would be $|00\rangle$. It's straightforward to verify that an X error acting on another qubit would lead to a different final state of the ancillary qubits, thus yielding a different syndrome. Conversely, a type Z error acting on any qubit within the same block generates an identical syndrome. A QEC code,

where the syndrome does not allow for the precise identification of the error, is called degenerate. In contrast, in a non-degenerate code, each unique syndrome is associated with a distinct error. Shor's QEC code is degenerate and inefficient, in fact, it is possible to protect the information of one logical qubit using only five physical qubits [167]. The minimum number of ancillary qubits necessary to protect quantum information against errors is known as the quantum Hamming bound, more details are reported in Appendix E.

2.3.5 Noise digitalization

In the Shor's QEC code presented in Section 2.3.4 we have shown how it is possible to correct X and Z errors. The correction of these errors is sufficient to save the logical state against a general single-qubit error, the demonstration is general.

It is crucial to note that a type Y error, generated by the application of the Pauli Y gate, is simply the sum of a type X and a type Z error. Therefore, the subsequent correction of X and Z errors also corrects Y errors. To understand how a general error operates consider a bipartite quantum system described by the Hilbert space $\mathcal{H}_1 \oplus \mathcal{H}_E$. The first system consists of the qubits, while the second system represents the environment. An error occurs when there is a coupling between the qubits and the environment. This process can be described by a unitary operator (U) acting on the state $|\psi\rangle_1 |0\rangle_E$:

$$U |\psi\rangle_1 |0\rangle_E = |\phi\rangle_1 |e\rangle_E.$$

Using Kraus formalism it is possible to observe that the transformation U acts, on the computational basis states, as:

$$\begin{aligned} U |0\rangle_1 |0\rangle_E &= |0\rangle_1 |e_1\rangle_E + |1\rangle_1 |e_2\rangle_E, \\ U |1\rangle_1 |0\rangle_E &= |0\rangle_1 |e_3\rangle_E + |1\rangle_1 |e_4\rangle_E. \end{aligned}$$

The states $\{|e_i\rangle_E\}$ are not necessarily orthogonal or normalized. Given the state $|\psi\rangle_1 = \alpha |0\rangle_1 + \beta |1\rangle_1$ the result of the transformation is:

$$U |\psi\rangle_1 |0\rangle_E = \alpha |0\rangle_1 |e_1\rangle_E + \alpha |1\rangle_1 |e_2\rangle_E + \beta |0\rangle_1 |e_3\rangle_E + \beta |1\rangle_1 |e_4\rangle_E \quad (2.19)$$

It is now necessary to introduce the following states:

$$|e_I\rangle := \frac{1}{2}(|e_1\rangle + |e_4\rangle), \quad |e_Z\rangle := \frac{1}{2}(|e_1\rangle - |e_4\rangle), \quad |e_X\rangle := \frac{1}{2}(|e_2\rangle + |e_3\rangle), \quad |e_Y\rangle := \frac{1}{2}(|e_2\rangle - |e_3\rangle).$$

With some manipulations, Equation 2.19 becomes:

$$U |\psi\rangle_1 |0\rangle_E = |e_I\rangle_E |\psi\rangle_1 + |e_X\rangle_E \sigma_x |\psi\rangle_1 + |e_Z\rangle_E \sigma_z |\psi\rangle_1 + |e_Y\rangle_E \sigma_y |\psi\rangle_1.$$

The measurement of the syndrome projects the state into one of the four possible states. Thus, the correction of X and Z (and consequently Y) errors restores the original state. The collapse of the wavefunction, after the projective measurement, reduces a general error to a finite number of possible errors that can be corrected. However, it is important that the measurement of the syndrome does not provide information about the logical state, or it would be irreversibly modified.

To protect information against multiple errors, we need to introduce more general concepts. The three Pauli matrices, along with the identity, form a basis for the space of 2×2 unitary matrices. Hence, any single-qubit unitary transformation can be expanded in terms of Pauli operators. Similarly, any unitary transformation acting on n qubits can be expanded in terms of tensor products of Pauli operators:

$$\{X, Y, Z, I\}^{\otimes n}.$$

With considerations similar to those for single-qubit errors, we can describe any process that couples a system of qubits with the environment as follows:

$$U |\phi\rangle_1 |0\rangle_E = \sum_k (E_k |\phi\rangle_1) |e_k\rangle_E.$$

Where, the states $\{e_k\}$ are neither orthogonal nor normalized, and the operators $\{E_k\}$, which implement a transformation of the qubits, can be written as tensor products of Pauli operators. In this section, $\{E_k\}$ should not be confused with Kraus operators. The syndrome measurement makes the system collapse into the state $(E_a |\phi\rangle_1) |e_a\rangle_E$, and errors can now be corrected by applying the operator E_a on the system of qubits. This procedure is called the digitalization of noise, and shows that it's possible to correct a continuous set of errors acting on many qubits just by correcting only X and Z errors acting on single qubits.

The weight of an operator E_a is defined as the number of qubits where it acts non-trivially. A code that can correct up to t errors is able to correctly identify and reverse the action of an operator E_a with weight less than or equal to t . Then the set of correctable errors (ε) is defined as:

$$\varepsilon := \{E_a \mid \text{Weight}(E_a) \leq t\}.$$

Consider two states of the physical qubits encoding logical states, $|i\rangle$ and $|j\rangle$, and two different errors $E_a, E_b \in \varepsilon$, for a non-degenerate code:

$$\langle i |_L E_a^\dagger E_b |j\rangle_L = \delta_{ab} \delta_{ij}.$$

It is possible to obtain a similar relation also for degenerate codes [168].

The concepts introduced in the last two sections are useful to understand a special class of QEC codes known as topological surface codes. This kind of codes is the main topic of the work proposed in Section 4.1.

2.4 Qibolab and the transpiling problem

During my PhD experience, I had the opportunity to collaborate with Quantum Research Center of the Technology Innovation Institute in Abu Dhabi². This research center is focused on the development of quantum technologies, from the characterization and design of hardware qubits to the development of quantum algorithms.

An interesting project that I had the opportunity to take part in is the development of the Qibo framework. This project is divided into three main parts: Qibo, Qibolab, and Qibocal. Qibo is a quantum computing simulator for the simulation of quantum circuits on classical computers. Qibolab is a software library that allows for the control of quantum computing hardware (superconductive qubits). Finally, Qibocal is a software library that implements calibration procedures for superconductive qubits.

In this project, my main contribution was the development of a transpiler for Qibolab. The code has now been moved to the Qibo repository and is available on GitHub³. The development of Qibolab has led to a paper published in 2024 [35].

In this section, the part of the paper regarding the transpiler is included, along with the abstract, introduction and conclusions to provide context for its development.

The development of efficient tools for the simulation, control and calibration of quantum hardware is essential to accelerate research in quantum computing. Moreover, the transpiling of quantum circuits is an interesting problem that can be tackled using both classical algorithms and machine learning techniques. For these reasons I believe that the work presented in this section is relevant in offering a complete overview of the current challenges and progress in quantum computing research.

2.4.1 Abstract

We present Qibolab, an open-source software library for quantum hardware control integrated with the Qibo quantum computing middleware framework. Qibolab provides the software layer required to automatically execute circuit-based algorithms on custom self-hosted quantum hardware platforms. We introduce a set of objects designed to provide programmatic access to quantum control through pulses-oriented drivers for instruments, transpilers and optimization algorithms. Qibolab enables experimentalists and developers to delegate all complex aspects of hardware implementation to the library so they can standardize the deployment of quantum computing algorithms in an extensible hardware-agnostic way, using superconducting qubits as the first officially supported quantum technology.

2.4.2 Introduction

A successful deployment of quantum computing algorithms requires quantum hardware and middleware software dedicated to instrument control for specific quantum platform technologies.

The goal of middleware is to provide standardized software tools which abstract heterogeneous software interfaces from high-level applications. From quantum computing algorithms based on the quantum circuit paradigm, to low-level driver instructions dedicated to a specific experimental setup including instruments.

²<https://www.tii.ae/quantum>, 2025.

³<https://github.com/qiboteam/qibo>, 2025.

A proper implementation of middleware software accelerates research from theory to experiments by reducing the amount of effort and expertise required to operate a quantum platform and develop novel quantum algorithms.

Nowadays, the major challenges of middleware, as a research accelerator, include the need of standard code procedures for quantum control algorithms, calibration and characterization, all extensively tested and reviewed. This software should be designed in such a way that it could be reused by similar experiments in multiple research laboratories dedicated to quantum hardware design and fabrication. Therefore, one of the expected positive side effects of the development of middleware is the generation of a database of algorithms and procedures built and maintained by a large research community. As an example, it is possible to find similar cases in other research fields such as data analysis tools [169] and Monte-Carlo event generators [170] for high-energy physics and artificial intelligence [171].

Since the beginning of 2020, despite the growing interest in quantum computing and the recent developments in quantum hardware platforms, we have observed the lack of a standard middleware open-source framework dedicated to self-hosted quantum platforms. There are software libraries dedicated to quantum computing such as `Cirq` [172] and `TensorFlow Quantum` [173] from Google, `Qiskit` [174] from IBM, `PyQuil` from Rigetti [175], among others [176–179]. However, many of these software libraries have been promoted just to grant users access to freeware and/or commercial cloud-based platforms, hence no full-stack open-source library for quantum algorithms, from simulation to quantum hardware control was available.

Moreover, specialized quantum hardware solutions such as `QCodes` [180], `PyCQed` [181] or `Labber` [182] offer too rigid a structure to seamlessly incorporate all the other essential features that a full-stack solution requires.

Therefore, we started developing Qibo, an open-source middleware framework for quantum computing, by establishing an international collaboration network involving laboratories in universities and research institutions located in Europe, Asia and America.

In this manuscript we present for the first time Qibolab, a software library which unlocks Qibo’s potential to execute quantum algorithms on self-hosted quantum hardware platforms. We provide a dedicated application programming interface (API) for quantum circuit design, qubit calibration, instrument control through arbitrary pulses, driver operations including sweepers and transpilation into a given platform topology using its native gates. A successful implementation of Qibo will deliver to the research community a first prototype of extensible hardware-agnostic open-source *hybrid quantum operating system*, fully tested and benchmarked on superconducting platforms.

2.4.3 Transpiler

Logical quantum circuits for quantum algorithms are hardware agnostic. Usually, an all-to-all qubit connectivity is assumed, while most current hardware only allows the execution of two-qubit gates on a restricted subset of qubit pairs. Moreover, quantum devices are restricted to executing a subset of gates, referred to as native [183]. This means that, in order to execute circuits on a real quantum chip, they must be transformed into an equivalent, hardware-specific circuit.

The transformation of the circuit is carried out by the transpiler through the resolution of two

key steps: connectivity matching [184] and native gates decomposition [185]. In order to execute a gate between two qubits that are not directly connected, SWAP gates [186] are required. This procedure is called routing. As on NISQ devices, two-qubit gates are a large source of noise, this procedure generates an overall noisier circuit. Therefore, the goal of an efficient routing algorithm is to minimize the number of SWAP gates introduced. An important step to ease the connectivity problem is finding an optimal initial mapping between logical and physical qubits. This step is called placement.

The native gates decomposition in the transpiling procedure is performed by the unroller. An optimal decomposition uses the least amount of two-qubit native gates. It is also possible to reduce the number of gates of the resulting circuit by exploiting commutation relations [187], KAK decomposition [188], or machine learning techniques [189].

Qibolab implements a built-in transpiler with customizable options for each step. The main algorithms that can be used at each transpiler step are reported below with a short description. The initial placement can be found with one of the following procedures:

- Trivial: logical-physical qubit mapping is an identity.
- Custom: custom logical-physical qubit mapping.
- Random greedy: the best mapping is found within a set of random layouts based on a greedy policy.
- Subgraph isomorphism: the initial mapping is the one that guarantees the execution of most gates at the beginning of the circuit without introducing any SWAP.
- Reverse traversal: this technique uses one or more reverse routing passes to find an optimal mapping by starting from a trivial layout [190].

The routing problem can be solved with the following algorithms:

- Shortest paths: when unconnected logical qubits have to interact, they are moved on the chip on the shortest path connecting them. When multiple shortest paths are present, the one that also matches the largest number of the following two-qubit gates is chosen.
- SABRE [190]: this heuristic routing technique uses a customizable cost function to add SWAP gates that reduce the distance between unconnected qubits involved in two-qubit gates.

Qibolab unroller applies recursively a set of hard-coded gates decompositions in order to translate any gate into single and two-qubit native gates. Single-qubit gates are translated into U3, RX, RZ, X and Z gates. It is possible to fuse multiple single-qubit gates acting on the same qubit into a single U3 gate. For the two-qubit native gates it is possible to use CZ and/or iSWAP. When both CZ and iSWAP gates are available the chosen decomposition is the one that minimizes the use of two-qubit gates.

The benchmarking of a full transpiling pipeline can be complex, as the results may vary across different chip architectures, and a trade-off between performance and execution time needs to be

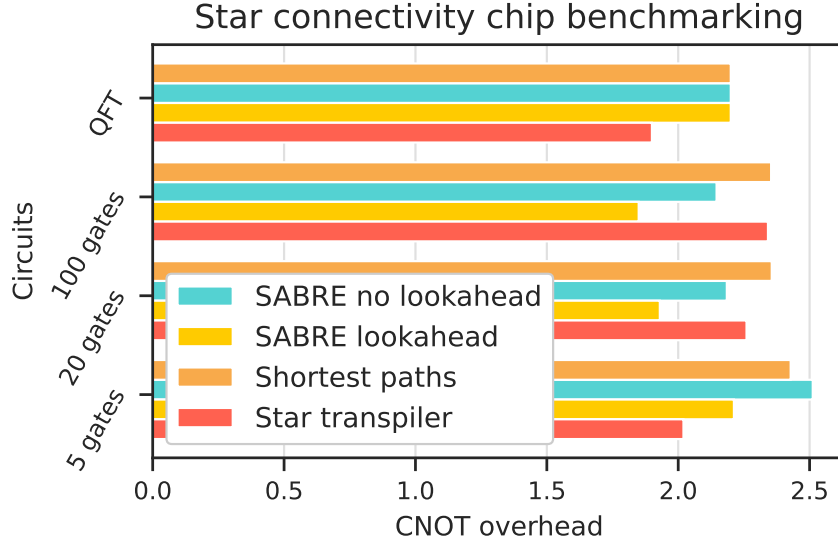


Figure 2.22: Benchmark of the performance of the built-in Qibolab transpilers (routing pass) evaluated considering the CNOT overhead. Circuits of different types have been considered in order to test the transpilers on both structured and unstructured circuits. In particular we have considered a five qubits Quantum Fourier Transform (QFT) circuit and random circuits with 5, 20 and 100 gates. The results for random circuits have been averaged over 50 circuits.

taken into account. We refer the general benchmarking problem to the specific literature [191] and focus on a more specific use-case.

Figure 2.22 reports the performance of the routing pass algorithms implemented in Qibolab on a five-qubit chip with a star connectivity. In this kind of chip, five qubits are arranged with a central qubit connected to all the remaining four qubits. The algorithm performance has been evaluated as the CNOT overhead, which is the number of CNOT gates in the routed circuit divided by the number of CNOT gates in the original circuit. The algorithm performance has been tested on five-qubit circuits composed of 10, 20, and 100 CNOT gates, taking an average over 50 random circuits. Moreover, we have tested the algorithms on a structured circuit: the five-qubit QFT. The SABRE algorithm has been tested with and without the lookahead. The results have been compared with a transpiler designed for that connectivity (star transpiler) that swaps the central qubit in the chip based on the successive gate. All the routing algorithms have been tested starting from an initial trivial layout except for the star transpiler that has a built-in placer, which explains the better performance of this algorithm on short circuits.

Figure 2.22 shows that SABRE with a lookahead reaches the best performance on the star connectivity chip. The execution time on this simple case is not significant as all algorithms perform in a fraction of a second even for the longer circuits. However, other tests have shown that the scalability of SABRE is better than shortest paths as the number of possible shortest paths increases drastically with the number of qubits in highly connected chips.

In summary, the Qibolab transpiler shows good performance in making abstract quantum circuits executable on small NISQ devices. In the future, we aim to develop new efficient and scalable algorithms for the next generation quantum chips with a high number of qubits. An interesting approach for this problem regards the use of machine learning techniques, in particular reinforcement learning [192].

Chapter 3

Machine Learning

The creation of an artificial general intelligence, an algorithm capable of learning and thinking as a human being, has always been an attractive topic for mankind. Research in this area has not yet succeeded in the creation of this kind of system. However, the progresses have led to the development of Machine Learning (ML) algorithms, capable of solving specific tasks by learning from a set of examples. An important class of ML algorithms is the artificial Neural Network (NN), where the computation is performed by a network of computational units called neurons. Although the first research in this field dates back to the second half of the last century, neural networks have not found real applications until the last few decades because they required too much computational power. In recent years, the progress on Graphic Processing Unit (GPU)s, able to parallelize a great number of calculations, have allowed the application of neural networks to real-life problems [193]. In order to solve difficult tasks with great efficiency, the complexity of neural networks has increased, leading to a new branch of artificial intelligence, deep learning.

This Chapter is organized in two main sections.

The fundamental concepts of ML are reviewed in Section 3.1.

In Section 3.2, we introduce more advanced ML techniques and models, which find applications in quantum computing as discussed in Chapter 4.

3.1 Fundamentals

This section gives an introduction to ML. Section 3.1.1 reports the definition of machine learning as well as an analysis of the main differences with traditional programming. The main tasks, field of application and learning paradigms of ML are reviewed in Section 3.1.2 and Section 3.1.3. Section 3.1.4 introduces general concepts like the loss function, gradient descent, overfitting and underfitting using the simple example of a linear regression. Section 3.1.5 explains the principle of operation of a NN starting from the artificial neuron. The simple feed forward architecture is then introduced as well as the backpropagation algorithm, necessary for training.

3.1.1 Comparison with standard programming

The term machine learning was coined by Arthur Samuel in 1959. The first definition was: *computational methods capable of emulating typical behaviours of humans, and the animal world in general,*

based on experience, without being explicitly programmed. This definition introduces the first fundamental difference between traditional programming and machine learning: ML algorithms are not programmed with strict rules; they learn them from experience. In traditional programming, data are analyzed following a set of instructions given by the programmer. On the other hand, with ML, a set of data and correct answers are given to the algorithm. The algorithm finds out the rules that link data to answers and can be applied to solve cases that didn't belong to the original set of data. These concepts are summarized in Figure 3.1.

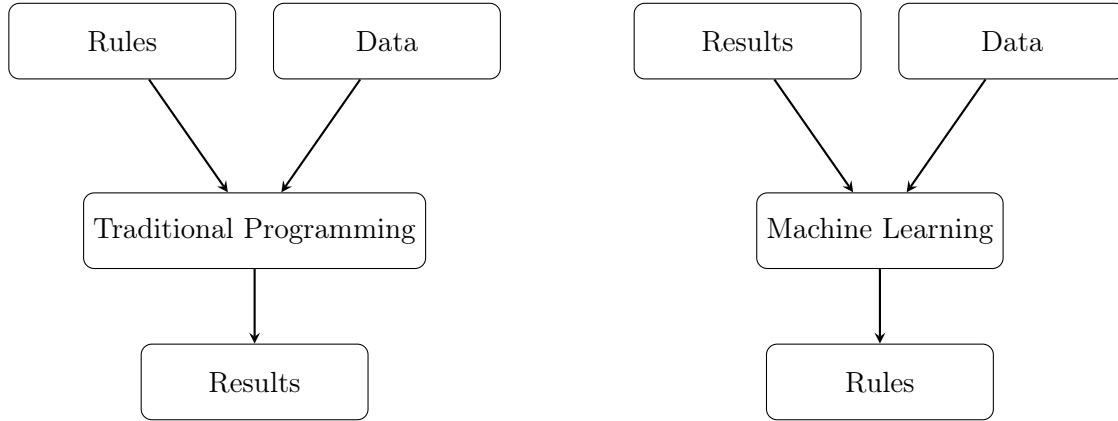


Figure 3.1: Schematic representation of the main difference between traditional programming (left) and machine learning (right).

Machine learning is a subfield of artificial intelligence, a field of research aimed at automating intellectual tasks typically performed by humans. The definition by Arthur Samuel may give the mistaken impression that ML aims to create algorithms capable of human-like thinking, known as artificial general intelligence. While such systems are at the frontier of research, machine learning algorithms are currently used for a wide variety of tasks. These ML algorithms cannot perform rational thinking but surpass classical algorithms in solving problems that are too complex to be described by a set of mathematical rules, such as image recognition or language processing.

A more formal definition of machine learning has been given by Tom Mitchell in 1998, it states that: *an algorithm is said to learn from experience (E) with respect to a given task (T) and a given measure of performance (P), when P in performing T improves with the E.*

To understand this definition it is necessary to explain with more detail the terms experience, task and performance.

Experience refers to all the empirical information given to the algorithm. The algorithm learns from a set of examples, usually a database of features with labels, called the training set. To speed up the training process and improve performance, it is useful to exploit all prior knowledge of the problem, such as invariants, correlations, and probability distributions.

Task means the kind of problem that the algorithm has to solve. The main differentiation can be made between classification, regression, clustering, ranking and generative tasks. These tasks will be described in detail in Section 3.1.2 with their fields of application.

Performance is a quantitative measurement of how well the ML algorithm solves the problem it has been trained for. To measure performance accurately, the algorithm must be tested on

new examples. For this purpose, the performance measurement is carried out on the test set, a set of known examples that don't belong to the training set. The performance measurement depends on the task. For example, accuracy, which is the percentage of correct answers, is widely used for classification problems.

Machine learning algorithms have their limits. The main limitation comes from the fact that learning is based on the training set, a database of features necessary to solve the chosen problem. If the data are corrupted or do not provide enough information, no algorithm can function properly. Another limitation arises from the need for prior knowledge in ML algorithms. This concept is highlighted by two important theorems. The “ugly duckling theorem” [194] (S. Watanabe, 1969) states that in the absence of any prior knowledge, there is no privileged feature representation, meaning that classification is not possible without some bias. Additionally, the “no free lunch theorem” [195] (D. Wolpert and W. Macready, 1997) states that any two ML algorithms are equivalent when their performance is averaged across all possible problems. These arguments demonstrate that the choice of data representation is crucial for any ML algorithm. In the earliest ML algorithms, the set of input features was chosen by the programmer based on the specific problem. With advancements in ML techniques, such as neural networks, the algorithm can automatically identify the best representation of the input features. This process is called representation learning. The most advanced algorithms are capable of organizing ideas and concepts hierarchically by building new representations of data from the initial representations. These techniques are known as hierarchical representation learning or deep learning.

3.1.2 Tasks

Machine learning algorithms have the advantage of being very flexible and can be adapted to solve various kinds of problems. The tasks can be divided into five fundamental classes: classification, regression, clustering, ranking, and generation. Below is a brief overview of these tasks along with possible applications.

Classification

In classification problems, there is a limited set of possible predicted outputs. Usually, algorithms must assign a label to an element by analyzing the features that describe it. For example, email providers use a classification algorithm for spam detection. The algorithm uses the text of the email (features) to classify it as spam or not (label). This is a binary classification problem since there are only two possible outputs. An example of a multi-class classification problem, where there are more than two possible outputs, is the recognition of handwritten digits. An algorithm solving this problem has to analyze an image (a matrix of pixels) and label it with a number from zero to nine. Classification algorithms find application in almost every field. Two important examples come from Medicine and Physics. In Medicine, classification algorithms are being tested for image detection and segmentation, for example from magnetic resonance [196]. In high energy physics, classification algorithms are employed in colliders to classify particles and decays [197].

Regression

In regression problems, the goal is to find a useful approximation to the function that underlies the relationship between the inputs and outputs. Regression algorithms have a continuous set of possible outputs, usually a numerical value, but it may also be the description of an image. For example, in product price prediction, a linear or polynomial regression algorithm is used to forecast the selling price of a product. The algorithm outputs a price (continuous value) based on the characteristics of the product by learning from similar products that have already been sold. The main fields of application of regression algorithms are risk assessment, score prediction, market forecasting, and other predictions like weather, population growth, and life expectancy. In high energy physics, regression algorithms can be used to estimate particle parameters from decay data such as energy, mass, or decay time [198].

Clustering

In clustering problems, the goal is to detect potentially useful groups of similar input data. A more sophisticated task is to determine the distribution of data within the input space, known as density estimation. Understanding the data probability distribution is crucial for identifying outliers, which forms the basis of anomaly detection algorithms. Clustering and related algorithms are widely used in marketing for customer segmentation to enhance recommender systems. Other applications include medicine, where clustering helps identify similar symptoms to improve diagnosis. In high energy physics, clustering and anomaly detection algorithms play a significant role in identifying rare events [199]. This class of problems often involves projecting data from a high-dimensional space into two or three dimensions for better visualization. Dimensionality reduction algorithms [200] are employed for this purpose.

Ranking

In ranking problems, the goal is to maximize a numerical reward signal. The algorithms employed for these problems do not have a fixed dataset; they interact with an environment and make decisions. Typically, these algorithms are trained using reinforcement learning, as discussed in Section 3.1.3. For instance, algorithms trained to play chess or similar games use input data comprising the positions of pieces on the board. The algorithm then selects the best move in each turn to achieve victory. Another example involves self-driving cars, where algorithms must make real-time decisions. Ranking algorithms find applications in all decision-making tasks. One significant application is in robotics for autonomous operation and intelligent decision-making. This is crucial as autonomous robots can be deployed in hazardous situations without risking human lives.

Generation

Generative algorithms have the capability to create new samples with characteristics similar to those in the training dataset. This task typically involves performing probability density estimation and then sampling from this learned distribution. Some of the most impressive advancements in generative algorithms have been in natural language processing, enabling the generation of coherent text [201]. These algorithms are now capable of generating diverse samples including images, audio, and even video. The ability to generate new data is valuable for enhancing the quality of datasets

used to train various ML algorithms. In high energy physics, generative algorithms are extensively employed to simulate new events [202]. These simulations are crucial for predicting outcomes in complex collider experiments.

Multi-task

The classes of problems described represent a general partition of the most common tasks performed by ML algorithms. Sophisticated algorithms may need to accomplish more than one task. For example, to produce a complete self-driving algorithm, it is necessary first to classify the objects captured by the cameras (classification) and then determine the best decision to control the car (ranking). The ability to deconstruct a complex problem into a pipeline of simple tasks is a fundamental skill in programming ML algorithms. In recent years, there has been significant enthusiasm regarding the potential for creating ML algorithms capable of autonomously dividing a problem into smaller tasks using the so-called “divide-and-conquer” strategy [203, 204].

3.1.3 Learning paradigms

Different kinds of problems require different learning paradigms, mainly because the characteristics of the training set vary from problem to problem. For example, the correct answers may be known, as in most classification problems, or unknown, as in clustering tasks. There are three main learning paradigms: supervised, unsupervised, and reinforcement learning.

Supervised learning

This learning paradigm is used when the mapping between the input features and the correct output is known for the training set. The algorithm generates a function that is modified during the learning process to match the known input and output data. As the answers for the training set are known, this method is used to train algorithms designed to predict future outcomes, such as classification and regression algorithms. Dividing the data into training and test sets is fundamental for evaluating performance.

Unsupervised learning

This learning paradigm is used when the correct outputs for the input data are unknown. Algorithms trained with unsupervised learning are primarily used to analyze large datasets to uncover hidden structures. Clustering, dimensionality reduction, and density estimation are typical tasks to which unsupervised learning is applied. Since these algorithms work on the entire input data, dividing the data into training and test sets is not necessary.

Reinforcement learning

This learning paradigm is used when there is no fixed training set and the algorithm must interact with an evolving environment. Reinforcement learning is primarily developed to train algorithms for decision-making and ranking problems. During the learning process, the algorithm makes decisions and receives feedback based on the outcomes. For example, a self-driving algorithm receives a “reward” (positive score) for staying on track and a “penalty” (negative score) for a crash. With

sufficient training, the algorithm learns the best actions to maximize the final score. More details regarding this training paradigm can be found in Section 3.2.5.

The described learning paradigms are not the only possibilities for training a ML algorithm. For example, semi-supervised learning, a hybrid between supervised and unsupervised learning, is used for training sets where the correct outputs are known only for a few examples. Another approach is online learning, which updates the algorithm directly after a prediction, and is applied when the training set is a continuous stream of data. Online learning is commonly used in recommender systems for online marketing and can be applied to classification and regression problems where the function mapping inputs to outputs changes over time. With this learning paradigm, it is not necessary to save the entire training set. This section has introduced the main learning paradigms, tasks, and applications of ML algorithms. The many applications of these algorithms demonstrate their great flexibility. A schematic summary of the applications, related learning paradigms, and tasks is shown in Figure 3.2.

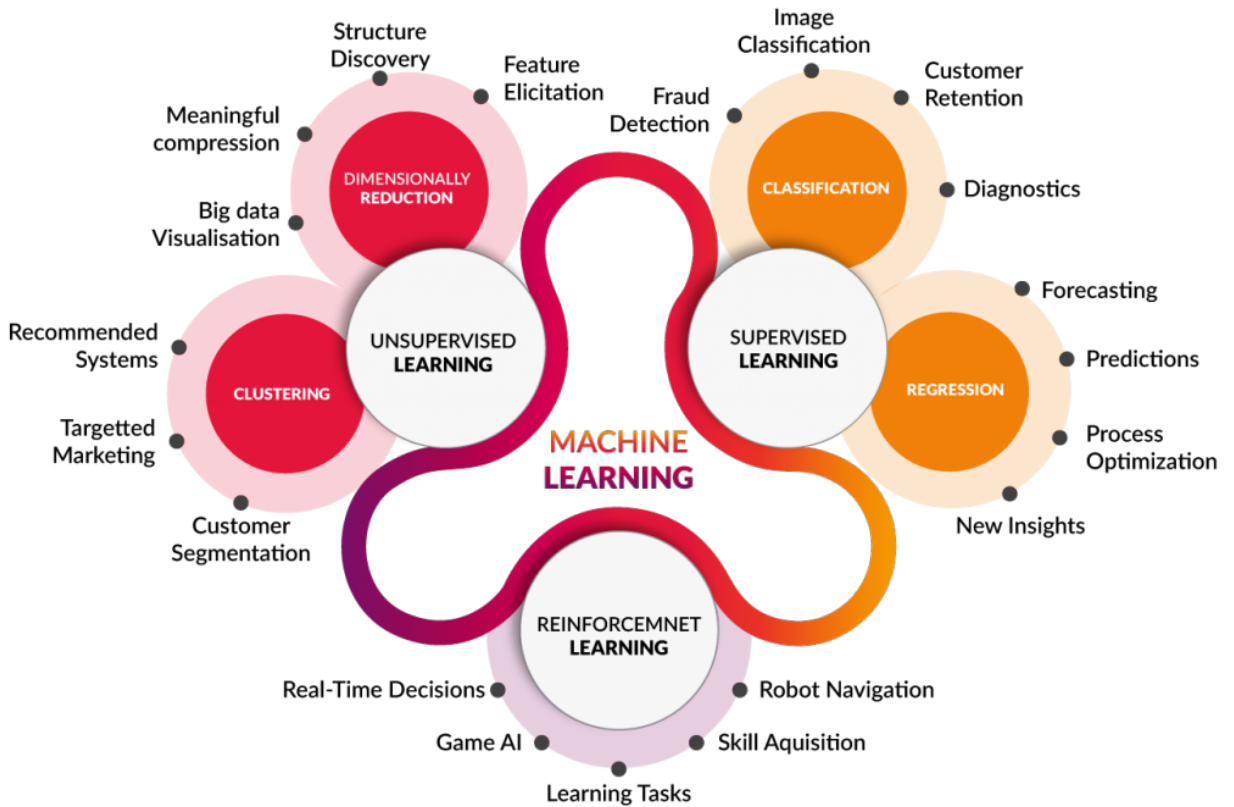


Figure 3.2: Schematic summary of the main learning paradigms, tasks and applications of machine learning.

3.1.4 Training

In this section we will review the basics of how it is possible to train a ML model. For this purpose we will consider a simple example, the linear regression algorithm, which is used to find a linear function that best approximates a given dataset. This function can then be applied to predict unknown values. Although it is possible to solve this problem analytically, in this section, we will adopt a ML approach. This example serves to introduce key concepts such as the loss function,

gradient descent, overfitting, and underfitting.

Consider the dataset shown in Figure 3.3, where a linear relationship within the data is apparent. This linear relationship is a bias, or an a priori assumption derived from observing the data. Given the input data x and the label y , the algorithm's objective is to find the optimal parameters A and B that fit the data in the form:

$$y = Ax + B.$$

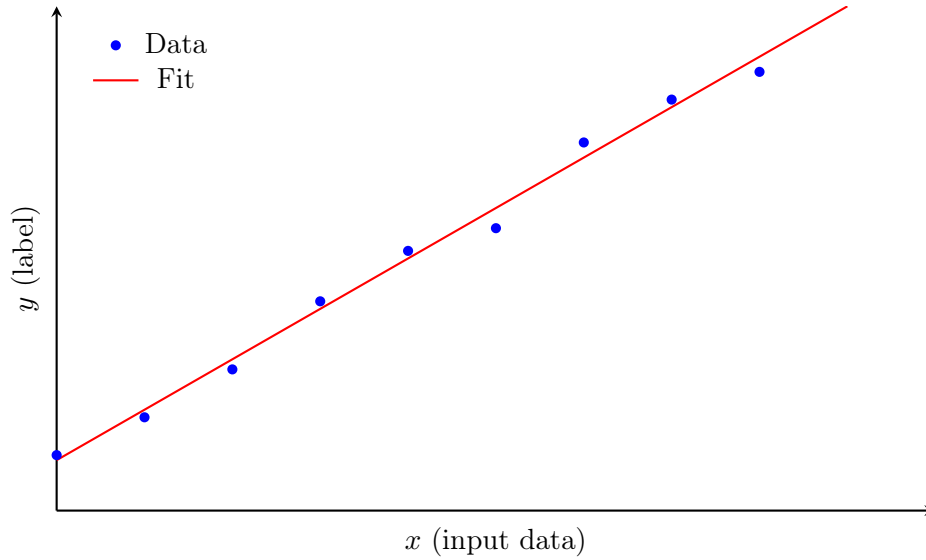


Figure 3.3: Example of a dataset for with a linear correlation between the input data x and the label y .

Loss function

The loss function, or cost function, is a mapping from an event or set of values to a real number called loss. The loss quantifies the penalty for an incorrect prediction, indicating how poor the model's prediction was. If the model's prediction is perfect, the loss is zero; otherwise, the loss is positive. The choice of a loss function depends on the task and is determined by the programmer. Some characteristics make certain loss functions preferable to others. For instance, continuity and differentiability are beneficial as they facilitate optimization, especially for algorithms that rely on computing gradients. The most important loss functions for classification and regression are outlined in Appendix F. Consider the linear regression problem, a good loss function should evaluate the total distance between the data and the fit. In this example, we will use the Mean Squared Error (MSE) loss function, as shown in Appendix F.

Given that predictions are made using the linear function $y_i = Ax_i + B$, the specific cost function becomes:

$$\text{MSE-Loss} = \frac{1}{N} \sum_{i=1}^N [(Ax_i + B) - y'_i]^2,$$

where $\{y'_i\}$ are the original data points. It is important to note that $\{y'_i\}$ and $\{x_i\}$ are fixed values, making the MSE a function of the parameters A and B . The goal is thus to find the optimal parameters A^* and B^* that minimize the loss function.

Gradient descent

Gradient descent is an optimization technique used to reach the minimum of a function through iterative steps starting from an initial random point. At each step, also known as epoch, the gradient of the function is calculated to identify the direction of steepest descent. The parameters are then adjusted in that direction until a minimum is found.

More formally, for a generic loss function J that depends on the parameters $\{\theta_i\}$, the parameters are updated at each epoch as follows:

$$\theta_i^t \rightarrow \theta_i^{t+1} = \theta_i^t - \alpha \frac{\partial J}{\partial \theta_i},$$

where the parameter shift is controlled by the hyperparameter α , known as the Learning Rate (LR). In linear regression, calculating the gradient involves two partial derivatives. The parameter updates at each epoch are given by:

$$A^t \rightarrow A^{t+1} = A^t - \alpha \frac{2}{N} \sum_{i=1}^N x_i [(Ax_i + B) - y'_i]$$

$$B^t \rightarrow B^{t+1} = B^t - \alpha \frac{2}{N} \sum_{i=1}^N [(Ax_i + B) - y'_i]$$

The learning rate is not explicitly present in the cost function, but its value is crucial for the convergence of the algorithm. In fact, a small learning rate requires more time for convergence and makes the algorithm more sensitive to local minima and saddle points. On the other hand, a high learning rate may cause the algorithm to diverge. An example of how gradient descent behaves differently depending on the learning rate is shown in Figure 3.4. These observations highlight the importance of fine-tuning the learning rate.

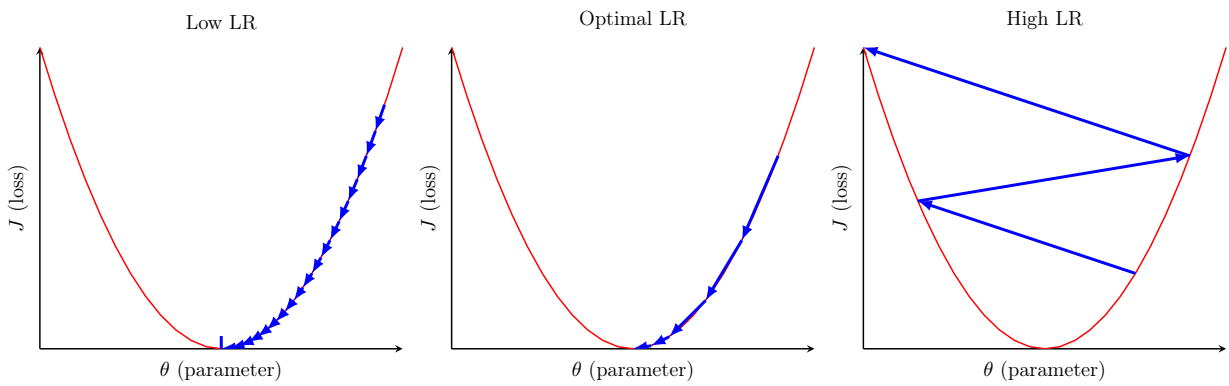


Figure 3.4: Consequences of the learning rate choice on the behavior of the gradient descent algorithm when optimizing the loss function $J(\theta)$.

With a low learning rate (left), the algorithm requires many updates to reach the minimum and is more sensitive to local minima.

An optimal learning rate (center) allows the algorithm to rapidly converge to the minimum.

A too high learning rate (right) results in drastic parameter updates, which may lead to divergent behavior.

More advanced optimization methods, which involve additional hyperparameters, are presented in Appendix H. The tuning of hyperparameters is usually performed by introducing a validation set, dividing the original dataset into three subsets. The training set is used to fit the parameters of the

model with various values of the hyperparameters. In the current example, this involves finding the optimal values of A^* and B^* for different learning rates α . After parameter fitting, a performance measurement is performed on the validation set to find the best tuning of the hyperparameters. In this example, we select the value of α that minimizes the loss function on the validation set. Finally, a performance evaluation is carried out on the test set using the best hyperparameter values, providing an unbiased assessment of the final model fit.

There is no strict rule for dividing the data into training, validation, and test sets, though it is essential to train the model on a substantial amount of data. Generally, the training set comprises about 60% – 80% of the initial dataset, with the remainder split equally between the validation and test sets.

Optimization methods based on gradient descent can encounter challenges with non-convex functions that contain multiple local minima. Alternative optimization approaches, which do not rely on gradient descent, include the Monte Carlo method [205], which performs a random sampling of the entire input variable space. While Monte Carlo methods are not sensitive to local minima, they can be inefficient for high-dimensional problems. Another approach involves genetic algorithms, which approximate solutions by applying evolutionary schemes inspired by genetic theory [206]. Simulated annealing [207] is also commonly used to approximate the global minimum in functions with numerous local minima, where gradient descent may otherwise become stuck.

Overfitting and underfitting

Consider the dataset shown in Figure 3.5. In this case, it is impossible to draw a straight line that fits the data well. Linear regression with a single variable has only two parameters, making it inadequate for modeling a complex relationship between inputs and outputs. This issue of having insufficient parameters to model the training data is a common machine learning problem known as

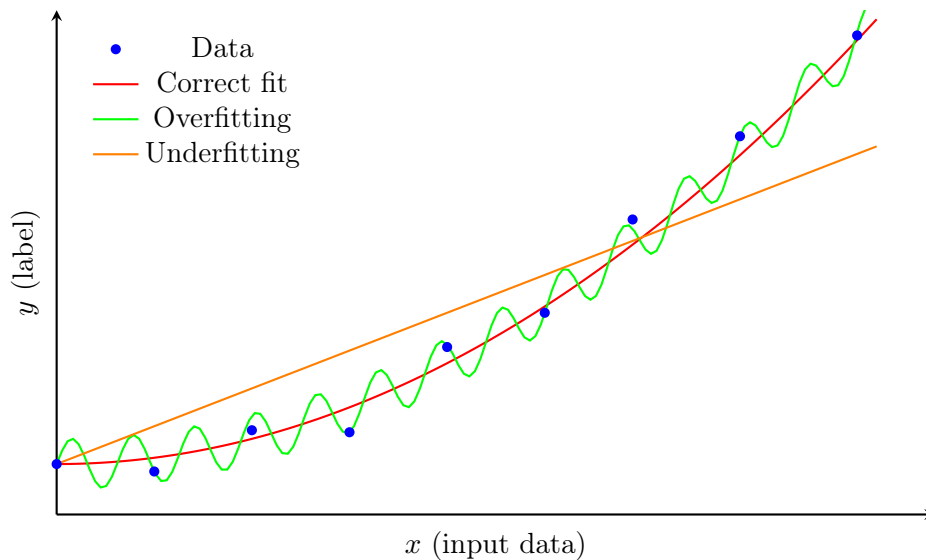


Figure 3.5: Dataset without a linear correlation. A linear function (orange) is too simple to fit the model (underfitting).

An high order polynomial (green) perfectly fits the data but is not a good predictor (overfitting).

A quadratic function (red) is a good trade-off between complexity of the model and generalization properties

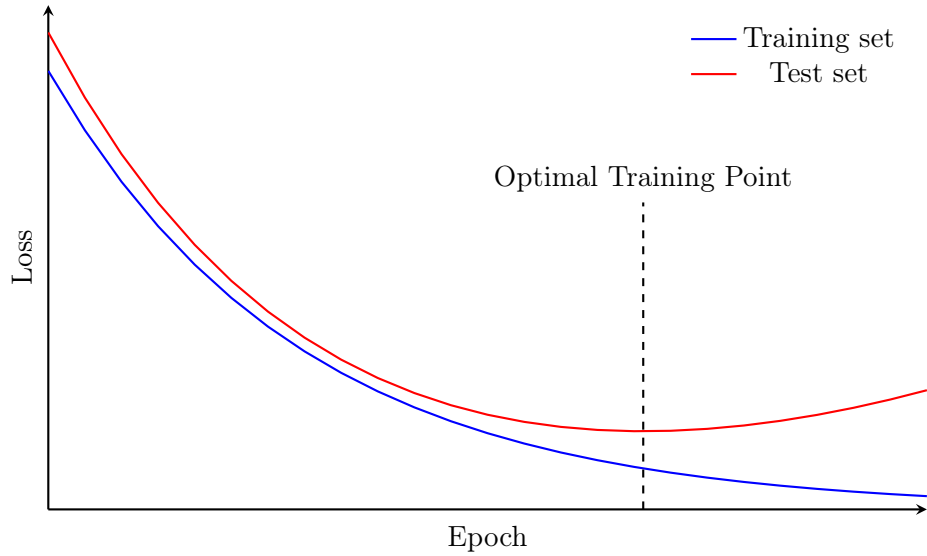


Figure 3.6: Common behavior of the loss function computed on the training and test sets during the training process. The optimal training epoch (early stopping) is at the minimum of the loss function evaluated on the test set.

underfitting or high bias. This problem can be addressed by adding more parameters to the model. In this example, polynomial regression may be used:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_n x^n$$

A polynomial fit of order n has $n + 1$ free parameters, enabling it to capture more complex relationships between input and output data. However, adding too many parameters introduces the risk of overfitting. The model may yield a fitted curve that exactly passes through the training data points but performs poorly on unseen data. This issue, where the model fits the training dataset too closely and lacks generalization, is known as overfitting or high variance.

An effective model must find a balance between overfitting and underfitting. The number of parameters and other hyperparameters that control this trade-off are chosen by the programmer based on testing and experience. In this example, a second-order polynomial might be a suitable model to fit the data.

Underfitting is primarily caused by an insufficient number of parameters in the model. It is also possible for a well-structured model to exhibit high bias if it has not been trained sufficiently. The problem of underfitting can be addressed by increasing the complexity of the model and/or extending the training duration.

The main cause of overfitting is often due to an excess of parameters. Reducing the number of parameters may help mitigate overfitting. However, it is also possible to use regularization methods that add a penalty term to the cost function for large parameter values. Regularization techniques commonly employed to avoid overfitting in NNs are detailed in Appendix I.

Overfitting may also result from excessive training. This effect can be observed by monitoring the algorithm's performance on the training set compared to the test set. Figure 3.6 shows the typical behavior of loss on these sets during training. The optimal point to stop training is when the

error on the test set reaches its minimum. Using a validation set for this purpose is often more effective, in a process called early stopping. This approach allows enough time for tuning parameters while preventing the algorithm from over-specializing on the test set, thus preserving generalization capabilities.

Finally, the size of the training set is crucial for avoiding overfitting. A complex model requires a large dataset; otherwise, the algorithm may end up “memorizing” the training set rather than learning generalizable patterns. In some cases, data augmentation techniques can be used to expand the dataset size [208]. For instance, in image classification tasks, new images can be created by rotating, flipping, or adding noise to the images in the original dataset. This approach is especially beneficial when obtaining a large dataset is challenging.

3.1.5 Neural networks

The human brain is a computational system with remarkable learning capabilities, enabling it to perform complex tasks by learning from experience. Consequently, many scientists have aimed to understand the brain’s mechanisms to reproduce them artificially in machines capable of autonomously performing increasingly complex tasks.

Today, we understand that the brain comprises a network of biological computational units called neurons. These neurons are organized in layers that process information from simple features to abstract concepts. For example, in image recognition, light is captured by the eye and converted into nerve impulses; the initial layers of neurons analyze simple geometric patterns. These patterns are then combined by subsequent layers to recognize objects, letters, or faces. The final layers interpret abstract concepts and link images to memory. This intricate structure grants the brain high learning capabilities and generalization properties that remain unmatched by artificial computational machines.

The concept of an artificial neural network was proposed by McCulloch and Pitts in 1943 [209], inspired by biological neural networks. Research in this area remained mostly theoretical throughout much of the last century, as computers lacked the processing power needed to handle the computations required by large NNs. However, in recent decades, NNs have become widely applied to practical problems thanks to the computational power offered by GPUs and TPUs.

Artificial neurons

The first model of an artificial neuron, known as the perceptron, was proposed by Rosenblatt in 1958 [210]. This computational unit takes multiple inputs, $\{x_1, x_2, \dots, x_n\}$, and produces a single binary output. The output is computed using weights, $\{w_1, w_2, \dots, w_n\}$, which are real numbers that represent the importance of each input. If the weighted sum of the inputs (z) exceeds a threshold (b), the output (y) is one; otherwise, it is zero:

$$y = \begin{cases} 0, & \text{if } \sum_j x_j w_j \leq b, \\ 1, & \text{if } \sum_j x_j w_j > b. \end{cases}$$

Modern artificial neurons can produce a continuous range of outputs. To compute this output, the weighted sum of the inputs is passed through a non-linear activation function, $f(z)$. The equivalent of the threshold in the perceptron model is now called the bias. To simplify notation, this bias is

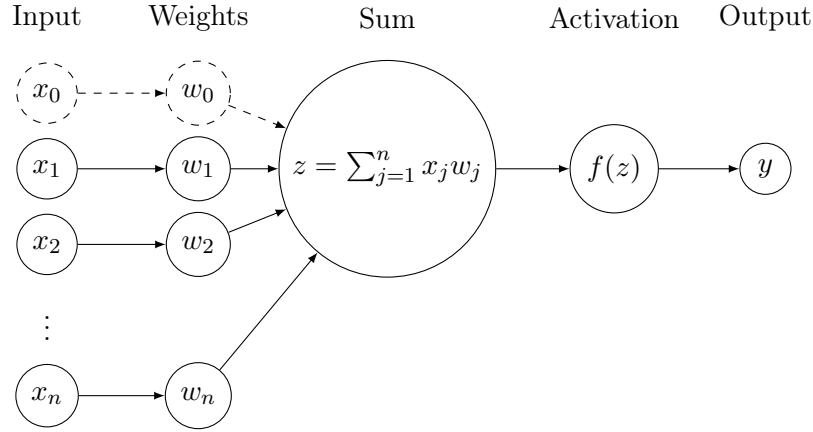


Figure 3.7: Schematic representation of an artificial neuron. The input signals $\{x_1, x_2, \dots, x_n\}$ are combined with weights $\{w_1, w_2, \dots, w_n\}$, and a bias input x_0 (shown with dashed lines). The weighted sum, denoted by z , serves as the input to the activation function $f(z)$, which then computes the output y . A perceptron model is obtained by using a step function as the activation function.

represented as a weight w_0 with a corresponding input $x_0 = 1$, so that:

$$y = f(z), \quad \text{where} \quad z = \sum_{j=0}^n x_j w_j.$$

A schematic representation of the artificial neuron is shown in Figure 3.7.

The choice of the activation function is important for the correct behavior of the neural network. The most commonly used activation functions are listed in Appendix G. A useful property for the activation function is differentiability, as it facilitates the implementation of the backpropagation algorithm, which will be covered in the next paragraphs. To accelerate the training process, it is also recommended to use activation functions that are centered around zero ($f(0) = 0$) and that do not saturate ($f'(x) \neq 0$ as $x \rightarrow \infty$).

Neurons can be combined to compute any possible logic function. This can be demonstrated by implementing a NAND gate, a universal gate that outputs a false logical value only if both inputs are true. For this demonstration, consider a neuron with a sigmoid activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}},$$

the proof can be adapted to other activation functions as well.

Neuron output values greater than $1/2$ ($z > 0$) will be interpreted as true, while values less than $1/2$ ($z < 0$) will be interpreted as false. Similarly, for inputs, a value of one is considered true and zero as false. Consider the following set of weights: $w_0 = 3$, $w_1 = w_2 = -2$. The truth table of the logic function computed by this neuron is shown in Table 3.1, which is equivalent to a NAND gate. This demonstration shows that a network of neurons can act as a universal computational system. However, neurons are not limited to acting as logic gates. The ability to adjust weights in response to external stimuli, without direct programming intervention, enables a neural network to learn to solve complex problems where explicitly defining a circuit of NAND gates would be impractical.

Table 3.1: Truth table of a NAND gate obtained using a neuron with a sigmoid activation function and weights: $w_0 = 3$, $w_1 = w_2 = -2$. For inputs: *True* = 1, *False* = 0. For outputs: *True* if $\sigma(z) > 1/2$, *False* if $\sigma(z) < 1/2$.

x_1	x_2	$z = \sum_{j=0}^2 w_j x_j$	$\sigma(z)$
0 (F)	0 (F)	3	$> 1/2$ (T)
0 (F)	1 (T)	1	$> 1/2$ (T)
1 (T)	0 (F)	1	$> 1/2$ (T)
1 (T)	1 (T)	-1	$< 1/2$ (F)

Feed forward neural networks

The simplest model of a NN is the Feed Forward Neural Network (FFNN). In this architecture, neurons are organized into layers, with information flowing only from one layer to the next. From a mathematical perspective, a FFNN is described by a directed acyclic graph. In this graph representation, shown in Figure 3.8, neurons are represented by circles, and information flow is represented by directed arrows. Typically, each neuron is connected to every neuron in the following layer; a network structured this way is called dense or fully connected. However, some connections may be removed to reduce model complexity, a strategy often used to avoid overfitting.

The first layer of a FFNN is called the input layer. It does not perform any calculations but simply contains the inputs to the neural network, $\{x_1, x_2, \dots, x_n\}$, so the number of neurons in this layer is determined by the number of input features. The last layer is called the output layer, and the number of neurons it contains is determined by the number of elements in the output vector, $\{y_1, y_2, \dots, y_k\}$. For example, in regression and binary classification, it consists of only one neuron, while in multiclass classification, it generally contains as many neurons as the number of total classes. For the input and output vectors we will use the superscript x^i and y^i to indicate the i^{th} element of the training set and the respective output.

Between the input and output layers, there is an arbitrary number of hidden layers. It is useful to denote the i^{th} neuron in the k^{th} layer with the notation a_i^k . The number of hidden layers and the number of neurons within each are important hyperparameters of the algorithm. Increasing these numbers adds complexity to the model but slows down the training process and increases the risk of overfitting.

A FFNN with just one hidden layer is called shallow. It is possible to show that this type of NN can approximate any function with arbitrary precision. The proof of this theorem, known as the universal approximation theorem (arbitrary width case), can be found, for sigmoid activation functions, in [211]. However, the number of neurons in the hidden layer increases exponentially with the dimensionality of the function.

A bounded version of this theorem, where the number of neurons does not grow exponentially, requires an arbitrary number of hidden layers. This is known as the arbitrary depth case; a proof for Lebesgue integrable functions, using ReLU activation functions (Appendix G), can be found in [212].

Before analyzing the training process of a FFNN, it is necessary to introduce additional notation. To distinguish between the different weights of neurons, we denote them as $w_{i,j}^k$, as shown in Figure 3.9.

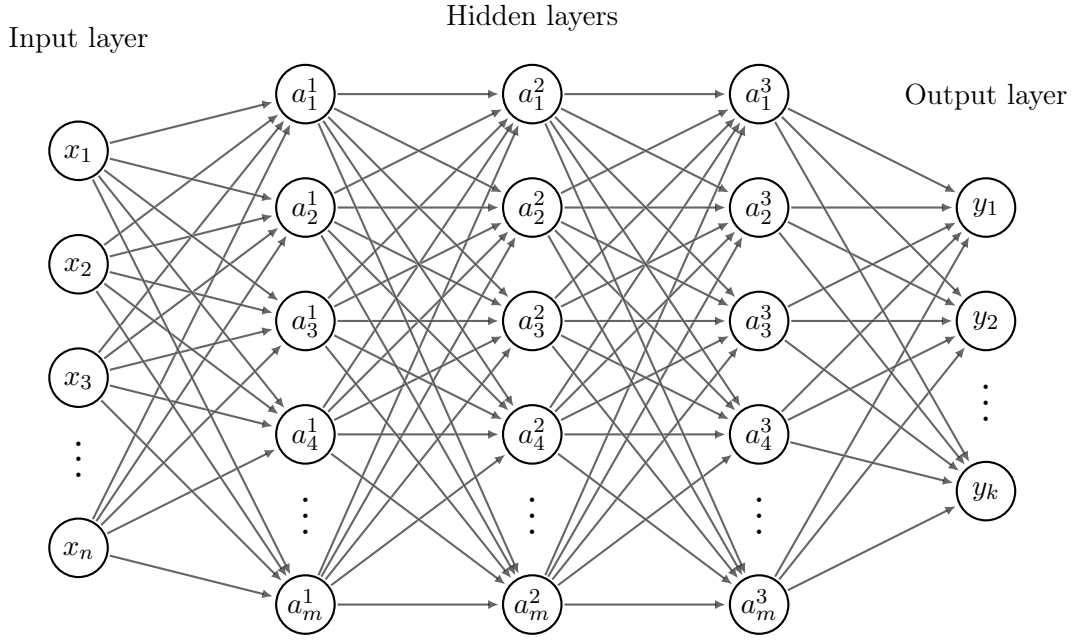


Figure 3.8: Graph representation of a dense FFNN with three hidden layers. The i^{th} neuron in the k^{th} layer is denoted as a_i^k .

- The superscript (k) indicates the layer to which the neuron belongs.
- The first subscript (i) denotes the number of the neuron in the current layer.
- The second subscript (j) denotes the number of the neuron in the previous layer.

With the notation w_i^k , we indicate the vector of weights for the i^{th} neuron in the k^{th} layer. The first element of this vector, $w_{i,0}^k$, is the bias.

Typically, in a NN, the same activation function is used for all neurons, except in the output layer. However, it is possible to use different activation functions in different layers or even for individual neurons within the same layer. The output of the activation function of the i^{th} neuron in the k^{th} layer is denoted as a_i^k , which is the same notation used to identify the neuron in Figure 3.8. To account for the bias, a_0^k can be set equal to one. The same notation, z_i^k , is used for the input of the activation functions.

Backpropagation

Neural networks are trained using the backpropagation algorithm, which is based on the gradient descent technique. This algorithm was developed by Paul Werbos in 1974 [213] based on the works of Arthur E. Bryson and Yu-Chi Ho [214] and Stuart Dreyfus [215]. The backpropagation algorithm gained attention years later with the work of Geoffrey Hinton, David Rumelhart, and Ronald Williams [216], where they demonstrated its effectiveness in training multilayer neural networks.

The idea behind the backpropagation algorithm is that for continuous activation functions, a small change in the weights leads to a small change in the output. To update the weights, it is thus possible to compute the partial derivatives of the loss function (L) with respect to individual weights and biases:

$$\frac{\partial L(w)}{\partial w_{i,j}^k}.$$

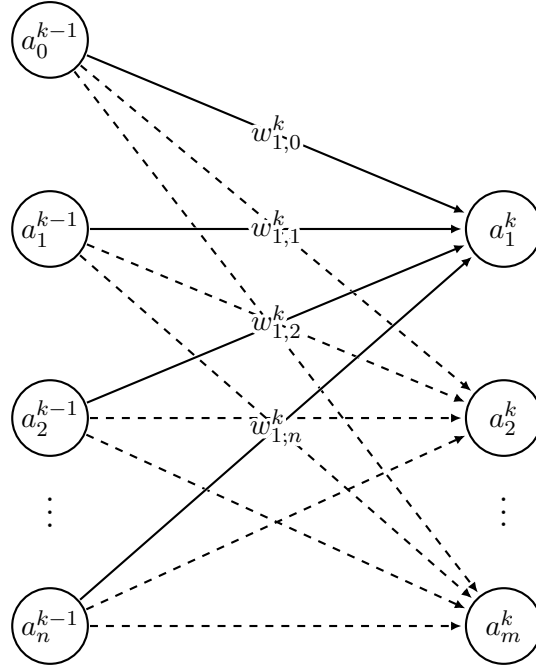


Figure 3.9: Notation to indicate the weights and activation functions in a FFNN.

It is useful to introduce the local gradient δ , representing the partial derivative of the loss function with respect to the input of the activation function (z) of the given neuron. Using the notation δ_j^k , we identify the local gradient of the j^{th} neuron in the k^{th} layer:

$$\delta_j^k = \frac{\partial L(w)}{\partial z_j^k}. \quad (3.1)$$

The backpropagation algorithm consists of the following steps:

1. Given the input vector x , the NN uses fixed weights to compute the output $y(x, w)$, storing the values of z_j^k and a_j^k .
2. The neuron errors for the output layer are computed using the equation:

$$\delta_j^{\text{out}} = \frac{\partial L_j(y_j)}{\partial y_j} \phi'(z_j^{\text{out}}),$$

where ϕ is the activation function of the last layer.

3. The neuron errors of the hidden layers are computed recursively:

$$\delta_j^k = \sum_l \delta_l^{k+1} w_{l,j}^{k+1} \phi'(z_j^k),$$

where, in this case as well, ϕ is the activation function of the neuron in question.

4. The weights are updated using standard gradient descent:

$$w_{j,i}^k \rightarrow w_{j,i}^k - \alpha \frac{\partial L_j(w)}{\partial w_{j,i}^k} = w_{j,i}^k - \alpha \delta_j^k a_i^{k-1}.$$

The first step of the algorithm is known as the forward phase, while the last three steps are known as the backward phase. A more detailed description of this algorithm, along with a derivation of these equations, can be found in Appendix J.

Backpropagation is a relatively simple algorithm; however, for neural networks with many neurons and layers, it can become computationally expensive. Various techniques can accelerate the learning process.

More advanced optimizers, such as Stochastic Gradient Descent (SGD) and adaptive learning rate methods, are described in Appendix H. Additionally, input normalization can speed up learning. It is recommended to normalize inputs to the range $[0, 1]$ using the MIN-MAX technique, where the input vectors are redefined as follows:

$$x^i \rightarrow \frac{x^i - \min_i(x^i)}{\max_i(x^i) - \min_i(x^i)}.$$

Another possible technique is Gaussian normalization:

$$x^i \rightarrow \frac{x^i - \mu_x}{\sigma_x},$$

where μ_x and σ_x are the mean and variance vectors of the inputs.

Another important aspect is the encoding of labels. For classification algorithms (when the number of classes is not too high), it is useful to use one-hot encoding. Each label is represented as a binary vector with a single non-zero element equal to one. For example, for three classes, the labels would be:

$$[1, 0, 0], \quad [0, 1, 0], \quad [0, 0, 1].$$

Proper weight initialization can help reduce redundancies, such as different neurons or groups acting in the same way. Initial weights must break the symmetry of the system. If two hidden neurons with the same activation function are connected to the same input, they should have different initial weights; otherwise, a deterministic algorithm like backpropagation would update them identically. Heuristic experience suggests that initializing weights randomly with a distribution centered at zero is advantageous. Commonly used distributions include a uniform distribution or a Gaussian distribution. Typical values for the variance of these distributions are on the order of $1/\sqrt{m}$, where m is the number of inputs to the layer.

3.2 Advanced topics

This section introduces some advanced NN models and training techniques used in ML. Several of these models have been applied in the works discussed in Chapter 4 to improve the reliability of quantum computing. Additionally, the quantum counterparts of some of these ML models are presented in Chapter 5.

Section 3.2.1 introduces the concept of deep learning, specifically multilayer NN architectures capable of solving complex tasks. In Section 3.2.2, we review the main aspects of Convolutional Neural Network (CNN), a specialized NN architecture designed for image analysis. Section 3.2.3 discusses the AutoEncoder (AE) architecture and its application to anomaly detection. In Section 3.2.4, we explore how ML can be used for generative tasks, with particular emphasis on diffusion models. Finally, Section 3.2.5 provides insights into the Reinforcement Learning (RL) training paradigm.

3.2.1 Deep learning

As stated in Section 3.1.5, the human brain’s ability to execute complex tasks relies on combining simpler features to create progressively more complex ones. This process is imitated in ML through hierarchical representation learning, or deep learning. A natural implementation of this concept is a NN with many layers, also known as Deep Neural Network (DNN), where the complexity of the features increases as the information passes from one layer to the next. The first model of a DNN was implemented in the late 1960s by Ivakhnenko et al. [217], and it was a fully connected FFNN with eight hidden layers. However, this algorithm was very slow and encountered numerous issues during training. The most famous of these is the vanishing gradient problem, which affects the training of the early layers of the NN and will be discussed in more detail in the following paragraphs. Another issue with DNNs is that increasing the number of layers also increases the model’s complexity, which raises the risk of overfitting.

It is important to understand why deep neural networks are necessary for accomplishing complex tasks. The universal approximation theorem states that a shallow FFNN can approximate any function with arbitrary precision. However, there are some classes of problems that can only be solved with an exponentially increasing number of neurons in a shallow architecture. If the neurons are distributed across multiple layers, the number of neurons increases only polynomially [218]. There are additional advantages to using DNNs. The high number of parameters seems to have a

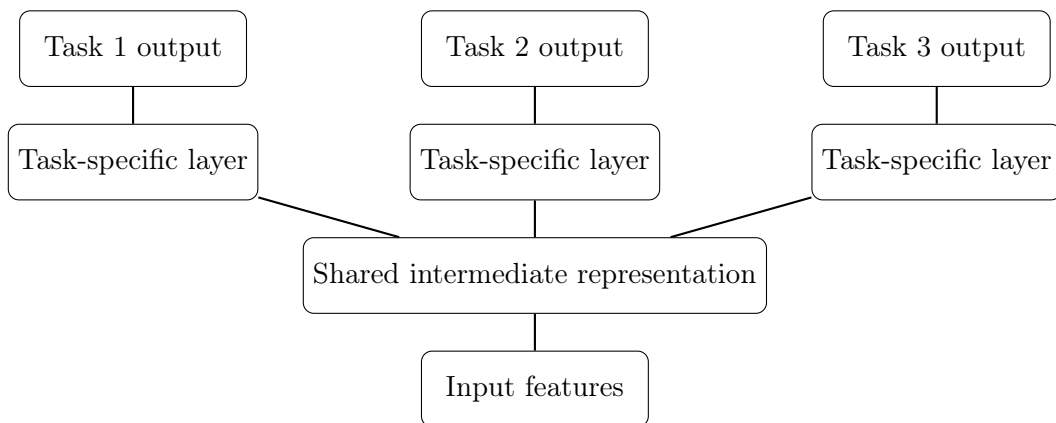


Figure 3.10: A deep neural network may use intermediate features to accomplish different tasks.

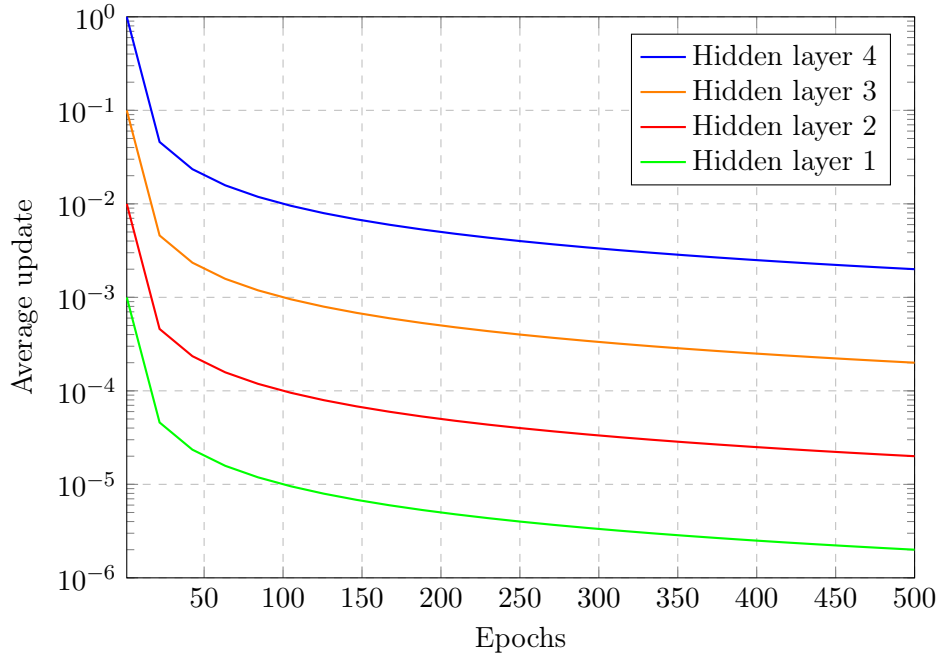


Figure 3.11: Typical behavior of the average update of the parameters in different layers of a FFNN during the training process.

smoothing effect on the loss-function landscape, making it easier to reach the absolute minimum of the function [219]. Moreover, the intermediate features, or sub-features, of a DNN can be used by the same neural network to perform different tasks, as depicted in Figure 3.10.

The vanishing gradient problem manifests as a slow update of the parameters in the first layers of a DNN. For example, Figure 3.11 shows the average update of the parameters in different layers of a FFNN during the training process. As the first layers are trained too slowly, they cannot fully utilize their expressive power to enhance the model’s complexity. Moreover, as the initial weights are assigned randomly, the first layers act as a random feature map, risking a reduction of the initial information. The vanishing gradient problem arises because the update of the parameters is determined by the backpropagation algorithm. The neuron error in a hidden layer depends on the derivatives of the activation functions of the subsequent layers. If these derivatives are smaller than one, the neuron error diminishes exponentially during backpropagation. Since the parameter updates directly depend on the neuron error, this leads to a very slow training process. Consider, for example, a toy model of a neural network with n hidden layers, each containing a single neuron with a sigmoid activation function, as depicted in Figure 3.12. The update of the first parameter (w_1) depends on the derivative of the cost function with respect to this parameter:

$$\frac{\partial L}{\partial w_1} = \sigma'(z_1) \left[\prod_{i=2}^n w_i \sigma'(z_i) \right] \frac{\partial L}{\partial y}.$$

The derivative of the sigmoid function always remains below the value of $1/4$ (Figure 3.12). Consequently, the parameter updates scale by this factor in each layer, with the average update of the first layer being on the order of $(1/4)^n$.

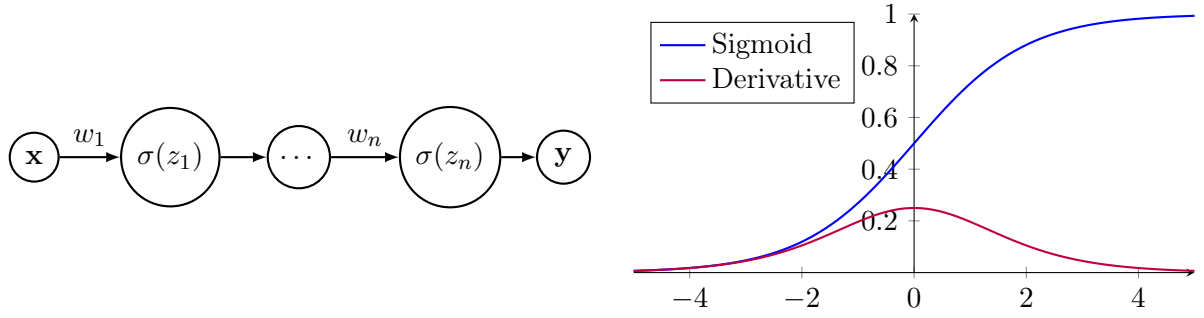


Figure 3.12: Toy model of a FFNN with n hidden layers with just one neuron (Left). Sigmoid function and its derivative (Right).

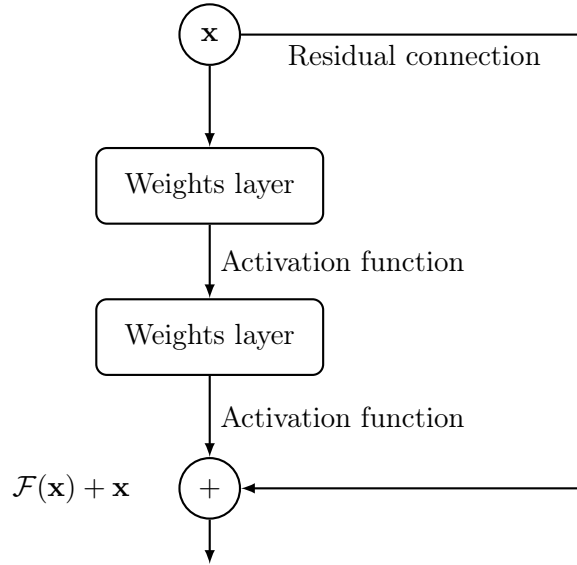


Figure 3.13: Example of a residual connection in a neural network. In this example the initial information skips two layers and is summed element-wise with the processed information.

An erroneous solution to the vanishing gradient problem might involve using an activation function with a high derivative or starting with large initial weight values. In this case, the gradient would grow exponentially as it propagates backward through the layers, leading to an exploding gradient problem. When there are many layers, the situation becomes intrinsically unstable. The only way for all layers to learn at the same speed is if the gradient products approximately balance out. However, this balancing is highly unlikely to occur by chance. As a result, when using standard gradient-based learning techniques, different layers in the network tend to learn at different rates. To mitigate unstable gradient problems and efficiently train DNNs, many solutions have been developed.

- The use of activation functions that don't saturate and produce larger derivatives is highly recommended in deep neural networks. Today, sigmoid activation functions are typically used only in the output layer for classification problems. The most commonly used activation functions are ReLU-like functions (Appendix G).
- To prevent activation functions from reaching regions with small derivatives, batch normalization layers can be introduced. These layers normalize the inputs, usually with Gaussian normalization centered at zero.

- In some deep NN architectures, residual or skip connections are employed. These connections bypass the activation functions and propagate information directly to subsequent layers, as illustrated in Figure 3.13.
- To accelerate the learning process, pre-training techniques can be used. For example, in greedy layer-wise pre-training [220], layers are trained one at a time in an unsupervised manner. This unsupervised pre-training initializes the system in a stable configuration close to a local minimum, facilitating the supervised training phase.

By employing these precautions, it is now possible to train large DNN architectures with hundreds of layers.

3.2.2 Convolutional neural networks

Convolutional neural networks have been developed for image recognition tasks [221, 222]. Digital images are essentially matrices. For example, an image of $n \times m$ pixels in 8-bit grayscale is represented as an $n \times m$ matrix, where each entry is an integer in the interval $[0, 255]$. For color images, three matrices are used to store the different intensities of red, green, and blue for each pixel. CNNs are not limited to image processing. Since these architectures exploit spatial symmetries and correlations between nearby features, they can be applied to other tasks where the inputs are matrices with specific spatial relationships and correlations.

CNNs are characterized by the use of convolutional layers that are not fully connected and have weight-sharing constraints. These properties reduce the total number of weights, speeding up training and reducing overfitting. We will now analyze the specific characteristics of convolutional layers in more detail.

Local receptive field

In a CNN, the input neurons (one for each pixel of the image) are not fully connected to the neurons in the first hidden layer. Connections exist only for a small spatial region of the image, called the local receptive field, as illustrated in Figure 3.14. A convolution operation, which we will discuss later, is applied over the pixels within this local receptive field. Typical sizes for the local receptive field are 3×3 and 5×5 input neurons. The local receptive field is moved across the entire image, and at each position, the result of the convolution operation is associated with a neuron in the first hidden layer. The shift of the local receptive field at each step is called the stride. A stride of one means shifting the local receptive field by one pixel at each step, covering every possible configuration. A stride of two means skipping one possible location of the receptive field at each step, reducing the number of neurons in the next layer by roughly half. Similarly, a stride of three reduces the output size by about a third, and so on. Pixels on the edges of the image are never at the center of the receptive field, which reduces the output dimension. To match the output size with the input size, a padding operation can be applied. Padding adds extra pixels (usually with a value of 0) to the image edges, allowing edge pixels to occasionally be at the center of the receptive field, which extends into these extra pixels beyond the edges. For an input of a convolutional layer with dimension $W \times W$, using a stride of S , padding of P pixels, and a local receptive field of $F \times F$

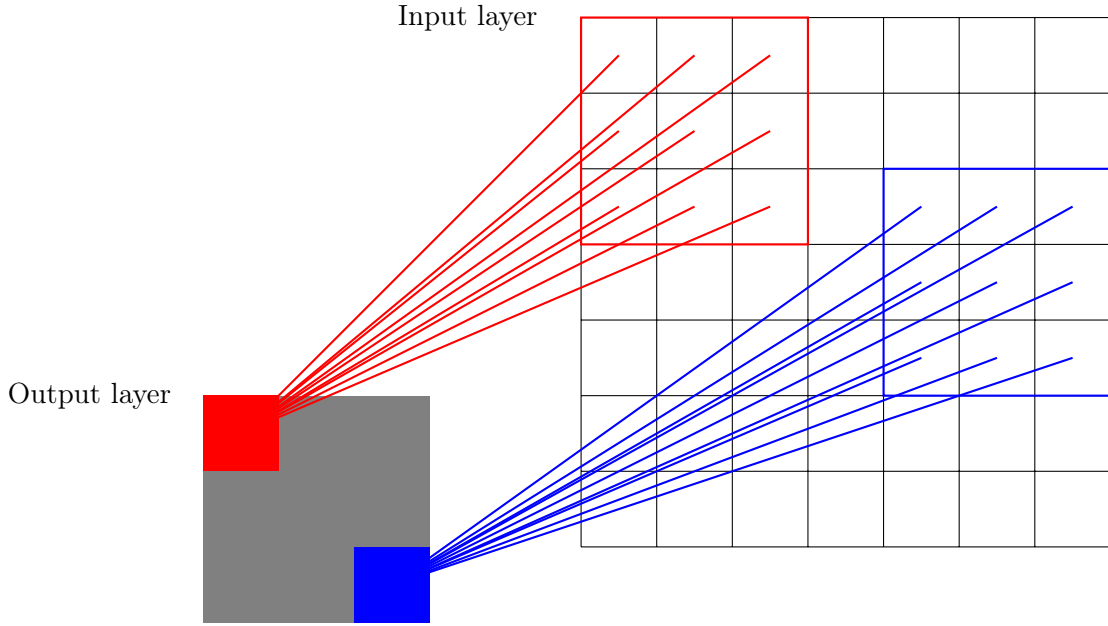


Figure 3.14: Input neurons in the local receptive field are connected to a neuron of the first hidden layer. In this example the local receptive field have dimension 3×3 .

pixels, the output will have dimension $N \times N$, where:

$$N = \frac{W + 2P - F}{S} + 1.$$

After the first layer, these operations are repeated for all the convolutional layers.

Kernels

The objects represented in an image are invariant under translation. This property can be exploited to reduce the number of weights by introducing shared weights, known as kernels. All neurons in a given hidden layer share the same weights, allowing them to identify the same feature of the image at different points. As a CNN needs to identify many elementary features, there will be numerous kernels associated with a hidden layer. Typical numbers are about 32 or 64 kernels for each layer. The number of kernels is an important hyperparameter that affects the total number of weights and has to be tuned based on experience and tests.

Kernels are used to perform convolution operations, as illustrated in Figure 3.15. During this operation, the values in the local receptive field are multiplied element-wise with the values of the kernel. The resulting value from this operation is then passed through an activation function.

Kernels are updated during the training process. They have the important task of learning weights to extract features from a set of local inputs. Moreover, because the kernel is applied across the entire image, the features it learns must be general enough to be applicable to any part of the image. The kernels, particularly those in the first layers, tend to be very general. Consequently, their values can often be imported from another CNN that has been trained with high precision. This process, called transfer learning [223], accelerates the training process, as the first layers do not need to be retrained.

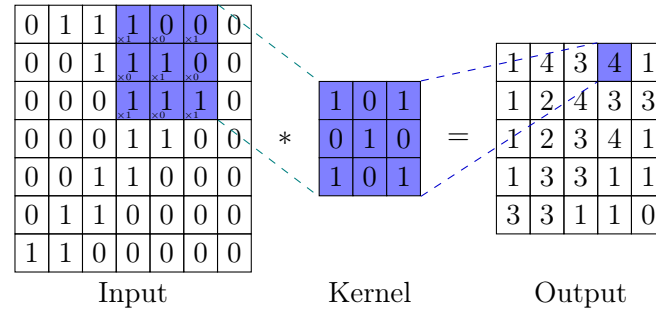


Figure 3.15: An example of convolution operation with a kernel of dimension 3×3 .

Pooling

Once a sub-feature has been identified in an image, knowing its exact position is less important than knowing its relative position with respect to other sub-features in the image. This observation can be exploited to reduce the number of weights by introducing pooling layers. The purpose of pooling layers is to simplify the information by extracting only the main features from a small region while making the neural network less sensitive to small translations of the image.

The most commonly used pooling operations are the max-pooling, which extracts the maximum value from a specific region of neurons, and average-pooling, which averages the output values in a small region.

A convolutional neural network (CNN) consists of multiple layers of neurons that perform different operations. The initial layers typically alternate between convolutional and pooling layers. The first convolutional layers aim to find a good representation of features. Once the main features have been extracted by these layers, the information is passed to dense layers that implement the final mapping. To achieve this, a flatten operation is necessary to convert the output of the convolutional layers into a vector format. An example of the architecture of a CNN is shown in Figure 3.16.

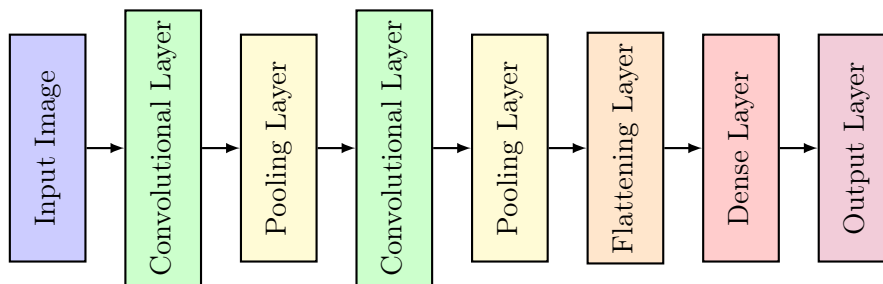


Figure 3.16: An example of an architecture of a CNN for image classification.

3.2.3 Autoencoder and anomaly detection

Anomaly detection describes a class of algorithms that aims at the identification of rare items, events or observations, which deviate significantly from the majority of the data and do not conform to a well-defined notion of normal behavior. Anomaly detection has recently gained increasing interest in many fields, from banking security to identify fraudulent transactions to high energy physics to search for unpredicted events. This technique requires only a dataset of standard data to train a

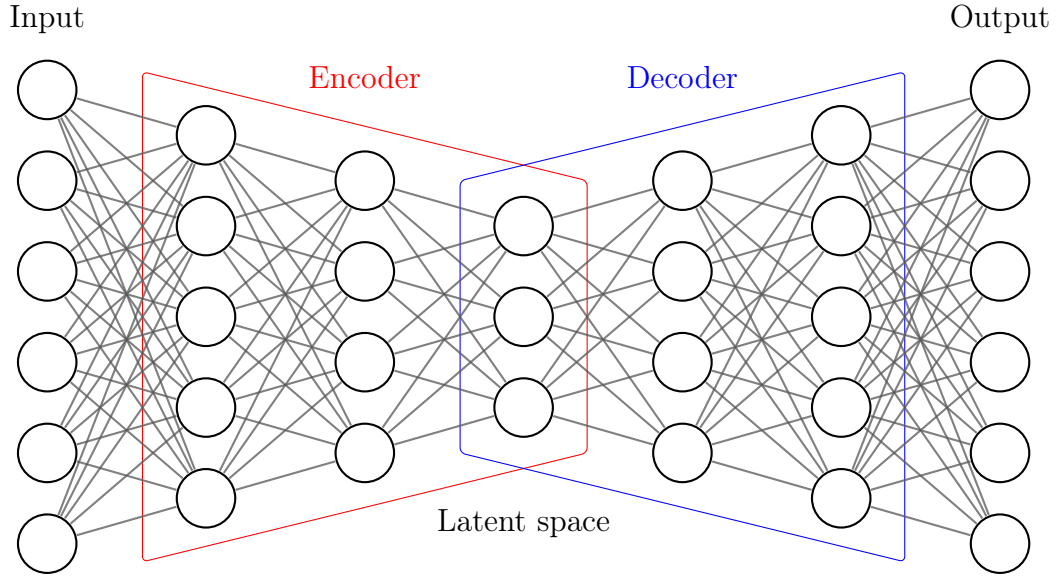


Figure 3.17: Schematic representation of an autoencoder architecture. The input data is compressed by the encoder, to a smaller number of features in the latent space. The decoder tries to reconstruct the original data from the compressed one.

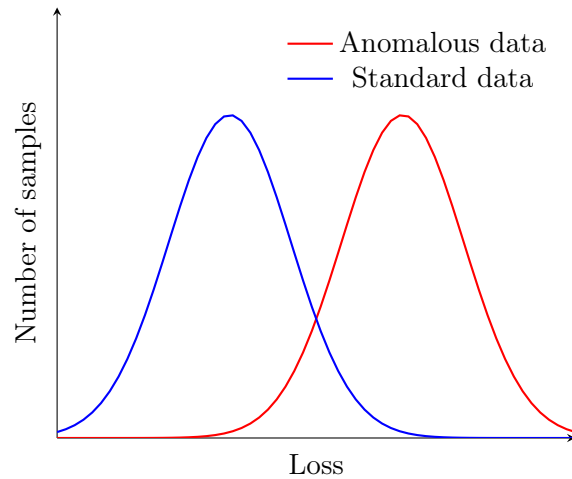


Figure 3.18: Typical behavior of the loss function distributions for standard and anomalous data in an anomaly detection problem. Anomalous data are usually expected to have a higher reconstruction loss for the autoencoder.

classifier model to distinguish anomalous data without requiring a specific description.

One way to implement an anomaly detection algorithm is to train a particular NN architecture called AE [224–226], utilized in various applications of unsupervised learning. An AE is composed of two main parts: an encoder and a decoder as reported in Figure 3.17. The encoder compresses initial data down to a small dimension (called latent dimension). The decoder inverts the process to reconstruct the original data from the compressed representation. The parameters of the neural network are trained in order to minimize the difference between the initial and reconstructed data. The loss function (also known as reconstruction loss) is therefore a measure of how accurately the reconstructed data resembles the original. For anomaly detection, the autoencoder is trained only on data samples belonging to the standard event class. When the trained model is applied to new samples we expect the loss function to have different values for normal and anomalous data. A

typical behavior of the loss evaluated over standard and anomalous data is reported in Figure 3.18. By choosing a threshold value for the loss function it is possible to classify an input based on whether its reconstruction loss lands above or below this threshold. The performance of the trained algorithm is usually presented in terms of the Receiver Operating Characteristic (ROC) curve, which shows the true positive rate as a function of the false positive rate at different classification thresholds [227], and in terms of the Area Under (ROC) Curve (AUC), which provides an aggregate measure of performance across all possible classification thresholds.

A quantum version of a classic AE will be proposed in Section 5.2 to perform anomaly detection directly on quantum data.

3.2.4 Generative models

Generative models are capable of generating samples with the same characteristics as a target dataset. The applications of this task are very wide, ranging from image, text, and video generation to data augmentation with synthetic data. It is worth noting the application of these models for the generation of new molecules or materials with specific characteristics. For example, AlphaFold [228] is a generative model trained to predict protein structures that can be used for drug discovery and medical science.

The idea behind a generative model is learning the probability distribution of a target dataset and then sampling from it to generate new data. The knowledge of this probability distribution can also be used to perform anomaly detection tasks.

The first proposed generative models are the restricted Boltzmann machines, dating back to 1986 [229]. However, it is only in the last decade that high computational power and new architectures have allowed the full exploitation of the generative power of artificial neural networks. In this section, we will briefly review some famous generative architectures, with particular regard to diffusion models, for which a quantum counterpart will be proposed in Section 5.3.

Variational autoencoders

A Variational AutoEncoder (VAE) is a modification of a standard Autoencoder (described in Section 3.2.3) proposed by Kingma and Welling in 2013 [230] that allows the generation of new data by sampling the latent space. The encoder is trained to map the input x to the latent distribution $q_{\theta}(z|x)$. This distribution is forced to resemble a known prior probability distribution $p(z)$, which is usually chosen as Gaussian: $z \sim \mathcal{N}(\mu, \sigma^2)$. This is achieved by adding the Kullback–Leibler Divergence (KLD), which measures the similarity between the two distributions, to the standard reconstruction loss.

$$\mathcal{L} = \mathcal{L}_{\text{Reconstruction}} + \text{KLD}(q_{\theta}(z|x)||p(z))$$

Once trained, it is possible to sample from $p(z)$ and use the decoder to generate new data. It is important to note that, to train the model, it is necessary to express z as: $z = \mu + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. This step is known as the reparameterization trick and ensures differentiability.

VAEs offer the advantage of being more stable during training compared to other generative models. Moreover, they provide a smooth and efficient latent representation that enables uncertainty quantification for anomaly detection. On the other hand, VAEs are known to generate blurry

outputs, and the Gaussian latent space can restrict the model’s ability to capture complex data distributions, potentially missing rare modes in the data.

Many variations of the VAE architecture have been proposed. Among them, it is worth noting the beta-VAE, which introduces a hyperparameter β to weight the KL divergence term and encourage disentangled representations in the latent space [231]. Other interesting ideas include the hierarchical VAE, which utilizes multiple layers of latent variables to increase the expressive power of complex data distributions [232], and graph-based VAEs, which can handle graph-structured data [233].

Generative adversarial networks

The Generative Adversarial Network (GAN) architecture was proposed by Ian Goodfellow et al. in 2014. This model is based on the “counterfeiter and police” scenario, where the former tries to trick the latter by creating fake banknotes that resemble the original ones. This adversarial game is mimicked using two neural networks: the generator and the discriminator. The generator (G) uses random noise z (usually Gaussian noise) to output synthetic data $G(z)$ that mimics the real data distribution. The discriminator (D) takes as input both real data x and synthetic data $G(z)$ and tries to classify them. The discriminator minimizes the classification error (binary cross-entropy loss is usually employed, see Appendix F), while the generator maximizes the uncertainty of the discriminator. This results in a min-max game represented by the objective function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim P_z} [\log(1 - D(G(z)))] .$$

The generator-discriminator system, in principle, should reach a Nash equilibrium when the generator creates synthetic data that perfectly matches the original, while the discriminator can only classify at random with a 1/2 probability [234].

Once trained, only the generator is required to create synthetic data.

GANs are capable of generating high-quality samples, addressing the problem of blurry outputs typical of VAEs. The main challenges affecting this architecture are training instabilities due to the simultaneous training of two NNs and high sensitivity to hyperparameters. Moreover, the generator may exhibit “cheating” behavior by learning to reproduce only a subset of the dataset, thereby reducing the variability of generated samples. It is also worth noting that this model is difficult to interpret, as there is almost no control over the latent space. To address this issue, the InfoGAN architecture maximizes the mutual information between a small subset of the latent variables and the observations, thus enhancing interpretability [235].

Another interesting variation of this architecture is the cycleGAN [236], which maps between two datasets to change, for example, the style of images (i.e., photos to paintings). Finally, in progressive GANs, used for image generation, the resolution of the generator and discriminator is increased over time to speed up and stabilize the training process [237].

Diffusion models

The diffusion model was proposed by Ho et al. in 2020 [238] and is a state-of-the-art generative model that offers better training stability with respect to GANs and better sample quality with respect to VAEs. The main idea of this method is to consider a diffusion process, represented by a

Markov chain, which maps an arbitrary distribution $q(\mathbf{x}_0)$ to a treatable distribution $\pi(\mathbf{x}_T)$, e.g., a Gaussian multivariate distribution. This is achieved through a Markov kernel $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, with $t \in \{1, \dots, T\}$. A parametric model is then trained to reproduce the inverse Markov chain (reverse trajectory), estimating the inverse transition probability $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Knowing $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, the target distribution $p_\theta(\mathbf{x}_0)$ can be computed as:

$$p_\theta(\mathbf{x}_0) = \int d\mathbf{x}_{1:T} \pi(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t).$$

A common choice for the forward conditional probability is the Gaussian kernel:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}),$$

where $\beta_1, \dots, \beta_T$ is a variance schedule. This choice allows to sample $\mathbf{x}_t$ at an arbitrary time step t in closed form:

$$\begin{cases} \mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \\ \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \end{cases}$$

with $\bar{\alpha}_t \equiv \prod_{s=1}^t (1 - \beta_s)$. When β_t is sufficiently small, the inverse conditional probability has the same functional form [239] of the forward kernel, hence:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}),$$

where the mean can be further parameterized as:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right),$$

with $\alpha_t \equiv 1 - \beta_t$. In this formula $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is a function approximator, implemented by a neural network, intended to predict ϵ from $\mathbf{x}_t$, with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This is equivalent to minimizing the variational bound on negative log-likelihood:

$$\begin{aligned} \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] &\leq \\ &\leq \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \equiv L. \end{aligned}$$

After training the model, it is possible to sample from $p_\theta(\mathbf{x}_0)$ by extracting a data point from the analytically treatable distribution $\mathbf{x}_T \sim \pi(\mathbf{x}_T)$ and applying the inverse Markov chain.

The described model is known as a denoising diffusion probabilistic model. It is possible to obtain a deterministic variant called the denoising diffusion implicit model [240], which allows for a reduction in the number of denoising steps. Speeding up the generative process is important for practical applications; for this reason, techniques like fast diffusion are being studied [241]. It is also possible to accelerate both the training and generative processes by performing the diffusion process in the latent space of an autoencoder, in the so-called implicit diffusion model architecture.

3.2.5 Reinforcement learning

RL is a powerful ML paradigm that trains an agent to make optimal decisions in a dynamic environment (Figure 3.19). From a mathematical perspective, a RL algorithm is described as a Markov Decision Process (MDP) [242]. An MDP is characterized by a tuple (S, A, P, R, γ) , where S represents the set of possible environmental states, A denotes the set of actions, $P(s'|s, a)$ is the transition probability of reaching state s' from state s by taking action a , $R(s, a)$ provides the immediate reward of taking action a in state s , and γ is the discount factor that balances the importance of immediate rewards against future rewards. In an MDP, the future state depends solely on the current state and action, regardless of the previous history. Solving MDPs involves finding an optimal policy that maximizes the expected averaged sum of rewards. This policy can be deterministic or stochastic, and can be represented by a function $\pi(s)$ that returns the action to be taken in state s . In a RL model, the policy $\pi(s)$ is implemented by a NN, which is trained by performing different episodes of agent-environment interaction. At the end of each episode (or batch of episodes), the average episode reward is calculated and used to update the weights of the NN using the backpropagation algorithm (Section 3.1.5).

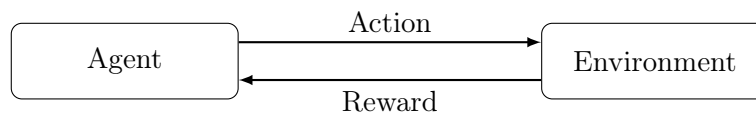


Figure 3.19: In a RL algorithm an agent is trained to perform optimal actions in an environment to maximize the reward.

In order to improve convergence to an optimal policy during the training process, different paradigms of RL have been proposed. These techniques offer strengths and disadvantages that depend on the specific problem. We will attempt to divide the possible algorithms into classes as depicted in Figure 3.20. The first main distinction is between model-free and model-based reinforcement learning. If we have full access to the dynamics of the environment, we are dealing with model-based RL. This is the case, for example, of an algorithm that learns how to control a robot where the results of the actions on the environment are well known. On the other hand, in model-free RL, the dynamics of the environment are unknown. This is the case, for example, of an algorithm learning to play chess where the policy of the opponent is unknown.

The second main distinction concerns value-based and policy-based algorithms. In value-based RL, the agent learns the value associated with a state s or a state-action tuple (s, a) . This is the case for the famous Q-learning strategy [243] and SARSA [244]. In Q-learning, the state-action values are updated based on the best action that could be taken in the next state. Since the agent gathers information about every possible policy, and not only the one it is following, this technique is defined as off-policy learning. In the SARSA algorithm, the updates are performed based on the current policy that the agent is following. Thus, the agent gathers information only on the current policy. Algorithms with this characteristic are known as on-policy.

In policy-based algorithms, the agents directly learn the mapping between the current state and the optimal action. Policy gradient methods [245], where the parameters of the policy are directly updated to maximize the reward, are the most commonly used in deep reinforcement learning today. The actor-critic paradigm [246] is a hybrid of value-based and policy-based algorithms, where the

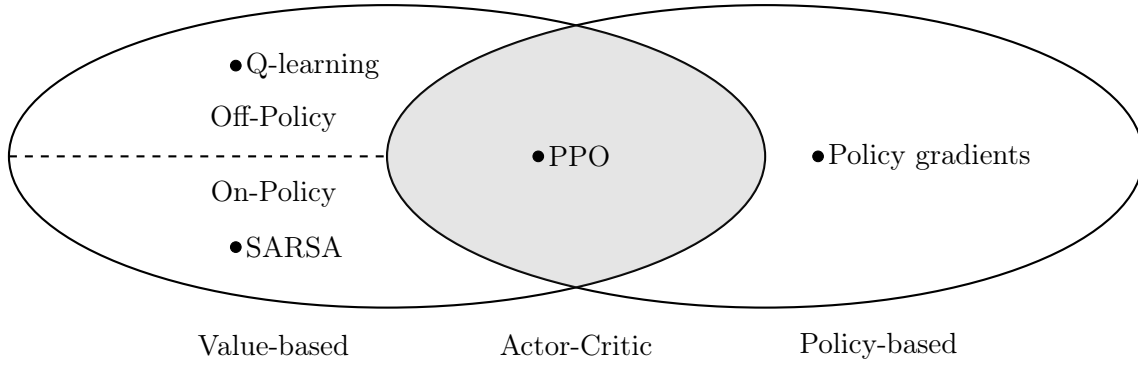


Figure 3.20: Schematic representation of the main reinforcement learning paradigms and some important algorithms.

agent is split into an actor that directly tries to find the best action and a critic that tries to learn the value of the current state-action pair.

Among the actor-critic methods, it is worth noting the Proximal Policy Optimization (PPO) algorithm [247]. PPO has been used in the work proposed in Section 4.2 for the good stability properties and because it can handle a continuous action space, which is necessary to model quantum noise. The idea behind PPO lies in restricting the policy updates to avoid too large changes in the policy, which may lead to instability during training. Namely, too large updates may lead the agent to regions with low expected cumulative reward, that may take a very long time to escape from. This is implemented in practice by clipping the loss function to discourage big policy updates:

$$\mathcal{L}_t(\theta) = \hat{E}_t[\min(r_t(\theta) \cdot \hat{A}_t, \text{clip}(r_t(\theta), \epsilon) \cdot \hat{A}_t)] .$$

where the probability ratio $r_t(\theta)$ is defined as:

$$r_t(\theta) = \frac{\pi_\theta^t(a_t|s_t)}{\pi_\theta^{t-1}(a_t|s_t)} .$$

This parameter evaluates the change in the policy going from step $t - 1$ to t , and is limited in the final loss through the clip function:

$$\text{clip}(r_t(\theta), \epsilon) = \begin{cases} 1 + \epsilon & \text{if } r_t(\theta) > 1 + \epsilon , \\ 1 - \epsilon & \text{if } r_t(\theta) < 1 - \epsilon , \\ r_t(\theta) & \text{elsewhere} . \end{cases}$$

In this way big changes in the policy will still have a limited impact on the loss. This clipped ratio is then multiplied by the advantage factor $\hat{A}_t$, as in the non-clipped case, which measures how good is on average the action taken at step t over all the other possible actions.

Finally, the minimum between the clipped and non-clipped reward is taken to provide pessimistic predictions and, thus, discourage larger jumps in the policy. PPO often outperforms algorithms like Q-learning, in environments with continuous action spaces or partial observability, where Q-learning struggles due to the need for discretization.

Even if RL is a powerful learning paradigm, it suffers some problems that generally make it more difficult to use with respect to classical supervised and unsupervised learning. Below are reported the most common problems encountered when using reinforcement learning, along with some possible solutions to mitigate them. As these problems are strictly correlated, some of the reported techniques have positive effects on addressing multiple issues.

Sampling efficiency: A RL agent learns to solve a task by interacting with the environment, where every interaction collected is considered a training sample. The number of samples needed to learn a task grows drastically with the complexity of the task. For example, state-of-the-art algorithms trained to learn chess require training for tens of millions of plays to achieve optimal results [248]. This number is incredibly high compared, for example, to humans, who are able to master the same task with thousands of iterations. Training RL agents can be time-consuming as this technique relies on brute force computation. The sampling inefficiency is mainly explained by the fact that an agent starts the training without any prior knowledge. Interesting studies are trying to find good ways to use knowledge learned from previous tasks to “jump-start” the algorithm, thus requiring less sampling to reach proper training [249].

Exploration vs exploitation: This famous problem regards finding a good trade-off between performing actions with known and unknown outcomes. If the algorithm relies too much on exploiting actions that are known to have a positive outcome, it may get stuck in a local optimum, as it doesn’t explore other strategies that may be more optimal. On the other hand, it is also important for the agent to use its previous knowledge to further optimize positive strategies. The trade-off between exploration and exploitation is usually controlled by hyperparameters. New interesting techniques, known as curiosity-driven exploration, push the agent to explore new actions in a self-supervised way to acquire knowledge that may become useful in future scenarios [250]. This allows the agent to reduce the number of samples required for training as well as increasing its generalization capabilities.

Sparse reward: This problem emerges for complex tasks where the reward signal is generated only when the task is accomplished. This makes the training process extremely slow, especially at the beginning when the agent struggles to achieve the task by performing random actions. This problem is strictly correlated with sampling efficiency. A possible solution relies on manually introducing intermediate rewards that help the agent in accomplishing the task. This process, known as reward shaping, must be used with extreme caution as it may lead to unwanted “cheating” behavior of the agent [251, 252]. The previously introduced curiosity-driven exploration helps mitigate the problem of sparse rewards. Another solution that can be applied for tasks where an almost optimal strategy is known is imitation learning [253], where the agent is helped to complete a task by observing successful behaviors. Finally, the experience replay technique [254], where previously sampled sequences of actions are replayed to improve the policy, has positive effects both on the sparse reward and sampling efficiency problems.

Chapter 4

Machine Learning for Quantum Computing

In this chapter, two novel applications of ML techniques to quantum computing are presented.

Section 4.1 reports on new results in the field of quantum error correction, where different kinds of NNs were used for the decoding of surface codes. The application of ML techniques to quantum error correction has been explored in previous works, as discussed in Section 4.1.4. The novelty of this work lies in its focus on the use of CNNs and the application of explainable machine learning techniques to better understand the behavior of the algorithm and enhance its performance. This research was published in 2023 [15] and follows the study carried out during the master thesis.

Section 4.2 presents a novel approach to quantum noise modeling using RL. Several algorithms for characterizing and modeling quantum noise have been proposed in the literature, as discussed in Section 4.2.3. The novelty of this work lies in the use of RL to reduce heuristic assumptions about the noise model, thereby increasing the flexibility of the algorithm. Preliminary results were published in the conference proceedings of AIQxQIA 2023 [22], while a complete paper is currently under review with the preprint available on arXiv [23].

This research has been presented at several conferences and workshops, including AIQxQIA 2024 in Bolzano, QC-INFN 2024 in Padova, and ICSC-Spoke10 2024 in Milan.

4.1 Convolutional neural network based decoders for surface codes

The background of this research can be found in Section 2.3, in particular in the Subsections 2.3.4 and 2.3.5 where the principles of quantum error correction are explained. Moreover, Section 3.2.2 provides a concise introduction to CNNs.

4.1.1 Abstract

The decoding of error syndromes of surface codes with classical algorithms may slow down quantum computation. To overcome this problem it is possible to implement decoding algorithms based on artificial neural networks. This work reports a study of decoders based on convolutional neural networks, tested on different code distances and noise models. The results show that decoders based on convolutional neural networks have good performance and can adapt to different noise models. Moreover, explainable machine learning techniques have been applied to the neural network of the decoder to better understand the behaviour and errors of the algorithm, in order to produce a more robust and performing algorithm.

4.1.2 Introduction

In recent years a lot of interest has grown around the possibility of constructing efficient quantum computers. One of the main problems regards the protection of quantum information from external noise. This process, known as decoherence, is due to the inevitable interaction between the qubits and the environment [255, 256].

A solution may come from QEC codes, where logical quantum information is stored in the degrees of freedom of a system composed of many physical qubits. In this way, the logical information can be restored even in the presence of single physical qubit errors [257, 258].

Surface codes are a family of QEC codes [259, 260], their implementation is simpler, with respect to other codes, because only near interactions between the physical qubits is needed. In a surface code a greater number of physical qubits gives better protection from errors, however, this makes the decoding procedure more complicated. The decoding algorithm is implemented by a classical computer and should be carried out fast to not slow down the quantum computation. With classical algorithms it is difficult to obtain a reasonable execution time incrementing the number of physical qubits [2]. A solution to this problem may come from the application of NNs. Simple models of neural networks already give good decoding accuracy with a constant execution time for small surface codes [261]. In order to decode codes with a larger number of qubits, more complicated neural network models have to be employed. This work focuses on decoders based on CNNs for different code distances, noise models and error probabilities. To improve the performance of the algorithms different model architectures have been studied to find the best error probabilities for the training set and to reduce the number of trainable parameters of the NN. In the last section of the article, to get a better understanding of the behaviour of neural network based decoders, explainable machine learning methods have been employed. An original technique to improve the decoder performance, based on data augmentation driven by the results of the model explainability, is also reported.

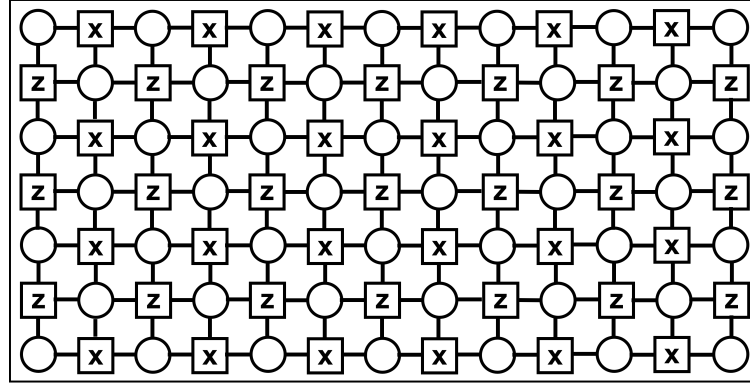


Figure 4.1: Schematic representation of a surface code. Data qubits are represented as circles while X and Z measurement qubits are represented as squares. Each qubit is connected with its nearest neighbours.

4.1.3 Surface codes

In a surface code physical qubits are arranged on a squared lattice; they are divided in two categories (Figure 4.1). The data qubits store the quantum information. The measurement qubits, also called stabilizers, are used to perform projective measurements, on the nearest neighbour data qubits. Measurement qubits are divided into Z and X measurement qubits, that are used to perform measurements of the respective Pauli operators. It is easy to observe that all the stabilizers commute, thus, after a measurement cycle the system collapses into an eigenstate of all the stabilizers [262, 263].

A surface code contains the logical information of a qubit. The logical state can be modified with logical operators. An operator of this kind has to preserve all the eigenvalues of the stabilizers [264]. On each border of a surface code there are only X measurement qubits or Z measurement qubits, these borders are called X sides or Z sides of the surface code respectively. Every chain of single-qubit X operators, that connect the X sides, works as an X logical operator. This is also true, in the case of Z logical operators, for chains of Z single-qubit operators that connect the Z sides. The minimum number of single-qubit operators, necessary to change the logical state, is equal to the number of data qubits on a side of the lattice, this important value is called the distance of the code (d) [260]. Figure 4.2 shows some examples of logical operators in a $d = 5$ surface code.

Errors

Unlike the classical bit, where only bit flip errors may appear, for a qubit there is a continuous set of possible errors. However, after a measurement cycle, the state of the qubits collapses into a discrete set of possible errors [256–258]. There are three main kind of errors that appear on a surface code. Data qubit errors are the classical bit flip (X errors), phase flip (Z errors) and a combination of both (Y errors). A single error of this type can be easily identified by observing a change of the eigenvalues in the neighbouring stabilizers (Figure 4.3). Measurement errors appear when a projective measurement fails. A single error of this type changes just the value of the considered stabilizer. Gate errors may appear during the measurement process when imperfect Hadamard and C-NOT gates are used. C-NOT gate errors are the most difficult to be identified because they affect two qubits.

It is possible to test different noise models by including only some types of the errors previously described and by changing their probability. In this work many simulations of surface codes have

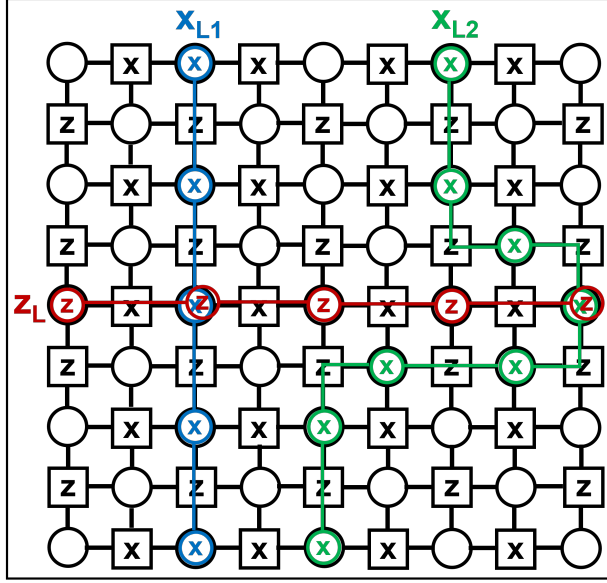


Figure 4.2: Examples of logical operators in a $d = 5$ surface code. Z and X logical operators are composed of chains of single-qubit Z or X operators connecting the Z or X sides respectively.

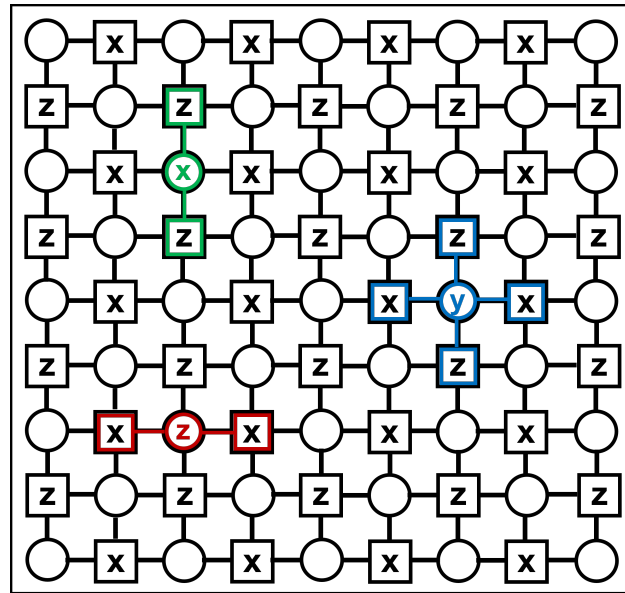


Figure 4.3: Signal error produced by single data qubit errors. The highlighted measurement qubits change their measurement value.

been performed to test different code distances, error models and probabilities. For this purpose a specific python library has been employed¹. The performance of a decoding algorithm can be very sensitive with respect to the considered error model and characteristics of the surface code [265–267]. In the depolarising error model only data qubit errors are considered. Given p the error probability, each data qubit is subject to an X , Z , or Y error with equal probability $p/3$. Perfect measurements and gates are assumed, thus only one measurement cycle is necessary. In the circuit noise model also the measurement qubits are subject to errors and gates are not assumed perfect. In this case more measurement cycles are necessary to identify the errors.

¹E. Villaseñor and B. Criger: https://github.com/evalvarez12/Distributed_Surface_Code.git, 2021.

Decoding algorithms

In a QEC code, the Hilbert space that describes the state of the physical qubits can be decomposed into a logical and an ancillary subspace [268]. After error correction the ancillary state, that increases the redundancy of information, is restored. This is not true for the logical state that may be modified, causing a logical error. The performance of a decoding algorithm can be tested by studying the rate at which logical errors appear, as a function of the single-qubit error probability. The Minimum Weight Perfect Matching (MWPM) algorithm restores the ancillary state with the minimum number of corrections [269–271]. This decoder is only nearly optimal for bit-flip noise [272] (independent X and Z errors) and it represents a standard benchmark for other decoders. When studying the logical error rate with respect to the single-qubit error rate a typical behaviour occurs. For low error probabilities, the decoding accuracy increases with the dimension of the code. On the other hand, for high error probabilities, increasing the distance of the code reduces the accuracy of the decoder [260, 266]. The cross-over of these two regimes occurs at the threshold error rate.

In order to use artificial neural networks, it is necessary to transform the original decoding problem in a classification problem. For this purpose, neural network based decoders are composed of two components. The simple decoder analyses the error syndrome and proposes a correction that matches with the syndrome. This algorithm may be implemented by a neural network but it is simpler to use naive decoder. In this work the simple decoder corrects each error with a chain of operators that connects it to the nearest border. In this way the ancillary state of the system is restored. The High Level Decoder (HLD) takes as input the error syndrome and tries to find out if the correction of the simple decoder has created a logical error. This is a classification problem that may be solved by a neural network.

4.1.4 Related works

Many different algorithms, based both on classical methods and machine learning techniques, have been tested for the decoding of surface codes. Apart from the MWPM previously introduced, examples of decoding algorithms not based on neural networks are the renormalization group decoder [273, 274], the cellular automaton [275], the maximum likelihood decoder [276] and the Markov chain Monte Carlo decoder [277]. For these algorithms it is difficult to find a good compromise between decoding accuracy and execution time [2]. As neural network based decoders show a good compromise between accuracy and execution time, many studies have been carried out to test different neural network architectures. The first studies in this sector [278] show that, for small surface codes, neural network based decoders have a decoding performance, similar to MWPM, for a depolarising noise model. When measurement error and imperfect gates are included it is possible to improve the MWPM algorithm taking into account of more complex noise models [279, 280]. However neural network based decoders remain interesting because they require constant execution time and can easily adapt to different noise models. Some problems arise for high distance surface codes as the number of possible error syndromes increases exponentially. This means that, for the correct training of the HLD, the training set have to increase with the dimension of the code to contain the most statistically relevant errors.

To scale the methodology to higher distance surface codes, some interesting approaches have been proposed. A recurrent neural network architecture has been tested by P. Baireuther et al.

for the decoding of correlated errors [3]. G. Torlai and R.G Melko have tested a decoder based on a stochastic neural network (Boltzmann machine) that is applicable to a wide variety of stabilizer codes [4]. Other interesting studies on deep learning based decoders have been carried out by S. Krastanov and L. Jiang [281]. S. Varsamopoulos et al. [261] where they have compared decoders based on feed forward neural networks and on recurrent neural networks. Recent works have tested decoders based on distributed neural networks [2], in order to reduce the size of the training set for high distance surface codes. Another interesting novel idea comes from the application of machine learning techniques to an ensemble of classical decoders [8]. Other decoders based on machine learning have been recently tested by D. Bhoumik et al. [9]. A state-of-the-art decoder, scalable to high distance surface codes, has been created by K. Meinerz et al. combining convolutional neural networks (to preprocess local information) with a conventional algorithm [5]. Other decoders based on convolutional neural networks have been tested on high distance surface codes for a depolarising or bit flip error models [6, 7]. Finally it is worth mentioning the recent AlphaQubit, a neural network decoder combining convolutional layers and attention that achieves state-of-the-art performance under realistic hardware noise up to distance 11 [282].

While this work concentrates on smaller dimension surface codes, with respect to the previously mentioned studies, some new elements have been introduced. A more realistic noise model, that includes measurement errors, has been decoded with a convolutional neural network. The performance of the HLD has been studied with respect to the choice of the training set error probability. The dilated convolution technique has been tested to scale to higher distance codes and to reduce the number of trainable parameters of the neural network. Moreover, in order to understand better the cases where the neural network fails, explainable machine learning techniques have been applied to the HLD.

4.1.5 Convolutional neural network based decoders

In this section the characteristics of high level decoders based on FFNNs and CNNs will be tested. Two different noise models will be considered, the simple depolarising noise and a more complicated noise model where measurement errors and more measurement cycles are present. Before the comparison some preliminary studies are necessary for the tuning of the hyperparameters. Furthermore, to improve the accuracy of the HLD, convolutional neural networks based on dilated convolution layers, able to better capture local features at different spatial scales, will be tested.

Depolarising noise

For the depolarising error model, datasets of 5×10^6 elements have been generated with a single qubit error probability $p = 0.1$ and for code distances $d = 7; 9; 11$. All the neural networks employed, both FFNN and CNN, share the following characteristics. ReLU activation functions have been used in all hidden layers while Softmax activation functions have been used for the output layer (see Appendix G). The loss function employed is categorical cross entropy (see Appendix F). The ADAM optimiser has been used (see Appendix H), with a mini-batch size of 32 elements. The neural networks have been trained for 20 epochs using 10% of the original dataset as evaluation set. FFNNs with different numbers of layers and neurons have been tested on each dataset. The best results, for all the code distances, have been obtained using three hidden layers with a number of

trainable weights reported in Table 4.1.

Table 4.1: This table reports the optimal number of trainable weights (P) of the feed forward neural network obtained for different code distances (d) for a depolarising noise.

d	P
7	5.67×10^5
9	6.4×10^6
11	3.9×10^6

For the convolutional neural networks, the input format consists of squared matrices with a number of rows and columns equal to $2d - 1$, where d is the distance of the code. Each matrix element represents a qubit of the surface code. The value of the data qubits is set to zero, while the value of the measurement qubits is 1 if no error has been detected or -1 if an error is detected. The first parameters of the CNN to be defined are the number of filters and the number of convolutional and dense layers. For the number of filters the best results have been obtained using 64 kernels of dimension 3×3 . The number of convolutional and dense layers that showed the best performance are reported in table 4.2. Other fine tests have been carried out to improve the performance, for example a stride with value 2 has been tested for the first convolutional layer but this reduced the overall performance. A small increase in the accuracy has been obtained with a padding on the first layer.

Table 4.2: This table reports best hyperparameters of the CNN for different code distances correcting a depolarising error model. d is the code distance, CL is the number of convolutional layers, DL is the number of dense layers, N is the number of neurons in the dense layers and P is the total number of trainable weights.

d	CL	DL	N	P
7	3	1	512	2.7×10^6
9	4	1	1024	8.0×10^6
11	6	2	1024	9.2×10^6

The accuracy obtained with the previously described models of neural networks is reported in Figure 4.4 and has been compared to the accuracy obtained with MWPM for different code distances and single-qubit error probabilities. The test on different error probabilities has been performed on datasets of 2×10^5 elements. For $d = 7$ codes the performance of the high level decoder, both based on FFNN or CNN, is greater than MWPM. For greater distance codes the HLD has a worse performance than MWPM. This is due to the fact that the dimension of the training set is still too small to include the most statistically relevant error syndromes, necessary for a correct training of the network. However, for these code distances, it is possible to observe the advantage in the use of a CNN with respect to a FFNN.

Depolarising plus measurement errors

In order to reproduce a more realistic error model it is necessary to include measurement errors and more correction cycles. In this section we will employ the following noise model: before a measurement cycle a depolarising noise is applied with single-qubit error probability $p = 0.01$. After applying the depolarising noise, an imperfect measurement cycle is performed. The measurement

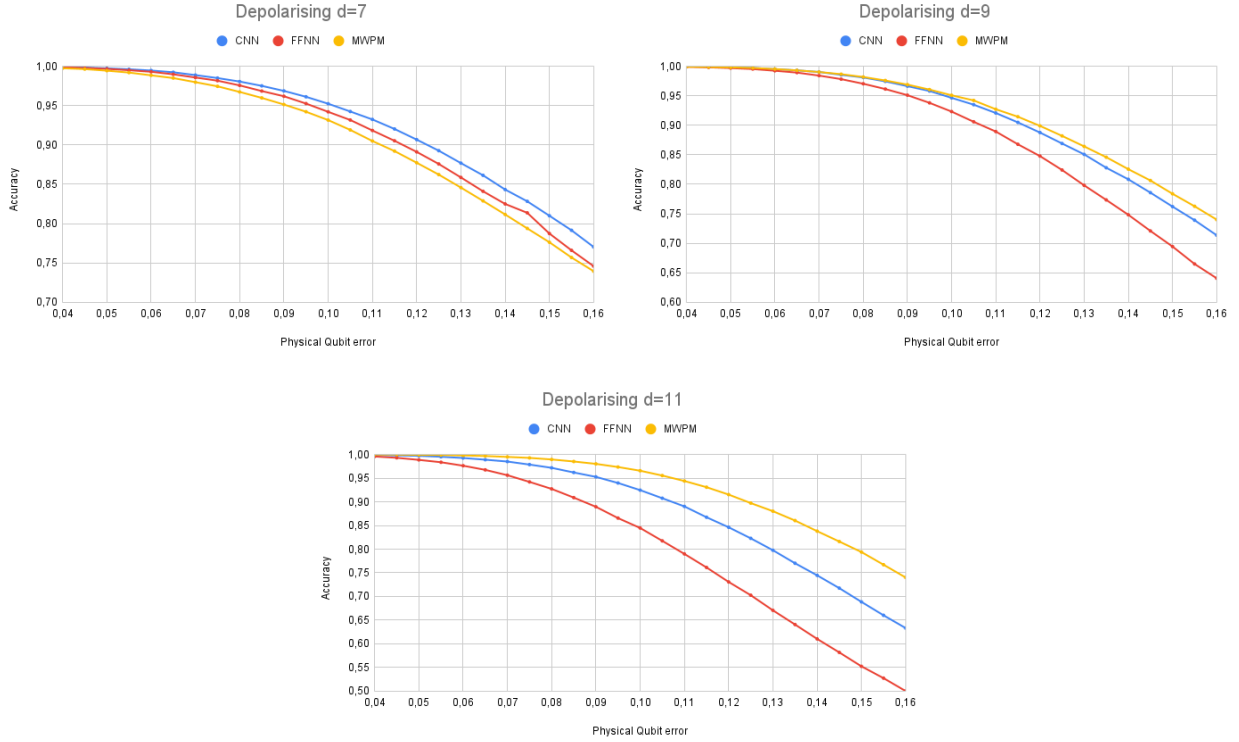


Figure 4.4: Comparison between accuracy obtained with HLD based on FFNN and CNN and the accuracy obtained with MWPM for different code distances (d) and single-qubit error probabilities (Physical Qubit error).

error probability has been set to $q = 0.01$. Three rounds of depolarising error plus imperfect measurements are carried out on each surface code. This error model is different with respect to the channel noise model where gate errors are included. However, this noise model is easier to simulate, and can be employed for an initial study of the behaviour of high level decoders based on CNN, when measurement errors and many correction cycles are present. The MWPM algorithm still works for a circuit level noise model with many measurement cycles, in this case the error matches are carried out on a three dimensional graph [265]. As benchmark, in this section, we employed the simplest version of the MWPM algorithm that considers all measurements as perfect. The performance of this algorithm is sub-optimal but can be improved by adjusting the weights in the matching graphs [283].

It is important to notice that three measurement cycles is not the standard way to benchmark noise models with measurement errors, that is carried out using as many measurement cycles as the distance of the code. However, we have decided to reduce the number of measurement cycles in order to speed up the generation of the datasets thus obtaining more training samples. In fact, the time required to generate large training sets as well as their size was at the limit of the available computational resources, in particular in Section 4.1.6 where many training sets with different error probabilities have been employed. Reducing the number of imperfect measurement cycles does not significantly change the decoding problem. As a reference, Figure 4.5 reports the performance of MWPM obtained on surface codes of dimension $d = 9$ and the previously described noise model using three measurement cycles and nine measurement cycles.

For the depolarising plus measurement errors model it is necessary to find the best hyperparam-

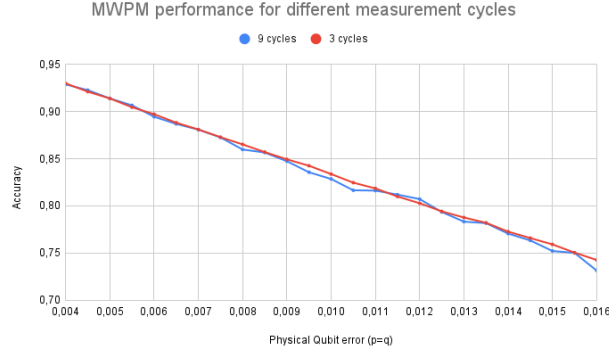


Figure 4.5: Performance of MWPM for the depolarising plus measurement errors model using a different number of imperfect measurement cycles (3 and 9). The depolarising error probability (p) has been set equal to the measurement error probability (q). For this noise model a different number of measurement cycles does not modify significantly the decoding problem.

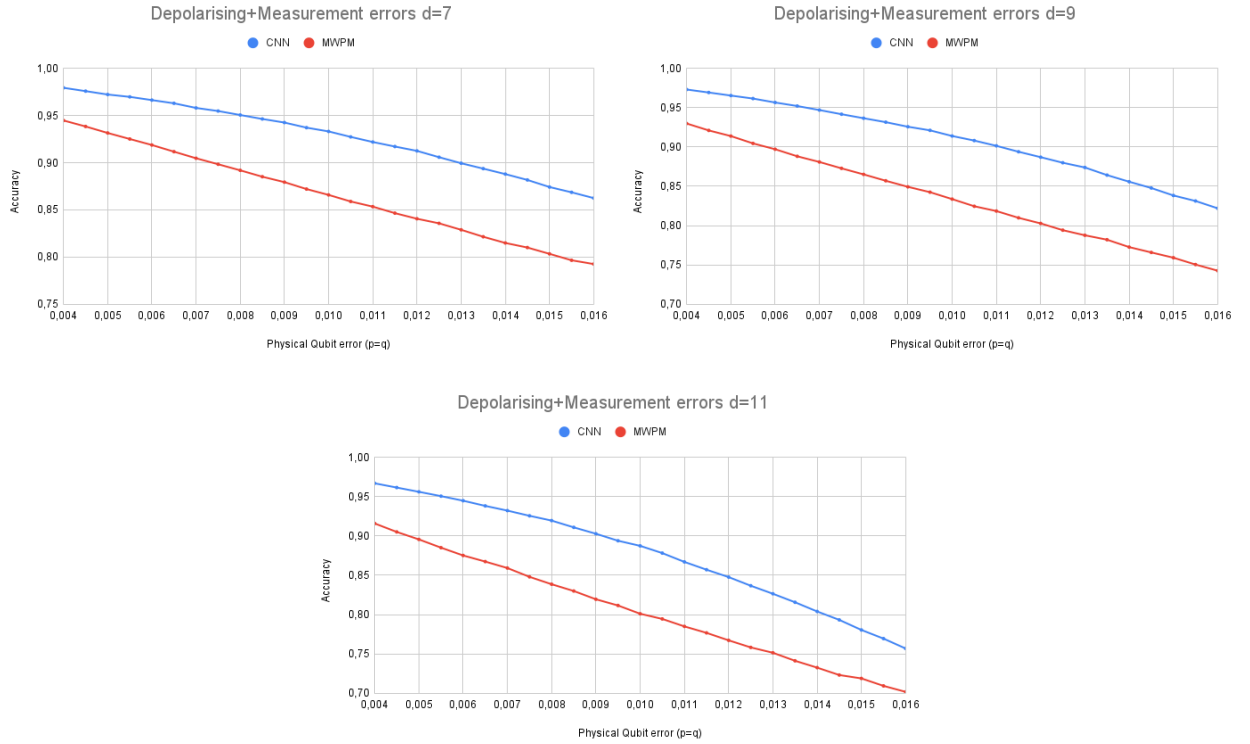


Figure 4.6: Comparison between accuracy obtained with HLD based on CNN and the accuracy obtained with the classical decoder MWPM for the depolarising plus measurement errors model, the depolarising error probability (p) has been set equal to the measurement error probability (q).

eters for the neural network employed in the HLD. All the characteristics of the neural networks, like kernel dimension and activation functions are the same employed for the depolarising noise. The hyperparameters that were changed to improve the performance regard the number of layers and neurons. The neural networks have been trained on datasets of 2×10^6 elements, using 10% of the original training set as the evaluation set.

For the CNN the best hyperparameters are reported in Table 4.3. Some tests have been carried out using a FFNN, however no interesting results have been obtained using this architecture.

The accuracy obtained with the HLD, for the depolarising plus measurement errors model, is

Table 4.3: Best hyperparameters of the CNN for different code distances for a depolarising plus measurement errors model. d is the code distance, CL is the number of convolutional layers, DL is the number of dense layers, N is the number of neurons in the dense layers and P is the total number of trainable weights.

d	CL	DL	N	P
7	3	2	1024	6.4×10^6
9	3	2	1024	12.2×10^6
11	4	2	512	7.7×10^6

reported in Figure 4.6 and has been compared to the accuracy obtained with MWPM. The test on different single-qubit error probabilities have been made on datasets of 2×10^5 elements. When measurement errors are included the accuracy of MWPM reduces sensibly while the HLD is able to better adapt to this different noise model.

Dilated convolution

In order to improve the performance of the HLD for high distance codes, it should be useful to increase the local receptive field without incrementing the number of weights of the kernels. This result can be obtained with a dilated convolution [284]. Let a standard convolutional layer use a kernel $K \in \mathbb{R}^{k \times k}$ with dilation rate $d = 1$. A dilated convolution with rate $d > 1$ applies K over the input feature map X by sampling input positions spaced by d . The output feature map Y at location (i, j) is defined as

$$Y(i, j) = (X *_d K)(i, j) = \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} K(a, b) \cdot X(i + d \cdot a, j + d \cdot b).$$

Because the kernel K still has k^2 weights, dilated convolution does not increase the number of parameters compared to a standard $k \times k$ convolution. However, its effective receptive field grows: the dilation rate stretches out the span of input positions that influence each output. In particular, the effective kernel size (in one dimension) becomes

$$k_{\text{eff}} = k + (k - 1)(d - 1),$$

so for a $k \times k$ kernel with dilation d , its receptive field behaves like a $(k_{\text{eff}} \times k_{\text{eff}})$ standard kernel. Thus one obtains larger context per layer (larger receptive field) without increasing the number of weights.

Three different implementations of a CNN with a dilated convolution have been tested. In the first case only the first layer of the CNN has a dilation factor equal to two. In the second case all the convolutional layers except the first one have dilation factor two and in the third case all the layers have this dilation factor. The training has been performed, for all code dimensions, on the same datasets described in Section 4.1.5 for the depolarising error model and in Section 4.1.5 for the depolarising plus measurement errors model. The best results have been obtained using a dilated convolution for all the convolutional layers except the first one. The performance of this CNN architecture has been compared with the same model without the dilation factor. The results are reported in Figure 4.7 for depolarising error model (top) and depolarising plus measurement

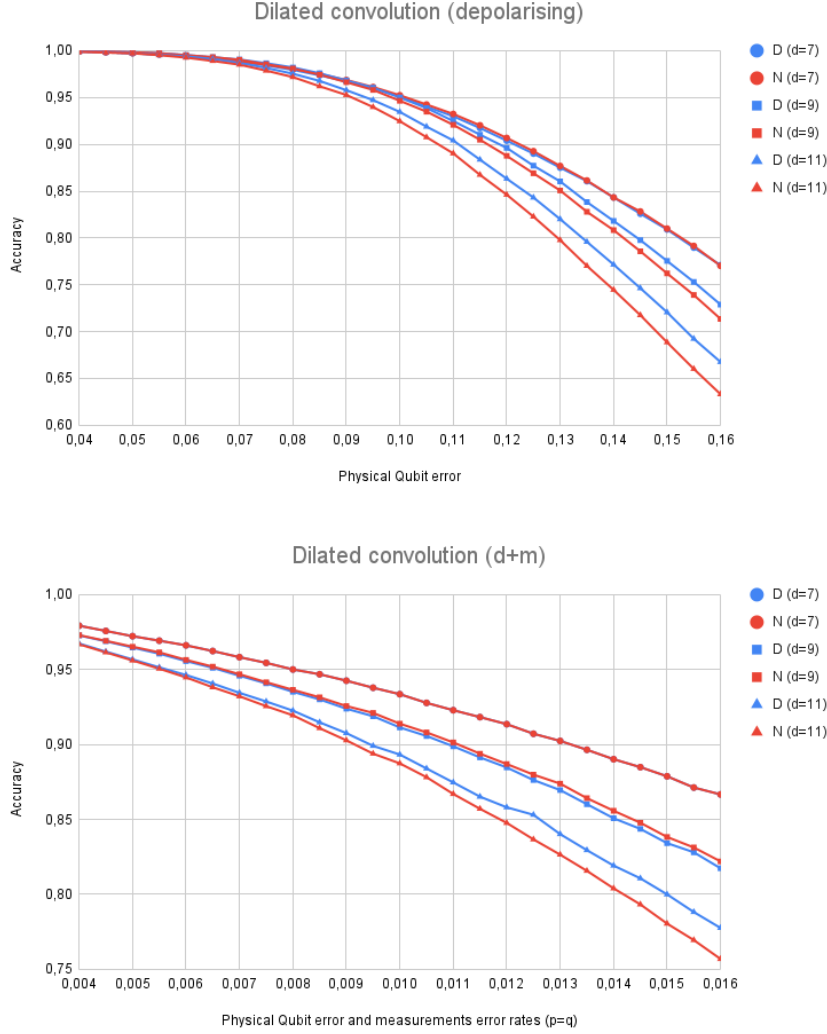


Figure 4.7: Comparison of the accuracy, on different code distances (d), obtained with a CNN with no dilation factor (N) and the same model with a dilated convolution with a dilation rate equal to two on all the convolutional layers except the first one (D). For the depolarising error model (Top) the neural networks have been trained on a dataset with $p = 0.1$. While for the depolarising plus measurement errors model (Bottom) $p = q = 0.01$ (three measurement cycles). The graphs report the accuracy of the CNN decoder obtained using the described neural network architectures for different physical qubit error rates.

errors model (bottom).

The results obtained in this study show that dilated convolution may be a good way to improve the performance of the decoder for high distance codes. In fact, while for codes of distance 7 and 9 the use of dilated convolution doesn't alter the performance sensibly, for codes of distance 11, where there are more convolutional layers, it is possible to obtain a performance improvement. The performance increase obtained with the dilated convolution is due to a reduction of the number of trainable weights in the neural network. In fact, no padding is used in the layers after the first one, so incrementing the local receptive field reduces the number of neurons of the first dense layer.

4.1.6 Choice of the training set

Neural networks are algorithms highly employed for their good generalisation properties. For the HLD it is possible to test these properties by training and testing the neural networks on datasets with different error probabilities. This is also very useful in real applications where the exact physical error rate is unknown.

For the depolarising error model, error probabilities of $p = 0.05; 0.075; 0.1; 0.13$ have been used to train the CNN for each code dimension. Each dataset is composed of 5×10^6 elements. The hyperparameters of the CNN are the ones described in Section 4.1.5 (Table 4.2). The performance has been evaluated on different error probabilities, the same datasets used in Section 4.1.5. The results of this study are reported in Figure 4.8.

Figure 4.8 shows that a better performance is obtained when the neural network is trained on a higher error probability. This is due to the fact that, in datasets with higher error probability, there are statistically relevant samples that are not present in datasets with a lower error probability. However, when the dimension of the code becomes greater, a high error probability for the training set may introduce rare but complex error syndromes that are difficult to learn from scratch and can destabilize the training process. This occurred, for instance, with codes of dimension 11 at error probability $p = 0.13$. In this case, reliable convergence was achieved only by initializing the network with weights obtained from a model previously trained at $p = 0.1$. This transfer of knowledge facilitates learning of the more complex syndromes present at higher error rates and ultimately improves the performance of the decoder.

A similar analysis has been carried out for the depolarising plus measurement errors noise model. Four datasets (2×10^6 elements each) with different error probabilities have been generated for different code dimensions. The depolarising error probability is indicated with p while the measurement error probability with q , the four datasets have error probabilities of: $p = q = 0.005$, $p = q = 0.0075$, $p = q = 0.01$ and $p = q = 0.013$.

For each code distance four neural networks have been trained using a single dataset. The CNN architecture and characteristics are the same of Section 4.1.5 (Table 4.3). The performance test with different error probabilities has been carried out on the same datasets used in Section 4.1.5. The results of this study are reported in Figure 4.9, they are similar to the results obtained for the depolarising error model: it is better to use a higher error probability to train the neural network. However, also in this case, a high error probability may cause the neural network not to train correctly. In fact for both codes of distance 9 and 11 it was possible to train the network on an error probability $p = q = 0.013$ only by starting from the weights of the neural network previously trained on a lower error probability.

4.1.7 Model explainability

In order to trust complex and less transparent algorithms like artificial neural networks, it is necessary to know why they fail or work correctly. In particular, for convolutional neural networks trained for classification, it is important to understand the inputs that influence more the final decision. Many methods have been developed to construct saliency maps of the input pixels for image classification [285]. For example, with the GradCAM and Occlusion algorithms [286, 287], it

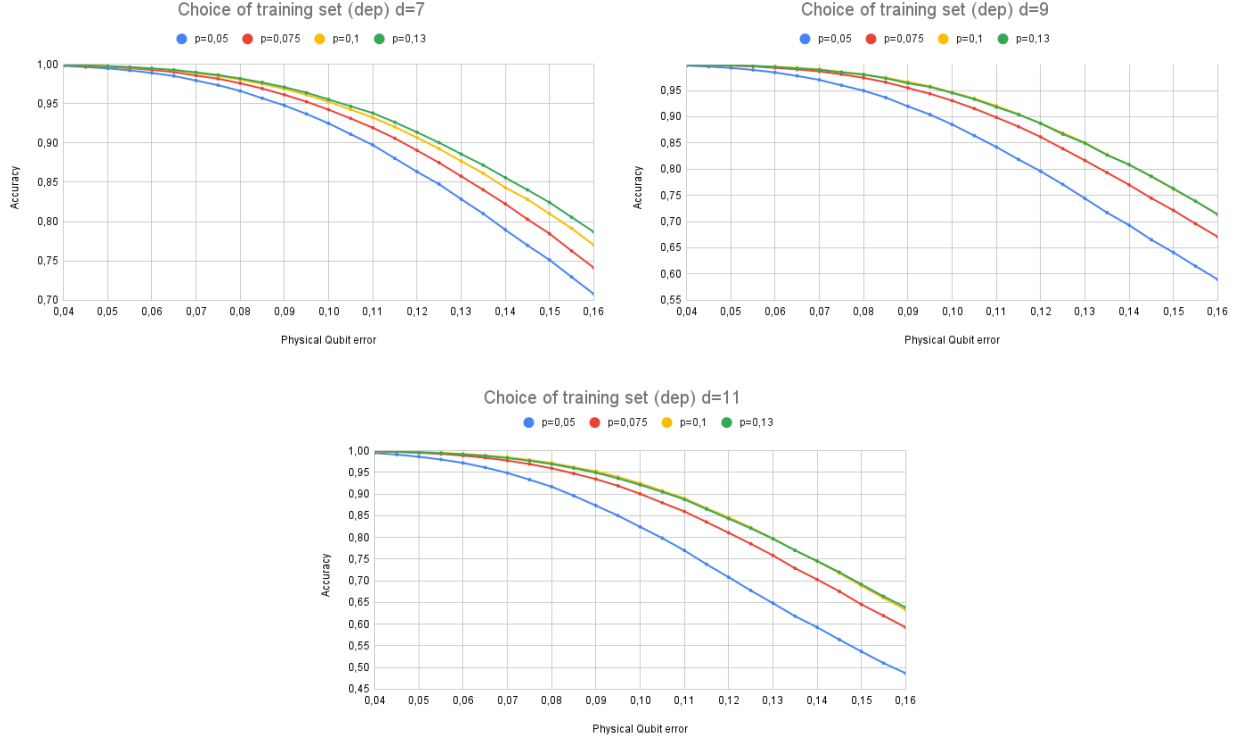


Figure 4.8: Accuracy of the HLD trained and tested on different single-qubit error probabilities for different code distances d . The training set single-qubit error probability (p) is reported in the legend.

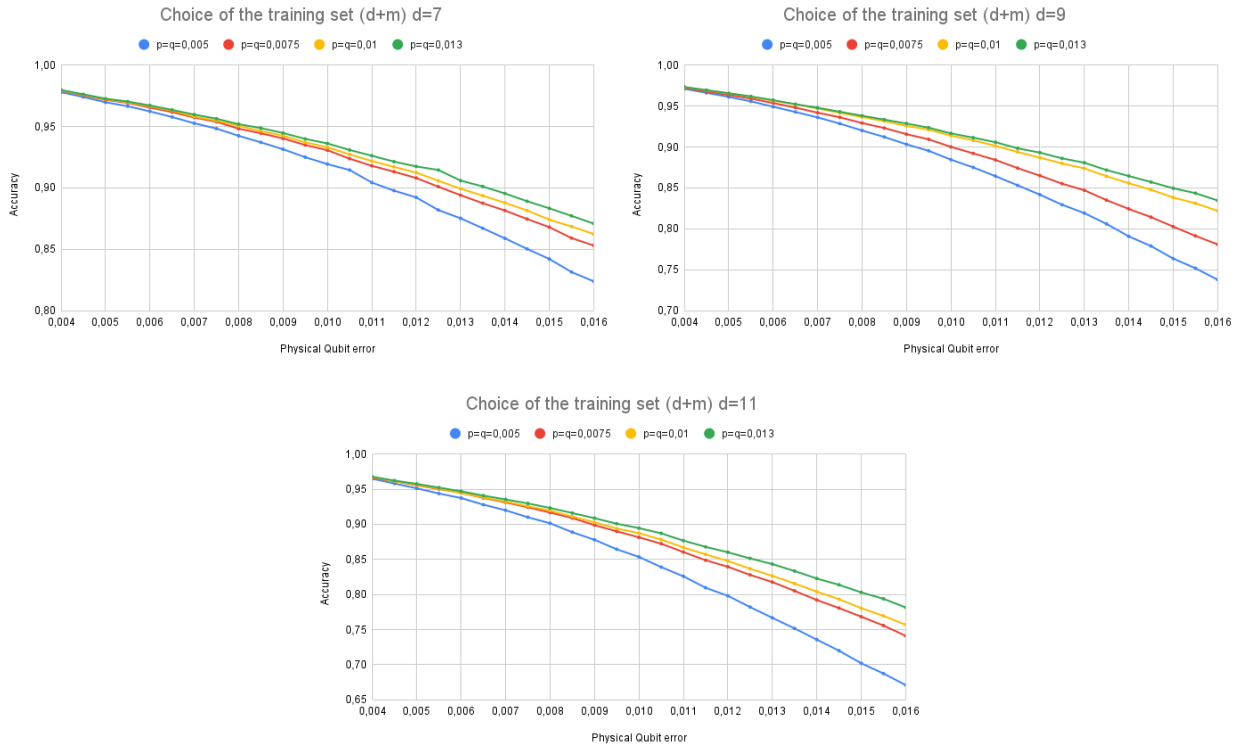


Figure 4.9: Accuracy of the HLD trained and tested on different depolarising and measurement error probabilities for different code distances d . The training set single-qubit error probability (p) and measurement error probability (q) is reported in the legend.

is possible to obtain an heatmap of the input pixels based on their relevance for the output probability. In the occlusion method several small regions of the image are systematically masked (all input pixels of the covered region are set to zero), and the changes in the loss function between the the occluded and standard image are recorded. The saliency value corresponding to a given masked region is given by:

$$Saliency(\text{region}) = [loss(image_{\text{original}}) - loss(image_{\text{masked}})]^2$$

We expect that a region that contributes more to the prediction of the model would change significantly the loss if masked. By shifting the masked region horizontally and vertically and repeating the process it is possible to construct the saliency map of the input features. For example, giving an image of 28x28 pixels and a occlusion patch of size 4x4 to mask the image, with a stride of 4 steps, an occlusion saliency map of size 7x7 is obtained. The salience map build in this way is then zoomed back to the original image resolution and overlaid to it.

Saliency maps help understanding if the trained HLD is performing as expected, and so allow to validate the algorithm, but they can also employed to better understand the errors of HLD in order to try to improve their performance. An original example of using salience maps to improve the algorithm itself is presented in Section 4.1.7.

Saliency map analysis

Occlusion method has been applied to the neural network of the HLD in order to understand how error syndromes are detected. Tests have been carried out for different code distances and neural networks trained on different error probabilities, the results obtained are very similar. In the following saliency maps of the input are reported for a depolarising noise model for codes of distance 11, using the neural network trained on an error probability $p = 0.13$. In fact this is the neural network that showed better results for codes of dimension 11, and with the depolarising noise model it is easier to find the more critical regions of the input. In the implementation of the occlusion method squared occlusion patches of the input of dimension 2×2 have been artificially set to zero. The dimension of the occluded window has been chosen in order to obtain a compromise between the smoothness and the granularity of the saliency map.

The first study regards how single errors or small chains of errors affect the final result, based on their position in the surface code (Figure 4.10). Two single X errors have been placed, one in the middle between the X sides (E_1), and one near an X side (E_2). The single decoder is able to correct E_2 but creates an X logical error when trying to decode E_1 . The Neural network is able to identify the introduction of a logical error, the region near E_1 contributes to the output probability while the region near E_2 doesn't contribute significantly. This is due to the fact that occluding the region of the error E_1 reverses the output of the neural network (X logical error can't be identified) thus significantly changing the loss. Two errors chains composed of three X errors have been placed one in the middle between the X sides (E_3) and one near an X side (E_4). The simple decoder is able to correct E_4 but creates a logical error correcting E_3 . The regions near the ends of the chain E_3 contribute more to the output than the regions near the ends of the chain E_4 . Also in this case this is due to the fact that covering the error syndrome of E_3 reverse the output of the CNN. Moreover, we have observed also in other examples, that errors placed near the centre of the

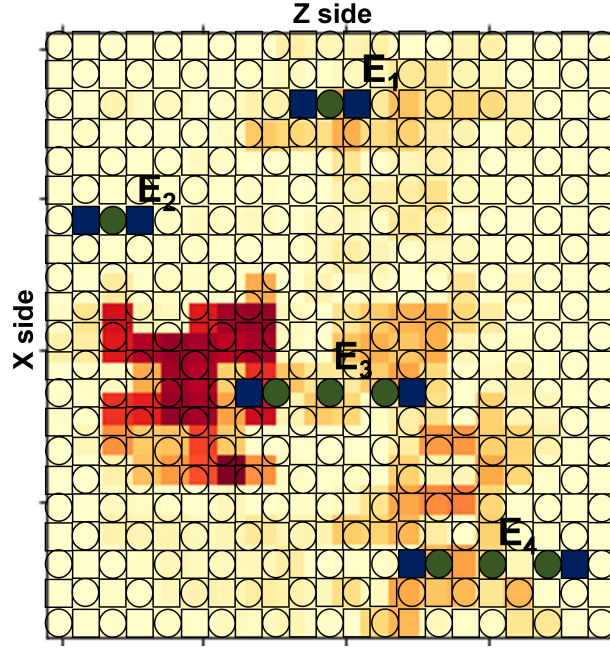


Figure 4.10: Saliency map of a manually generated error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubits with X errors are highlighted in green, while the measurement qubits that report the error signal are highlighted in blue. Two single X errors, E_1 and E_2 , and two errors chains composed of three X errors, E_3 and E_4 have been inserted in the surface code. The errors closer to the centre between the X sides (E_1 and E_3) are the ones that contribute more for the classification.

surface code influence more the output of the decoder. The left side of the chain E_3 contributes more to the output than the right side. Similar asymmetries can be found in other saliency maps and change with respect to the horizontal or vertical position of the errors chain as well as the presence and position of other errors in the code. For example in the case reported in Figure 4.10 the asymmetry is mainly due to the presence of the error E_2 . Note that there are active regions of the input between the right side of the chain E_3 and the left side of the chain E_4 . This first study considered only X depolarising errors, similar results can be obtained for single Z errors or small chains of Z errors. In this case the errors that contribute more to the output of the decoder are the ones placed near the middle between the Z sides.

For a depolarising error model the minimum weight perfect matching showed a better accuracy than the high level decoder, especially for codes of dimension 11. This means that there are cases where MWPM correctly identifies error chains while the high level decoder fails. An example is reported in Figure 4.11 where two error chains, composed of five errors, have been inserted in the code. The first chain is composed of X errors and is placed horizontally in the centre of the code, the second chain, composed of Z errors, is placed vertically in the centre of the code. Chains of this kind are very dangerous as they may be easily misidentified by a decoder and corrected by matching the error syndrome with the nearest border of the code thus creating a logical error. This is the case of the HLD, that is not able to identify the creation of an Y logical error after the correction of the simple decoder. In particular, the neural network classifies the error syndrome with no logical errors, this means that it is not able to find both the X and Z errors chains. Looking at the saliency map, reported in Figure 4.11, it is possible to notice that the most relevant regions of the input are the ones located between the extremity of the chains and the nearest border of the surface code.

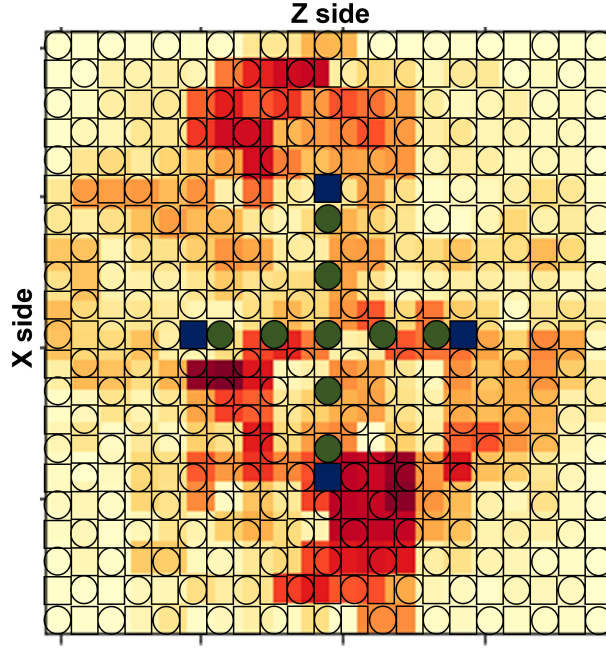


Figure 4.11: Saliency map of a manually generated error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubit with depolarising errors are highlighted in green, while the measurement qubit that report the error signal are highlighted in blue. Two errors chains composed respectively of five X (horizontal) and Z (vertical) errors have been inserted in the middle of the surface code. The HLD is not able to identify the creation of an Y logical error after the correction of the simple decoder.

This is probably due to the fact that the neural network decoder is looking for other errors in these regions in order to find the other extremities of the chains rather than matching X and Z syndromes with a single chain. This kind of error occurs almost every time a chain of length more than five appears in codes of distance 11. For codes of distance 9 and 7 the HLD correctly identifies chains of length 4 and 3 respectively. The MWPM is able to correct errors chains of length less than or equal $(d - 1)/2$, where d is the dimension of the code. This means that this algorithm can correct chains of length 5 for $d = 11$ surface codes, while the maximum number of correctable errors in the chains is equal to 4 for $d = 9$ and 3 for $d = 7$. This explains the performance reduction of the HLD with respect to MWPM for codes of distance 11. The introduction of more samples in the training set, containing error chains of length five, can be a solution to improve the performance of HLD. This conjecture has been studied in Section 4.1.7.

There are some cases where the HLD is able to correct an error syndrome of a depolarising noise model where MWPM fails. An example is reported in Figure 4.12, the error chain that gives problems to MWPM has been isolated by eliminating the other errors present in the code. After the correction of MWPM, an X error chain connecting the X sides of the code is introduced. On the other way, the HLD identifies all the logical errors introduced by the simple decoder. Looking at the saliency map in Figure 4.12 it is possible to notice that the central region of the surface code, where two isolated errors are present, is not really relevant for the output. On the other way, the region near the right X side, where there is a small error chain, is the one that mostly affects the output.

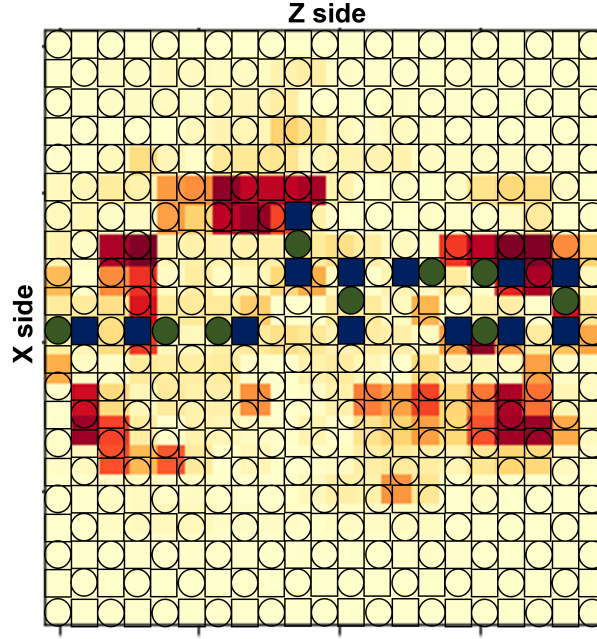


Figure 4.12: Saliency map of an error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubit with X errors are highlighted in green, while the measurement qubit that report the error signal are highlighted in blue. In this case the HLD is able to correct the error syndrome while the MWPM fails. This example has been generated using a depolarising error model (perfect measurements) with single-qubit error probability $p = 0.15$, the error chain responsible for the failure of MWPM has been extracted and analysed.

Performance enhancement with data augmentation driven by model explainability

Model explainability may be an interesting tool to improve the performance of ML algorithms. This can be done, for example, by analysing the critical issues in order to remove or attenuate them. In this section we have applied this idea to improve the performance of HLD.

Figure 4.11 showed that the CNN decoder fails to recognise some error chains of length 5 in a surface code of dimension 11. As these kinds of error chains can be corrected by MWPM, they account for the better performance of this decoder with respect to a HLD. In order to reduce this problem, we have generated an enhanced augmented data training set that includes some samples of these error chains. These special samples have been generated with the following procedure. An error chain, composed of five single-qubit errors of the same kind and on the same row or column, is added to the surface code. Both the row (or column) position of the chain, as well as the error locations, are randomly drawn. A single X error chain is added with probability $1/3$; with the same probability a single Z error chain is added and, in the other cases, both an X and a Z error chains are added.

The enhanced train set used to improve the performance of the CNN is composed of 7×10^6 samples divided as follows. 10^6 special samples containing error chains of length 5 as previously described. 5×10^6 samples with a single-qubit error probability $p = 0.1$ (standard train set employed in Section 4.1.5). 10^6 samples with a single-qubit error probability $p = 0.13$; a higher single-qubit error probability helps produce samples with longer error chains. The employed CNN is the same described in Section 4.1.5 for surface codes of dimension 11. As the enhanced train set contains samples of different kind, the training batch size has been set to 128 to increase training

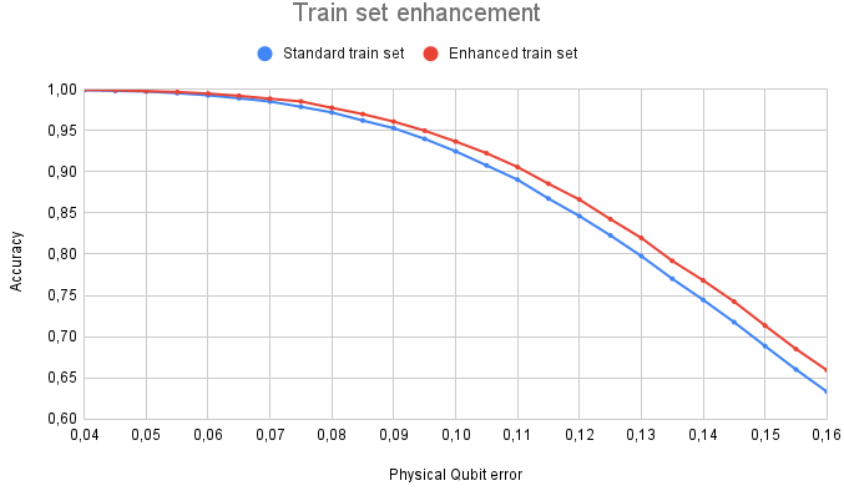


Figure 4.13: Performance comparison between HLD trained on a standard train set with single-qubit error probability $p = 0.1$ and the model trained on the enhanced train set that includes samples of error chains of length five.

stability. The model has been trained for four epochs, starting from the parameters trained in Section 4.1.5 on the standard train set with a single-qubit error probability $p = 0.1$. The trained model has been tested on different single-qubit error probabilities, the same test sets introduced in Section 4.1.5. Figure 4.13 reports the performance improvement obtained with the enhanced train set, with respect to the standard train set employed in Section 4.1.5. The performance improvement is significant, about 1% for single-qubit error probability $p = 0.1$ and up to 2% for higher single-qubit error probabilities. Moreover, the new model is able to correctly decode the example reported in Figure 4.11. The new saliency map, obtained with this model, is reported in Figure 4.14. A comparison with Figure 4.11 shows how the HLD trained on the enhanced dataset gives the same importance to X and Z error signals. This decoder is able to better understand the importance of the error signals at the end of the error chains, as well as the central region of the surface code.

4.1.8 Conclusions

Neural network based decoders have proved to be excellent algorithms for the decoding of surface codes due to their constant execution time, good accuracy and adaptability to different noise models. The use of a convolutional architecture, with respect to a dense architecture, helps scaling to higher distance codes. However, as incrementing the distance of the code increments exponentially the possible error syndromes, larger datasets are required for correct training. This makes difficult to apply a simple convolutional architecture to decode really high distance codes. Convolutional neural networks decoders remain interesting as they may be used as the first step of a more sophisticated decoder to process local information [5]. Moreover, it is likely that in the future only small distance surface codes will be available for the first tests.

The results obtained in this work are promising and helpful to improve the accuracy obtained with HLD based on convolutional neural networks. Different CNN architectures and training strategies have been tested for different code distances and noise models. The results suggest that using

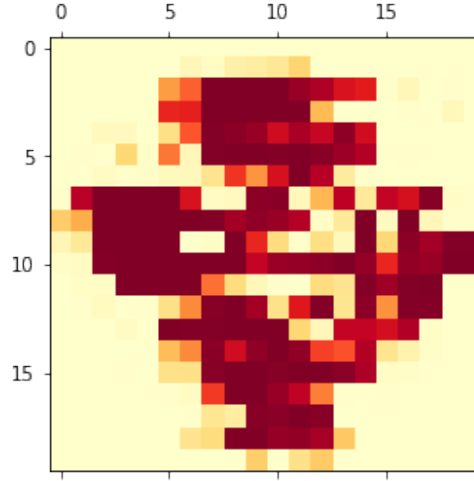


Figure 4.14: Saliency map of the same error syndrome reported in Figure 4.11, decoded with the HLD trained on the enhanced train set. The dark red regions are the ones that contribute more to the output, they are placed near the error signals and the centre of the surface code. This HLD is able to correctly identify the presence of error chains from a sparse error syndrome.

a training set with a higher error probability helps improving the performance of the decoder, and that successful training benefits can be obtained from a pre-training based on examples on a lower error probability. It was also shown how it is possible to use dilated convolution to reduce the number of parameters that can be trained with decoders based on very deep convolutional neural networks. This suggest a possible way to try to scale the convolutional neural networks based decoder to larger surface codes.

Finally, explainability methods have been proposed and applied to get insights on how error syndromes are classified by the neural network decoder. A better knowledge of the causes of the failure of the decoding is in fact fundamental to improve the performance, robustness and confidence in neural network HLD for real applications. In this respect an original example of use of the salience maps to improve the algorithm accuracy has been presented.

4.2 Quantum noise modeling through reinforcement learning

The background for this work can be found in Section 2.3 with particular regards to the Section 2.3.3 where the noise channels used in this work are presented. Moreover Section 3.2.5 provides an overview of reinforcement learning.

4.2.1 Abstract

In the current era of quantum computing, robust and efficient tools are essential to bridge the gap between simulations and quantum hardware execution. In this work, we introduce a machine learning approach to characterize the noise impacting a quantum chip and emulate it during simulations. Our algorithm leverages reinforcement learning, offering increased flexibility in reproducing various noise models compared to conventional techniques such as randomized benchmarking or heuristic noise models. The effectiveness of the RL agent has been validated through simulations and testing on real superconducting qubits. Additionally, we provide practical use-case examples for the study of renowned quantum algorithms.

4.2.2 Introduction

One important unsolved technological question regards the practical applicability of NISQ [288] computers. Despite the expectation that quantum computers will outperform classical computers in certain computational tasks [49, 50, 289], the usability and reliability of NISQ devices are hindered by a large error rate. These errors arise from gate infidelities, unwanted environmental interactions, thermal relaxation, measurement errors, and cross-talk [290–294]. Currently, there are techniques to mitigate these errors [71–74, 295, 296]. However, it has been demonstrated that any quantum circuit for which error mitigation is efficient must be classically simulable [297]. Therefore, it is widely regarded that quantum advantage will only be achieved with future generations of fault-tolerant quantum devices [15, 260, 298–300].

Despite the imperfect results obtained on NISQ devices, numerous algorithms have been developed and implemented on this hardware. Machine learning-inspired models, in particular, have shown promising results in recent years [40, 41, 301–306]. Testing and developing new algorithms in the NISQ era can be challenging due to limited access to quantum chips. Furthermore, the currently accessible quantum computers in the cloud are highly demanded, resulting in lengthy waiting queues [307]. In this context, emulating the noise of these devices emerges as an alternative to accelerate circuit testing.

This work aims to develop a model capable of learning hardware-specific noise for use in circuit simulations. Our novel approach employs reinforcement learning to train an agent to add noise channels that replicate the noise pattern of a specific quantum chip. This method minimizes heuristic assumptions about the noise model, thereby enhancing the adaptability and generalization properties. The algorithm has been validated on both simulations and real quantum devices hosted at the Quantum Research Center [308] of the Technology Innovation Institute in Abu Dhabi, demonstrating its ability to accurately predict different noise patterns.

The Qibo framework [309, 310] was used for the realization of this work. It provides `Qibo` [309, 310] as a high-level language API for crafting quantum computing algorithms, `Qibolab` [35, 311]

as a tool for quantum control, and `Qibocal` [312] for conducting quantum characterization and calibration routines. All the code developed for this work is available on GitHub². It is possible to use the code to reproduce the results as well as testing the algorithm under different noise conditions. The code is intended to be easily customizable in order to allow users to define their own RL agents for specific applications.

4.2.3 Related works

The characterization and modeling of quantum noise is an important research area that has yielded significant results in recent years [19–21]. While standard noise tomography techniques such as randomized benchmarking are useful for obtaining a standardized evaluation of noise, they lack the ability to capture specific characteristics that are essential for realistic simulations. These techniques can be improved by incorporating non-Markovian noise and realistic finite-time gate models, as proposed by Brillant et al. [313]. Other studies focus on reducing the resources required to perform these techniques and on maximizing the information extracted from experimental data [314–316]. Another line of research involves developing efficient algorithms to fit a Lindbladian to tomography data [314, 317, 318], providing compact noise models and quantitative measures of non-Markovianity. Finally, it is worth mentioning the tensor network approach proposed by Mangini et al. [319], which enables the study of correlated noise in systems with up to 20 qubits.

Various techniques based on machine learning have also been proposed for noise characterization. For instance, Zlokapa et al. introduced a deep learning method to identify noise patterns, aiming to reduce noise levels during the transpilation process [18]. Canonici et al. trained machine learning models on measurement data to infer noise parameters in neutral-atom quantum processors and additionally employed a reinforcement learning agent to design corrective control pulses that mitigate measurement noise [320].

Separately, reinforcement learning has been explored for a variety of quantum control and calibration tasks. In quantum gate design, several studies apply deep RL to optimize pulse shapes or gate parameters. For instance, Niu et al. trained an RL agent to optimize two-qubit gate pulses in the presence of leakage and stochastic errors [26]. In superconducting systems, Baum et al. experimentally employed deep RL to autonomously design single- and two-qubit gates without requiring knowledge of the device Hamiltonian [34].

Reinforcement learning has also been applied to improve qubit readout and measurement [321]. Chatterjee et al. demonstrated on superconducting hardware that an RL agent can tailor readout pulses to achieve state-of-the-art fidelity while significantly reducing readout time [322]. Similar applications of RL methods have been proposed for automated calibration and tuning. For example, Crosta et al. [323] developed a model-free RL agent that periodically adjusts calibration parameters to maintain optimal performance, without relying on an explicit noise model of the environment. Related frameworks have been introduced by K. Young [324] and Nguyen et al. [325] to accelerate the calibration of superconducting logic gates.

Finally, several works have applied RL to quantum error correction and quantum algorithm design. Nautrup et al. used RL to adapt surface-code error-correction circuits to unknown noise channels [326]. Olle et al. extended this approach to discover new error correction codes and encoding

²<https://github.com/qiboteam/rl-noisemodel>, 2025.

circuits tailored to a given hardware’s gate set, connectivity, and noise model [327].

4.2.4 Algorithm implementation

This section provides a detailed explanation of the proposed noise modeling algorithm, with a particular emphasis on the transformation of quantum circuits into input feature vectors suitable for neural network processing, as well as a comprehensive description of the policy, the training process and the datasets.

Circuit representation

To train the RL agent, we need to represent a quantum circuit as an array that can be readily processed by the policy neural network; we refer to this array as the Quantum Circuit Representation (QCR). In the following, we refer to a circuit moment as a collection of gates that act on a non-overlapping set of qubits and are conceptually executed in parallel. The circuit depth is defined as the total number of circuit moments, assuming that the division of the circuit into moments is optimized to minimize their number. The QCR has a shape of $[qubits, depth, encoding]$. The first dimension corresponds to the circuit’s qubits, while the second dimension represents the circuit’s moments, *i.e.*, a collection of gates that can be executed in parallel. The *encoding* dimension encodes the information regarding gates and noise channels acting on a specific qubit at a specific circuit moment. Its dimension is determined by the total number of native gates and noise channels the model allows for. In detail, considering a set of n single-qubit native gates $\{G_i\}$, the initial n entries of the *encoding* will contain a *one-hot* encoding of our native set. Namely, in the presence of the gate G_i , all the entries are set to 0 except for the i -th entry that is set to 1. In the presence of a two-qubit gate, CZ in our case, the entry relative to both the target and control qubit is set to 1. Specifically, in our case, we use $n = 2$ to encode the presence of an R_x or R_z gate. The $n + 1$ entry is set to 1 if a two-qubit gate is present and 0 otherwise, with the CZ gate being the native two-qubit gate in this work. For single-qubit circuits, this entry is always zero. The $n + 2$ entry encodes the rotation angle of single-qubit gates, normalized to the range $[0, 1]$. The remaining entries detail the parameters of the noise channels in the following sequence: Depolarizing channel, Amplitude damping channel, R_z coherent error, and R_x coherent error. If a noise channel is absent, the corresponding parameter value is set to zero. An example of the QCR for a two-qubit circuit, obtained using this procedure, is illustrated in Figure 4.15.

To enable the agent to adapt to circuits of varying depths, we further introduce the concept of kernel size k , similar to the kernels used in CNNs [328]. The kernel size establishes a “window” that restricts the number of circuit moments the agent can observe at any given time. For instance, with $k = 3$, the agent only observes the current moment and the immediately preceding and following ones. The window’s center starts from the first moment and slides one position at each step until the circuit’s end is reached. To ensure that the first and last moments of the circuit are at the center of a full-sized window, we apply a symmetric zero-padding of length $(k - 1)/2$ at the edges of the QCR. This approach is based on the heuristic assumption that a gate’s noise is most influenced by its temporally proximate gates. At a given moment m , therefore, a window $(m - \frac{k-1}{2}, m + \frac{k-1}{2})$ is extracted from the complete QCR of the circuit, effectively yielding a fixed dimension $[qubits, k, encoding]$ tensor that is fed as input to the agent.

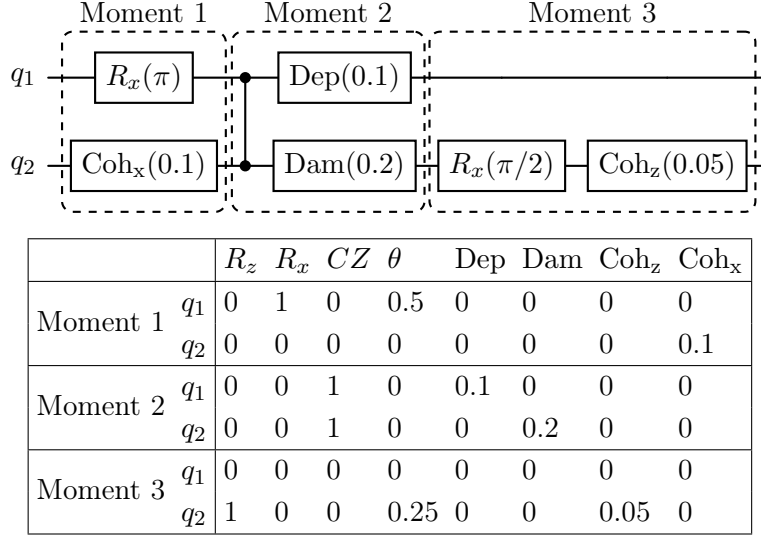


Figure 4.15: Example of a two-qubit quantum circuit (top) and its vector representation (bottom). A two-qubit circuit of depth three is represented as a tensor of size (2, 3, 8), where the first entry identifies the qubit, the second denotes the circuit moment, and the third specifies the type of gate or noise channel.

Policy

Agent's actions can be depicted following the same QCR schema. They are represented as matrices, whose individual rows and columns represent the qubits and the distinct noise channels respectively. Each entry of the action matrix is computed by a forward pass through the policy network, which consists of three main components. The Feature Extractor (FE) takes as input the QCR ($x^{(QCR)}$) and maps it to a high dimensional latent feature space $x^{(feat)}$,

$$x^{(feat)} = \text{FE}(x^{(QCR)}).$$

To efficiently capture the correlations between different qubits and moments, we employ a CNN, which should be suited for processing two-dimensional data. The Actor Policy (A_π), is a simple Multi Layer Perceptron (MLP) which takes as input the latent features and computes the actual action the agent is going to take ($x^{(action)}$).

$$x^{(action)} = A_\pi(x^{(feat)}).$$

Moreover, as DM is an actor-critic policy we have a separated critic policy C_π with the same MLP architecture but independent weights, whose role is to rate the action selected by the actor $x^{(rating)}$. The rating is extracted from the latent features,

$$x^{(rating)} = C_\pi(x^{(feat)}).$$

A schematization of the complete policy NN is reported in Figure 4.16.

We have performed different tests to determine the optimal architecture and number of parameters for the NNs used in our policy. These hyperparameters slightly change with the number of qubits. Here, we report the best characteristics observed for circuits with one and three qubits. For the feature extractor, we used a single convolutional layer with 16 filters for single-qubit circuits and

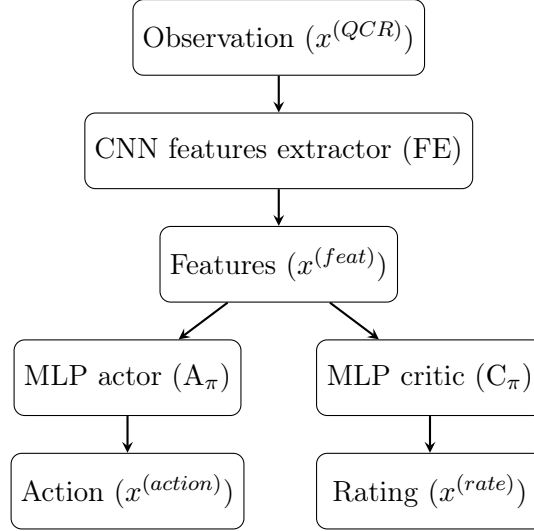


Figure 4.16: Schematization of the policy neural network for the DM algorithm. A common feature extractor (FE), composed of a CNN, maps the QCR to a high-dimensional latent feature space. The resulting feature vector serves as input to both the actor policy (A_π) and the critic policy (C_π). The actor policy is responsible for selecting the best action for the agent, while the critic policy estimates the future reward for the action chosen by the actor.

32 filters for three-qubit circuits. This convolutional layer is followed by a dense layer with a ReLU activation function. The optimal number of output features for this dense layer is 64 for single-qubit circuits and 32 for three qubits circuits. Both the actor and critic policies are implemented as MLPs with a hidden dense layer containing 256 neurons. The total number of trainable parameters in the entire policy NN is on the order of 10^4 . We experimented with increasing the number of parameters by adding additional convolutional layers and increasing the number of features in the feature extractor. However, we observed overfitting when the total number of parameters approached the order of 10^5 .

As detailed in Section 4.1.3, in this work we consider a set of four possible noise channels: depolarizing, amplitude damping, and coherent errors R_x and R_z . This means, that the output of our actor policy A_π is going to be a $(nqubits, 4)$ tensor encoding the predicted parameters of the inserted noise channels at that specific moment. A value of zero corresponds to no noise channel of that type inserted. For example, in a two-qubit circuit, the action

$$\begin{bmatrix} 0.1 & 0 & 0 & 0.2 \\ 0 & 0.05 & 0.3 & 0 \end{bmatrix}$$

indicates that a depolarizing channel with a depolarizing probability of 0.1 and an R_x coherent error with a rotation angle of 0.2 are added to the first qubit. Simultaneously, an amplitude damping channel with a damping probability of 0.05 and an R_z coherent error with a rotation angle of 0.3 are added to the second qubit. A sensitive hyperparameter of the algorithm is the maximum allowed value for each noise channel's parameter, P_{max} , which controls in practice the span of the search space. A smaller P_{max} yields a narrower parameter space making convergence faster. However, a too small value might limit the expressive capability of the model, precluding the agent the access to possibly large reward areas of the parameter space. We obtained good results by setting P_{max} as twice the effective depolarizing parameter derived from a RB experiment.

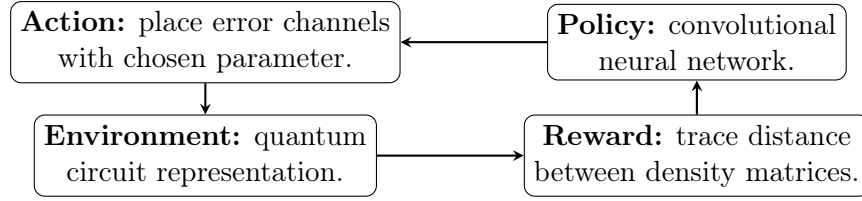


Figure 4.17: Training process of the RL algorithm. At each step the RL agent receives an observation of the environment, the quantum circuit representation of a non-noisy circuit. The agent policy performs an action that consists in putting any number of noise channels at a given circuit moment. At the end of the circuit the reward is computed to minimize the trace distance of the density matrix of the original noisy circuit and the reconstructed one.

Training procedure

The agent is trained to add noise channels to a noiseless quantum circuit in order to reproduce the noise pattern observed when executing the circuit in the presence of some target noise (whether simulated or real when executing on the hardware). This process is outlined in Figure 4.17. Each training episode begins with the agent receiving a randomly selected noiseless quantum circuit from the training set. Then, for each of the circuit’s moments, the agent observes the current QCR and takes an action: any combination of the selected set of noise channels, together with their corresponding noise parameters, is inserted in that precise moment, yielding an updated QCR.

The agent receives the reward at end of each episode and the policy’s NN parameters are then updated to maximize future rewards. The reward is taken to be a function of the distance between the target DM (ρ_{true}) and the DM of the noisy circuit generated by the agent (ρ_{agent}) under some selected metric. We have tested different metrics such as a simple element-wise MSE, density matrix fidelity, and Trace Distance (TD) [329]. Trace distance proved to be the most suitable metric, as it is both easy to compute and phase-invariant. Moreover, as shown by some of the results presented in Section 4.2.6, most of the information about the noise is contained in the diagonal elements of the density matrices [330]. Finally, this metric is specifically designed to quantify the experimental distinguishability between two quantum states.

TD can not be directly used as reward, so we have explored different functional forms including $1/\text{TD}^n$ and $-\log(\text{TD})$. The most effective form, as determined through empirical testing, has been found to be

$$R(\rho_{\text{agent}}, \rho_{\text{true}}) = \frac{1}{\alpha \text{TD}(\rho_{\text{agent}}, \rho_{\text{true}})^2 + \epsilon},$$

where ϵ is a small parameter introduced to prevent numerical instabilities and the hyperparameter α can be used to normalize the reward. This equation essentially penalizes high values of the TD with a low reward. The average episode reward, denoted as $\mathcal{R}$, is determined by averaging R over all the training circuits within a batch of episodes. A summary of the training procedure is provided in Algorithm 1. After numerous episodes, the agent is expected to learn the optimal placement of noise channels in a noise-free circuit to reconstruct the final density matrix of the real noisy circuit. Once trained, our algorithm should be capable of generalizing to previously unseen circuits, thereby enabling realistic noisy simulations.

Algorithm 1 Agent training procedure.

```

for episode in n_episodes do
  circuit = random_extraction(training_set)
  for moment in circuit do
    observation = agent.make_observation(circuit, moment)
    action = agent.action(observation)
    circuit.add_noise(action)
  end for
  generated_dm = extract_density_matrix(circuit)
  reward = compute_reward(generated_dm, ground_truth_dm)
  agent.update_policy(reward)
end for

```

Dataset generation

The RL algorithm requires training, testing, and evaluation datasets, which consist of ensembles of random quantum circuits and their corresponding final DMs ρ_{true} . These DMs serve as ground truth labels during the training phase of the algorithm (Section 4.2.4). In simulations, the ρ_{true} are computed analytically. However, for circuits executed on hardware, they can be obtained using quantum state tomography [331] or more efficient techniques such as classical shadow state reconstruction [332–334].

In this study, we utilize circuits with gates in the native gate set $\{R_x(\pi/2), R_z(\theta), CZ\}$ implemented in the quantum devices of the Technology Innovation Institute [308]. We also conducted preliminary tests using the native gates set of IBM quantum hardware [335], which employs the *CNOT* gate as the two-qubit entangling native gate. Modifying the native gates in our algorithm is a simple process and has not led to any significant changes in performance. The train set for single-qubit circuits is composed of circuits with a fixed number of Clifford gates extracted randomly. Clifford gates are chosen due to, both, their lower simulation cost and their large use in randomized benchmarking and shadow state estimation [333, 334, 336, 337]. In our specific case, since we consider R_x and R_z gates, the only allowed rotation parameters are multiples of $\pi/2$. For three-qubit circuits we have used a more sophisticated training set. Half of the training set is composed of circuits with a fixed depth with the gates and parameters extracted randomly. The second half of the training set is composed of Clifford circuits implementing randomly chosen three-qubit Clifford unitaries [338]. These circuits do not have a fixed number of gates or fixed depth. We observed an improvement in the generalization properties of the algorithm when using this mixed training set. For the performance evaluation we have used two different datasets. The first dataset is composed of non-Clifford circuits with fixed depth. The results obtained with this set demonstrate that the algorithm, even if trained on Clifford circuits, maintains its ability to generalize. The second dataset for performance evaluation, consist of Clifford circuits with varying depths. We have used this dataset to fit the RB noise model and compare it with the RL agent.

For the datasets used in simulations, we defined custom noise models. Specifically, we used two different custom noise models for circuits with one and three qubits. For single-qubit circuits, we applied a depolarizing channel with a depolarizing parameter of $\lambda = 0.02$ after each R_z gate, and an amplitude damping channel with decay parameter $\gamma = 0.03$ after each R_x gate. A coherent $R_x(\theta')$ error with angle $\theta_x = 0.04 \cdot \theta$ is introduced after each $R_x(\theta)$ gate. Similarly, a coherent

Noise	Gates	λ	γ	θ_x	θ_z
1 qubit	$R_x(\theta)$	0	0.03	$0.04 \cdot \theta$	0
	$R_z(\theta)$	0.02	0	0	$0.02 \cdot \theta$
3 qubits (high noise)	$R_x(\theta)$	0	0.03	$0.04 \cdot \theta$	0
	$R_z(\theta)$	0.02	0	0	$0.03 \cdot \theta$
	CZ	0.02	0.03	0	0
3 qubits (low noise)	$R_x(\theta)$	0	0.01	$0.015 \cdot \theta$	0
	$R_z(\theta)$	0.015	0	0	$0.02 \cdot \theta$
	CZ	0.015	0.01	0	0

Table 4.4: Noise models used in simulation to train and evaluate the RL algorithm in different conditions. The columns show the noise channels parameters: λ for the depolarizing channel, γ for the amplitude damping channel and θ_x and θ_z for the coherent error R_x and R_z respectively. The rows indicate the gates subject to the noise channel, a zero noise parameter means that the noise channel is not applied to that gate.

$R_z(\theta')$ error is added after each $R_z(\theta)$ gate, with $\theta_z = 0.02 \cdot \theta$. This noise model is not intended to be realistic but to test our algorithm on a gate dependent noise model. For three-qubit circuits, we used a similar noise on rotation gates and added a depolarizing channel with a depolarizing parameter of 0.02 and an amplitude damping channel with a decay probability of 0.03 after each CZ gate. In Section 4.2.6, we have also used a lower error rate noise model to test the algorithm’s ability to generalize to different noise models. A comprehensive summary of the noise parameters used in the simulations is provided in Table 4.4. Generating the training set is straightforward for simulations where the DMs are computed analytically. However, performing full state tomography to obtain these values on quantum hardware can be time-intensive. For this reason, we conducted preliminary tests on one and three-qubit simulated circuits to determine the minimal length of the training set. We trained the algorithm on datasets of varying sizes (from 10 to 10^3), using the noise models introduced in Table 4.4, and assessed the performance on a common evaluation set. We found that for three-qubit circuits, it is sufficient to train the algorithm on a dataset with more than 100 circuits to avoid overfitting and achieve optimal performance. For single-qubit circuits, a dataset with about 20 circuits is sufficient to obtain nearly optimal performance. This result may vary with the complexity of the noise model; however, it demonstrates that dataset generation is not a bottleneck for quantum chips with a small number of qubits.

4.2.5 Results

The following sections detail the results obtained from applying the proposed algorithm in both simulations and on quantum hardware.

Simulations

Our study begins by training the RL agent to emulate the custom noise model, introduced in Section 4.2.4, on single-qubit circuits. To train the model, we generated a dataset of 100 random Clifford circuits of depth 10. We have used 80% of the dataset for the training, reserving the remaining 20% for the test set. The performance of the agent is evaluated by determining the average fidelity between the DMs it produces and the actual noisy DMs. These values are computed across all episodes for both the training and test sets, as shown in Figure 4.18. The training process

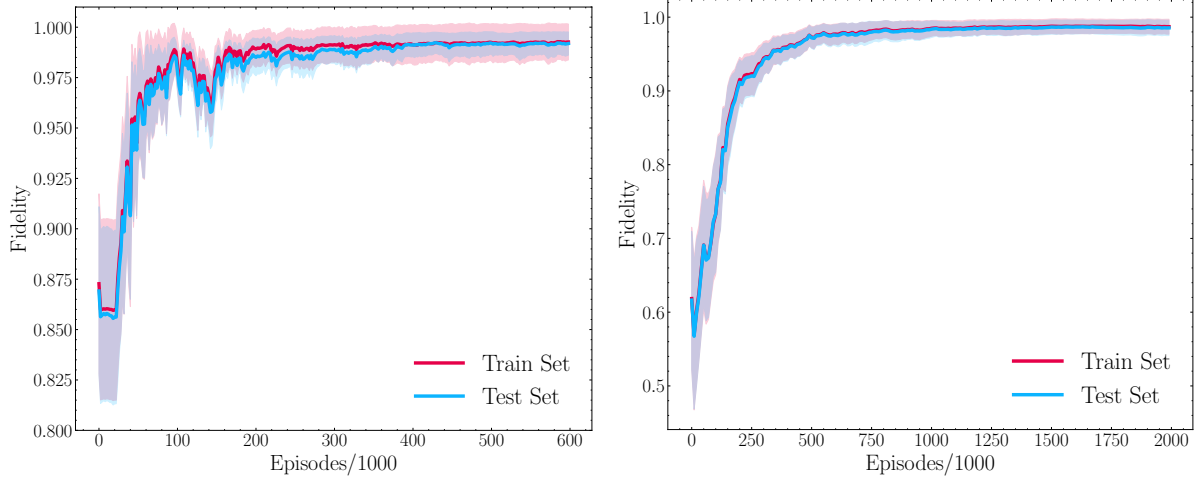


Figure 4.18: Average density matrix fidelity throughout the training process of the RL agent on single-qubit circuits (left) and three-qubit circuits (right). The metrics were evaluated on a dataset of 100 circuits for the single-qubit case and 800 circuits for the three-qubit case, using 80% for the training set and 20% for the test set. Shaded regions represent the standard deviation.

In the single-qubit case, convergence of the training process was reached after approximately 4×10^5 episodes, achieving an average fidelity of about 0.99 on the test set. For the three-qubit case, convergence was reached after approximately 1.5×10^6 episodes, achieving an average fidelity of about 0.98 on the test set. No significant overfitting was observed in either case.

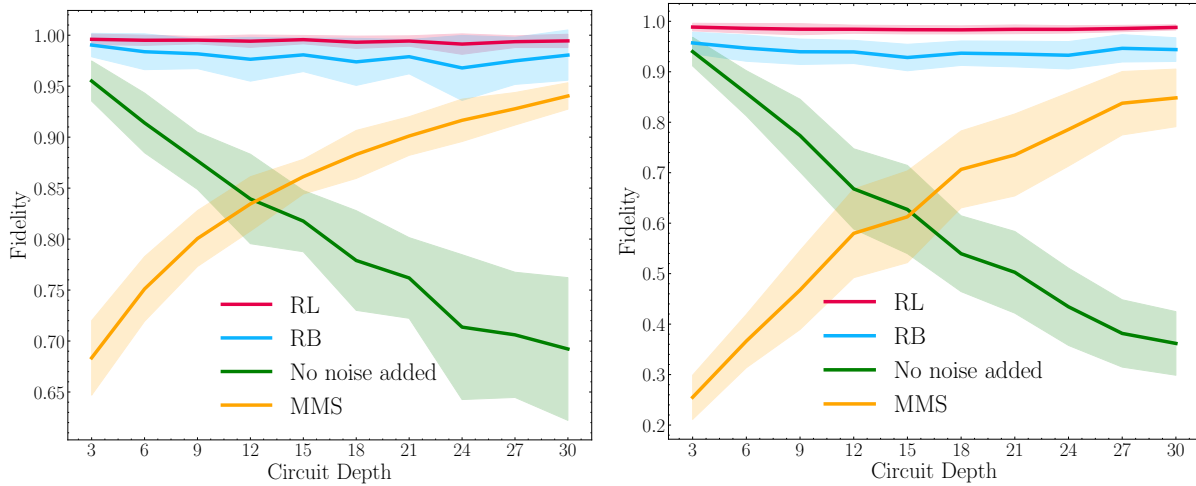


Figure 4.19: Performance evaluation of different noise models on Clifford circuits with depths varying from 3 to 30 for the single-qubit case (left) and the three-qubit case (right). Performance is measured using average density matrix fidelity; shaded regions represent the standard deviation. The RL agent was benchmarked against the RB method, the noiseless scenario, and the maximally mixed state (MMS). The absolute advantage of using the RL model is evident when compared to the case with no noise channel insertion. Moreover, in both cases, the RL model outperforms the RB model. Fidelity between the noisy circuit and the MMS is provided as a reference for the overall noise level—as the depth increases, the circuit’s behavior can approximate that of a completely depolarizing channel, converging toward the MMS.

	Train set	Test set	Evaluation set	Clifford depth 30
1 qubit	0.99 ± 0.01	0.99 ± 0.01	0.993 ± 0.003	0.99 ± 0.01
3 qubits	0.98 ± 0.02	0.98 ± 0.02	0.98 ± 0.01	0.98 ± 0.01

Table 4.5: Summary of the RL model performance across different training, test, and evaluation sets for single- and three-qubit circuits. Performance is reported as the average fidelity between the reconstructed and original density matrices, with standard deviation as the error. The training and test sets consist of Clifford circuits for the single-qubit case, and both Clifford and non-Clifford circuits for the three-qubit case. The evaluation set includes non-Clifford circuits of depth 15 in both cases. Fidelity on Clifford circuits of depth 30 is also reported as an alternative evaluation set.

converges after roughly 4×10^5 episodes, achieving an average fidelity of about 0.99 on the test set. The RL agent effectively learns to simulate the noise, exhibiting no signs of overfitting. For the evaluation set, we used a dataset of 100 random non-Clifford circuits with a depth of 15. The average fidelity of the RL agent on the evaluation set is 0.993, with a standard deviation of 0.003. This result shows that the agent is able to correctly generalize to non-Clifford circuits. A summary of the performance obtained by the model during the training and testing is reported in Table 4.5.

To further assess the model’s generalization capability, we evaluate it on random Clifford circuits of varying depths, from 3 to 30. Also in this case we evaluate the performance using the average fidelity between the DMs generated by the model and the actual noisy DMs. Figure 4.19 compares the performance of the RL agent with the RB noise model described in Section 4.1.3). We also offer a comparison with two limit cases: the DMs of noiseless simulation and the DM of the maximally mixed state (MMS). The noiseless simulation serves as a baseline, representing an agent that does not apply any noise channels, and thus provides a reference for evaluating the benefit of using the RL agent. The maximally mixed state (MMS) is included as an additional reference to indicate the total noise level; as circuit depth increases, the system’s behavior can approximate that of a completely depolarizing channel, converging toward the MMS. The RL agent demonstrates its adaptability to circuits of different depths, consistently outperforming RB. This performance suggests that while RB categorizes all noise sources as depolarizing, our algorithm can discern the specific characteristics of the noise.

We extended our simulation to three-qubit circuits to evaluate performance in the presence of two-qubit gates. We have used the high noise model on three qubits reported in Table 4.4. The training set consists of 800 three-qubit circuits, divided into 80% for training (640 circuits) and 20% for testing (160 circuits). This training set is composed of both Clifford and non-Clifford circuits as described in Section 4.2.4. The evolution of the average DMs fidelity throughout the training process is shown in Figure 4.18. While the algorithm is capable of learning the noise, the convergence is slower than in single-qubit circuits due to a larger action space requiring more episodes for exploration. Convergence is reached after approximately 1.5×10^6 episodes achieving an average fidelity of about 0.98. No signs of overfitting can be observed during the training phase. For evaluation, we used a dataset of 200 random non-Clifford circuits of depth 15, where the RL agent achieved an average fidelity of 0.98 with a standard deviation of 0.01.

Mirroring our previous single-qubit circuit experiments, we evaluated the performance of our RL agent against both the RB method and the limit cases of noiseless circuits and MMS across a variety of circuit depths. This comparison is reported in Figure. 4.19. In these scenarios, the

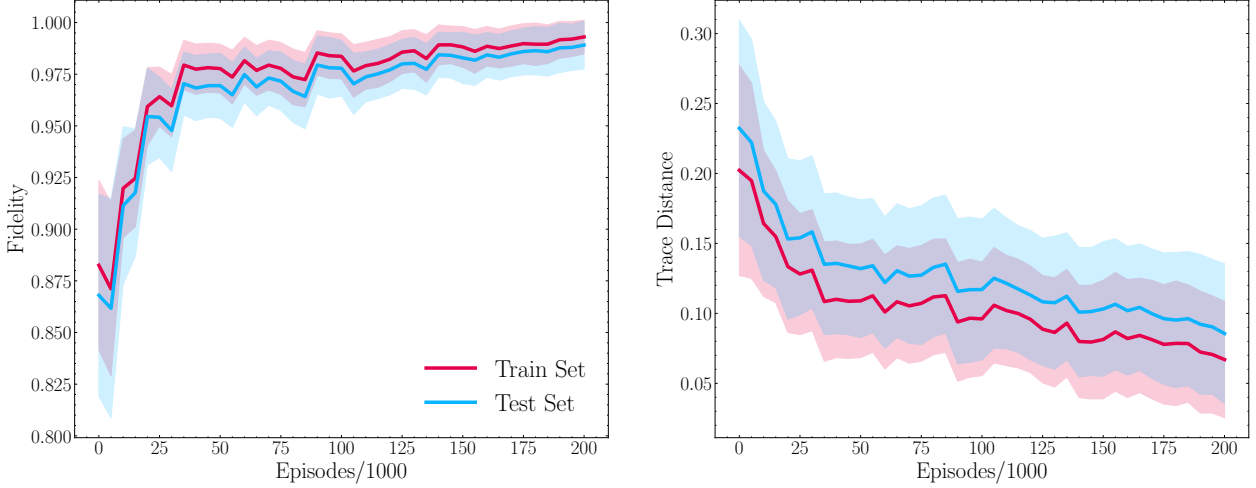


Figure 4.20: Average density matrix fidelity (left) and trace distance (right) throughout the training process of the RL agent on single-qubit circuits executed on a superconductive quantum chip. The metrics have been evaluated on a dataset of 60 circuits using 80% for the train set and 20% for the test set; shaded regions represent the standard deviation. Convergence was achieved after approximately 2×10^5 episodes, reaching an average fidelity of 0.99 on the test set. No evident signs of overfitting were observed.

RL agent consistently demonstrates its adaptability to circuits of different depths, surpassing the performance of the RB method for all circuit lengths.

A summary of the RL model performance on training, test and evaluation sets is reported in Table 4.5.

Quantum hardware

To test our algorithm on real quantum hardware, we used a single qubit in a superconducting transmon chip [339]. This 17 qubits chip has been produced by QuantWare³ and hosted at the Technology Innovation Institute of Abu Dhabi. The single-qubit gate fidelity, obtained with RB, is 0.996 with a readout fidelity of about 0.96. To compute the DMs of the circuits, we used state tomography, running 4×10^3 shots to compute each matrix. We attempted to mitigate measurement noise in advance to improve the fidelity of the DMs [340]. However, we observed that the RL agent performs better in learning the noise model without measurement error mitigation.

To train the algorithm, we collected a dataset consisting of 60 random circuits of depth 10, employing 80% of these circuits for the training set and the remaining 20% for the test set. Figure 4.20 reports the average DM fidelity and TD during training for the first $2 \cdot 10^5$ episodes, after this episode no performance improvements have been observed. At convergence the model reached an average fidelity of 0.99, with no evident signs of overfitting. This result is similar to the one obtained with simulations for single-qubit circuits.

Using a qubit with high gate fidelity makes the training process more challenging. The ground truth density matrices are affected by both gate errors, which are learned by the RL algorithm, and by shot noise and measurement errors. In the high gate fidelity regime, measurement noise and shot noise can have an impact similar to that of gate noise. These errors in the density matrices make the reward signal less precise, thereby worsening the convergence of the training process. For our dataset, we have computed the average error on the trace distance introduced by measurement

³<https://www.quantware.com>, 2025.

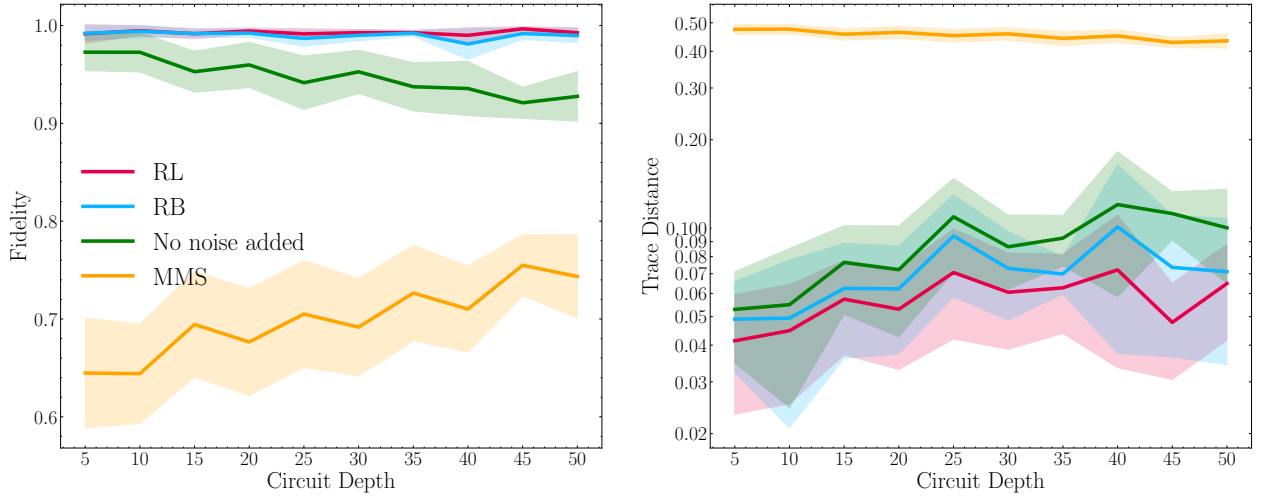


Figure 4.21: Performance evaluation of different noise models on single-qubit Clifford circuits with depths ranging from 5 to 50. The performance is assessed using average DM fidelity (left) and trace distance (right), with respect to the real noisy DMs obtained from execution on quantum hardware. The RL agent is benchmarked against the RB method, the noiseless scenario, and the maximally mixed state (MMS). In the trace distance plot, a logarithmic scale is used to enhance the distinguishability between RB and RL. The lower level of noise, compared to the simulation, can be inferred from the fidelity with the MMS, which does not reach the same values as in the simulation. A comparison between the RL and RB models shows that the former outperforms the latter on average; however, due to large errors, their performances are comparable within one standard deviation.

errors, shot noise and the additional gates needed to perform state tomography. The obtained value is 0.036, which explains the large error bars observed during training, as shown in Figure 4.20. To mitigate this problem it would be useful to train the algorithm on longer circuits so that more gate errors can accumulate. In our case, to help the training process, it has been fundamental to set the maximum noise parameter value P_{max} to a low value. As described in Section 4.2.4, we have set P_{max} to 0.008, twice the decay parameter obtained with RB.

The performance benchmarking of the RL agent with respect to RB has been performed as described in Section 4.2.5. For this test, we used circuits of length spanning from 5 to 50, 10 circuits for each length. As the noise level of the qubit is quite low, it is necessary to use circuits with higher lengths to extract the decay parameter for the RB method. Figure 4.21 reports the performance, in reconstructing the noisy DMs, of the trained RL agent compared with RB. The comparison also includes circuits without noise and the maximally mixed state. To better observe the performance differences between the RL and RB noise models, we used a logarithmic scale in the plot reporting the TD. On average, the RL agent outperforms the RB method for all circuit lengths. However, the RL agent performance is comparable within one standard deviation to RB. The high standard deviation obtained in the plot is due both to the limited number of circuits used in the test and the measurement errors that add some uncertainty on the ground truth DMs.

The results obtained in this section differ from those obtained in Section 4.2.5, where the RL agent clearly outperformed the RB method. However, it is important to highlight that the RL agent requires fewer hardware resources than the RB method. In the training of the RL agent, a dataset of 60 circuits with a depth of 10 has been sufficient. The dataset generation process on the employed hardware took just a few minutes, at the cost of some classical resources needed for training the

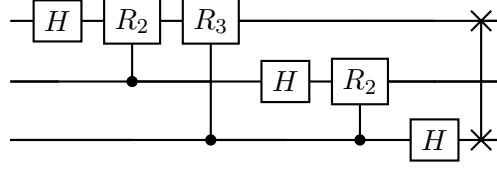


Figure 4.22: QFT circuit for three qubits. In this circuit, $R_k = R_z(2\pi/2^k)$ and H denotes the Hadamard gate. The last SWAP gate can be omitted as this reordering can be handled classically. This circuit implementation has to be transpiled to native gates before being passed to the RL agent.

algorithm. Conversely, to estimate the RB parameter with good precision, it is necessary to run many circuits for each depth.

It is interesting to note that calibration parameters, as well as other external factors that may affect the noise levels of a quantum chip, tend to vary over time. This implies that the RL agent should be retrained regularly on a new dataset to accurately capture the current noise conditions. This issue affects all noise modeling techniques. It is difficult to define a fixed time window during which the RL model, or any noise model, remains valid. This validity period depends on various factors, including the characteristics of the chip, refrigerator, electronics, and even the external conditions of the laboratory. Under stable conditions, the model can be considered valid until a new calibration routine is performed, which typically occurs on a daily basis.

4.2.6 Test on quantum algorithms

This section we assess our model’s performance on QFT and Grover’s algorithm circuits under simulated noise. These tests provide valuable insights into the generalization capabilities of the model and serve as a stress test and benchmark for overall performance.

The QFT, a quantum counterpart of the classical Fast Fourier Transform, is a fundamental component of numerous quantum algorithms, including the renowned Shor’s algorithm for factoring [49]. The QFT acts on a quantum state $|x\rangle$ of n qubits as

$$\text{QFT } |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i x k / 2^n} |k\rangle. \quad (4.1)$$

Being a unitary transformation, the QFT can be implemented as a quantum circuit. For a three-qubit system ($n = 3$), the QFT circuit is reported in Figure 4.22. The layer of SWAP gates included to reorder the qubits is omitted in our implementation, as this reordering can be handled classically if the QFT is the final operation in an algorithm.

Grover’s algorithm, another cornerstone of quantum computing, is renowned for its ability to search unsorted databases with quadratic speedup compared to classical algorithms [50]. The algorithm operates on a superposition of quantum states, and its goal is to find a specific state $|w\rangle$ that satisfies a certain condition defined by an oracle function. The key component of Grover’s algorithm is the Grover iterate, a unitary transformation that contain the information of the oracle. The Grover iterate is typically repeated $\mathcal{O}(\sqrt{N})$ times to maximize the probability of measuring $|w\rangle$, where N is the dimension of the system. We considered a two-qubit system and the target state $|11\rangle$. This configuration requires only one Grover iteration to find the target state. We utilized an ancillary qubit to construct the oracle, leading to the final circuit reported in Figure 4.23.

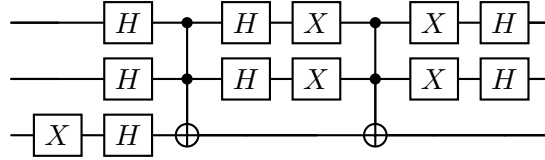


Figure 4.23: Grover’s search algorithm circuit, the target state is $|11\rangle$ and an ancillary qubit is required. In this configuration it is sufficient to interrogate the oracle only one time to obtain the correct result. The circuit has been transpiled to native gates before being used as input for the RL agent.

We transpiled the circuits to utilize the native gates R_z , R_x , and CZ . The gate number after transpilation and other significant circuit parameters are detailed in Table 4.6. A random non-Clifford circuit of the same type used for the training set is also included as a benchmark. It should be noted that the circuits implementing the QFT and Grover algorithms are substantially different from those used in the training set. The main differences include increased circuit depth, a lower fraction of two-qubit gates, and structural variations. Some gate patterns present in these circuits were likely never encountered by the agent during training, as they would be unlikely to arise through random circuit construction. All these factors contribute to making the generalization test performed in this section particularly challenging.

Circuit	Total gates	CZ gates	Depth
QFT	23	6	15
Grover	40	7	25
Random	23	12	15

Table 4.6: Number of gates and circuit depth of the transpiled circuits for QFT and Grover’s algorithm. A random circuit from the training set is also included as a benchmark to highlight the differences including depth and a higher fraction of two-qubit gates.

We have tested the algorithms using both the high and low noise models for three-qubit circuits detailed in Section 4.2.5. This approach served a dual purpose. Primarily, we aimed to test the algorithm’s generalization capabilities with noise models exhibiting lower error rates. Secondly, the noise model from Section 4.2.5 completely masked the final result for Grover’s algorithm. By using a noise model with lower error rates, we ensured that a peak at the target state in the final result remained discernible. We evaluated the performance of our RL agent against the RB noise model and the limit case where no noise is added to algorithm’s circuits. A comparison with the maximally mixed state is also given to give an estimate of the total level of noise. The fidelity between the reconstructed and original noisy DMs for the different noise models is detailed in Table 4.7. We also report the result obtained with the random circuit as a benchmark. In all instances, the RL agent obtained the best performance.

The final state probabilities for both the QFT and Grover’s algorithm circuits, measured in the computational basis, are shown in Figure 4.24. For the QFT circuit we report the result obtained with a high error noise model while for the Grover’s circuit we report the result obtained with a low error noise model that doesn’t destruct the expected result. Also in this case we report the result obtained with the random circuit previously introduced. The histograms compare the probabilities derived from the original noisy circuit, the reconstructed circuit using the RL agent, and the RB noise model. Given the depth of the circuits used, the RB noise model tends to average the output.

Circuit	Noise	RL	RB	No noise added	MMS
QFT	High	0.99	0.97	0.59	0.70
	Low	0.99	0.99	0.78	0.52
Grover	High	0.98	0.95	0.40	0.83
	Low	0.98	0.96	0.65	0.64
Random	High	0.99	0.93	0.61	0.62
	Low	0.99	0.98	0.77	0.51

Table 4.7: Fidelity between the density matrix reconstructed with a noise model (RL agent, RB, the limit case where no noise channels are added and the MMS) and the ground truth noisy one. The result is reported for QFT and Grover’s algorithm circuits for both a high and low noise models. The result obtained with a non-Clifford random circuit is also reported for comparison. The RL agent outperforms, or is comparable, with respect to the RB model in all the studied cases.

In contrast, the RL agent, with a few exceptions, aligns more closely with the probabilities obtained from the original noise model. This alignment is particularly noticeable in the QFT simulation counts for the state $|000\rangle$. This state has the higher probability due to the noise model’s amplitude damping channels, a feature that is successfully replicated by the RL agent but that is not possible to replicate with the RB model.

For a more detailed analysis, Figure 4.25 shows the absolute error between the DMs generated by the RL algorithm and the RB model compared to the ground truth noisy DMs. The errors in the RL model are well distributed across the DM, while the RB model’s errors tend to be higher along the diagonal. This effect was observed in many circuits, even during preliminary tests with the RL algorithm. As the depth of the circuit increases, the density matrix of noisy circuits tends to approach the MMS, resulting in most of the information being contained in the diagonal.

The results obtained in this section underscore the generalization capabilities of the proposed RL approach for noise modeling. The RL agent adapts to circuits with structures distinct from the random ones used in the training set and of significantly different depths. This allows the algorithm to be used for circuit simulations for interesting use cases like quantum algorithms and quantum machine learning when the exact noise model of the hardware chip is unknown.

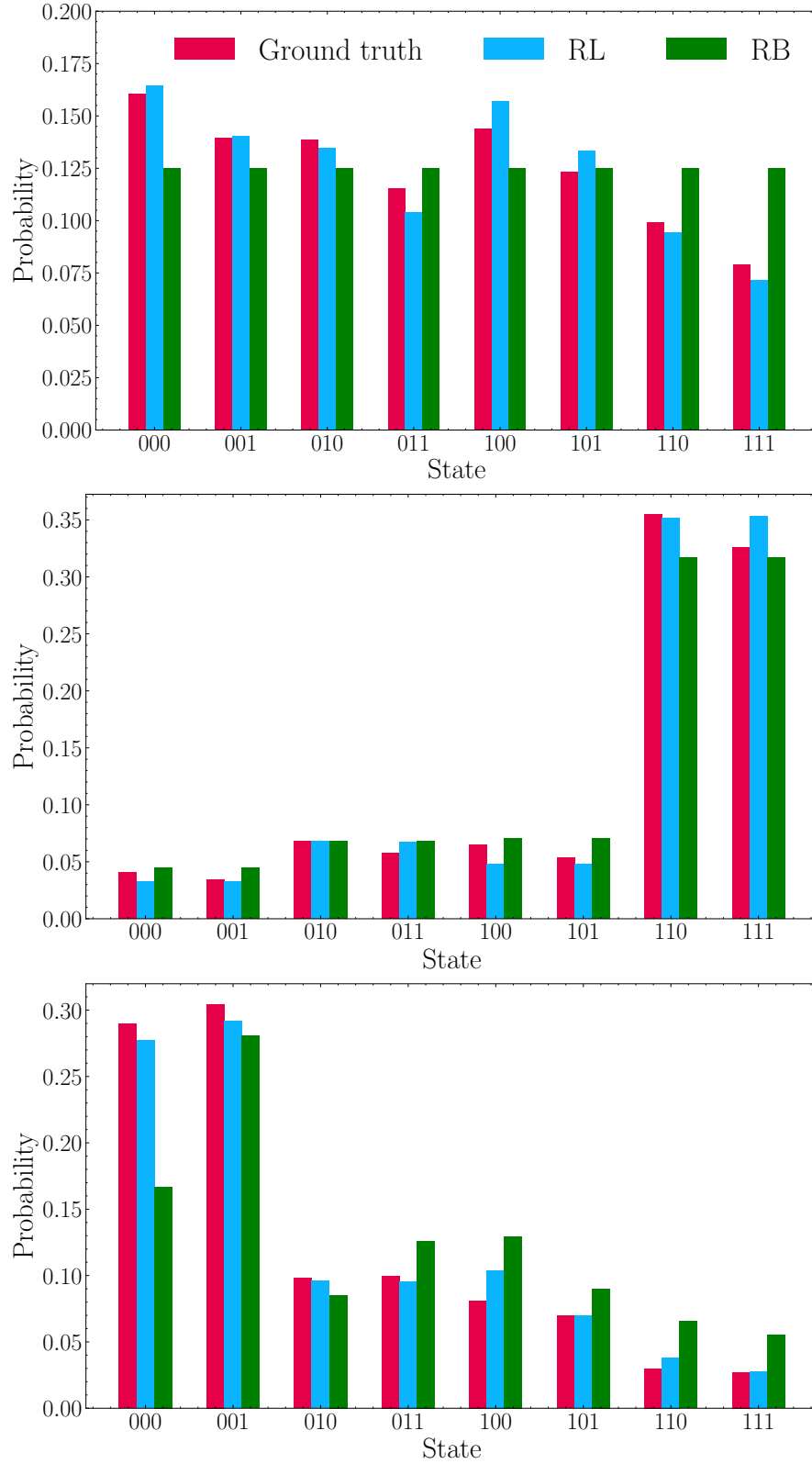


Figure 4.24: Computational basis states probabilities for the QFT circuit with a high error noise model (top), Grover's algorithm circuit with a low error noise model (mid) and a random non-Clifford circuit with a high error noise model (bottom). The histograms show a comparison between probabilities obtained with the ground truth noise model, the RL agent noise model and the RB noise model. It is possible to observe that while the RB model tends to flatten the distribution, the RL agent is able to reconstruct the peaks with more fidelity.

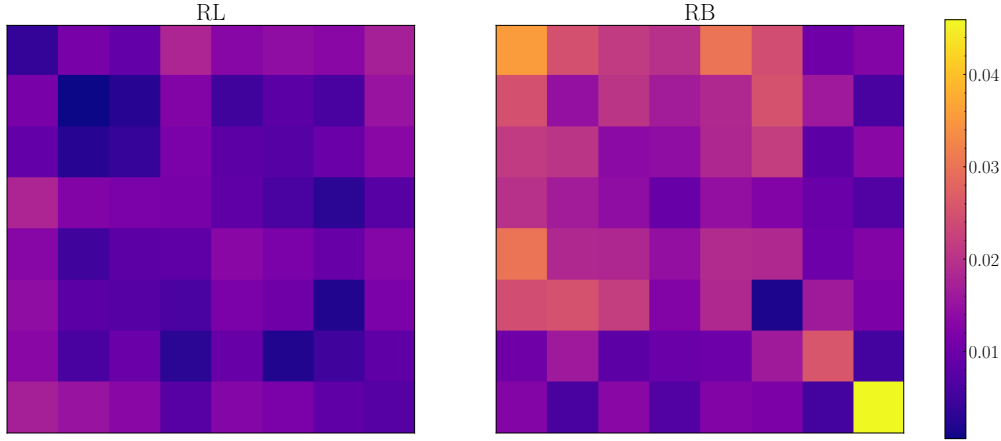


Figure 4.25: Heatmap of the absolute error between the ground truth noisy DM and the DM obtained with the RL agent (left) and RB model (right) for the three-qubit QFT circuit. It is possible to observe that the RL agent is able to better reconstruct the noisy DM not only in the diagonal part. This behavior has been observed in all the circuits tested in this section.

4.2.7 Conclusions

This work presents a reinforcement learning algorithm for replicating specific noise models in single and multi-qubit quantum circuits. This approach reduces heuristic assumptions about the noise model, enhancing generalization properties. Tests on simulated and quantum hardware circuits have demonstrated the model’s ability to learn complex noise patterns and generalize to unseen circuits. For a comprehensive evaluation and to show a possible use case, we tested the model on QFT and Grover’s algorithm circuits. In all occasions, the RL model consistently outperformed a common noise characterization method, randomized benchmarking both in the ability of reconstructing the density matrices and in the amount of quantum hardware resources needed. Possible future applications of the algorithm include not only reproducing the noise pattern of a specific hardware device. By learning the error patterns of qubits for specific gate types, the model could optimize the transpilation process [341], thereby enhancing quantum algorithms fidelity. Furthermore, using the knowledge of the noise for its mitigation could be an interesting approach.

The current model’s limitation is its scalability to circuits with many qubits. There are two main challenges to overcome. The first problem is that scaling the model would necessitate a significant increase in the number of actions. A larger action space would complicate and slow down the training process. The second problem regards obtaining the ground truth density matrices via quantum state tomography as it requires exponentially more measurements in the number of qubits. We are considering potential solutions to these challenges. One approach, to solve the second issue, could involve training the model with probability distributions derived from measurements, rather than density matrices. It is also possible to employ machine learning techniques to reduce the amount of measurements needed for quantum state tomography [342, 343]. To address the first issue, we could partition large circuits into smaller ones, facilitating parallel training of multiple smaller models. Furthermore, it could be beneficial to explore the use of graph neural networks in place of convolutional neural networks to encode qubit connectivity information into the model.

While these ideas require further validation, this work demonstrates that machine learning’s application to learn noise patterns within small quantum circuits is a promising proof of concept that could lead to future advancements.

Chapter 5

Quantum Machine Learning

In this chapter, we present novel results in the field of QML. In particular, we focus on the use of small quantum circuits that can be executed on NISQ devices. Before discussing the results obtained in this research, we introduce the fundamental concept of Parametrized Quantum Circuit (PQC) in Section 5.1. Section 5.2 presents an application of PQC to anomaly detection in High Energy Physics (HEP). This work paves the way for the application of QML algorithms in the analysis of quantum data, which may in the future be produced by quantum sensors. Several novel techniques have been explored to adapt the algorithm for execution on a superconducting chip provided by IBM. This research was published in 2023 under the title *Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits* [41]. The work has been presented at several conferences and workshops, including QC-INFN 2022 in Bologna, QAI 2023 in Naples, and EuCAIFCon 2024 in Amsterdam.

Section 5.3 introduces a novel study on the implementation of a quantum version of a diffusion model. In this project, I collaborated with the Quantum Computing Group at Sapienza University of Rome, primarily contributing to the implementation of a reduced version of the model on quantum hardware. As in the previous case, several modifications were made to ensure compatibility with IBM's superconducting chip. Preliminary results are available in the preprint *Quantum Diffusion Models* [42], while a more detailed version with complete results is currently in preparation. Early findings from this work have been presented at AIQxQIA 2023 in Rome, QC-INFN 2024 in Padova, and EuCAIFCon 2024 in Amsterdam.

5.1 Parametrized quantum circuits

Parametrized quantum circuits can be considered the quantum counterpart of classical neural networks. In this section, we will review their fundamentals, such as ansatz, expressivity, and training procedure. We will also present some drawbacks in the use of these kinds of algorithms, such as the limited expressive power and the difficulty in training due to the presence of barren plateaus. Finally, we will try to give a comparison between QML and classical machine learning, highlighting the possible advantages of the former.

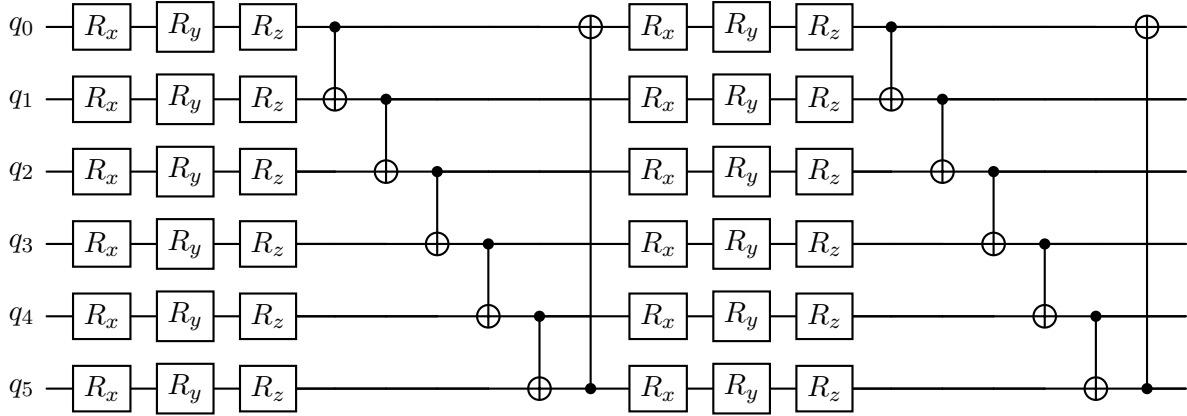


Figure 5.1: Hardware-efficient ansatz for a system of six qubits arranged in a ring topology with CNOT native gates. The circuit is composed of two layers of single-qubit rotations (R_x , R_y , R_z) and CNOT gates.

5.1.1 Ansatz

A PQC is a quantum circuit with a fixed gate structure and a set of gate parameters that can be optimized. The gate structure is called ansatz and defines many important properties of the circuit in a similar way to the architecture of a neural network. In a PQC ansatz, the presence of both entangling gates and parametrized gates is fundamental. The entangling gates must connect all the qubits in the circuit to ensure the possibility of creating entanglement; it is convenient to use fully entangling gates, such as CNOT gates, for this purpose. The parametrized gates have to be differentiable with respect to their parameters; for this reason, they are usually chosen to be single-qubit rotations. It is also possible to use parametrized entangling gates, such as controlled rotations. A PQC ansatz can usually be divided into a series of layers, each one composed of an identical block of gates. This is an analogy with the layers of neurons in a classical NN that can be useful to generate a hierarchical feature representation. The number of layers in a PQC is an important hyperparameter that increases the expressivity but may generate some training difficulty.

Problem specific ansatz

In classical neural networks, such as CNNs and graph neural networks, it is possible to exploit symmetries in the problem to reduce the number of parameters and enhance the performance of the model. In the same way, it is possible to reflect symmetries of the problem in the ansatz of a PQC. For example, in quantum chemistry, the unitary coupled cluster ansatz [344] helps exploit the symmetries of the electronic Hamiltonian for the Variational Quantum Eigensolver (VQE) [345]. In a recent work, D. Materia et al. [346] have shown how to use information obtained from classical chemistry to design molecule-specific ansatz for PQC used in quantum chemistry.

Other examples of variational algorithms worth noting include the Quantum Approximate Optimization Algorithm (QAOA) [347], which is used to approximately solve combinatorial optimization problems. This variational algorithm has a cost operator that is problem-dependent, while the ansatz structure is relatively generic. It is not always possible to exploit symmetries in the problem to construct an efficient ansatz. In this case, it is possible to optimize the ansatz structure with respect to the quantum hardware used to run the circuit. In the case of the hardware-efficient ansatz, the entangling gates are chosen to be the native gates of the quantum hardware. Moreover,

their placement in the circuit should reflect the chip connectivity to avoid the introduction of SWAP gates during the transpilation process. An example of a hardware-efficient ansatz for a system of six qubits arranged in a ring topology is reported in Figure 5.1. This circuit ansatz will also be used in the research presented in Section 5.2 and Section 5.3. Finally, it is worth mentioning the possibility of using an adaptive ansatz, where the structure of the circuit is optimized during the training process. An example of this kind of PQC for quantum chemistry can be found in [348] for quantum chemistry and in [349] for the solution of optimization problems.

Data encoding

A PQC implements a unitary transformation of the initial state. While it is already possible to execute QML circuits with quantum-origin data, the main challenge lies in the efficient encoding of this kind of data. In the future, there is the possibility that QML algorithms will be used to directly process the output of a quantum device, like a quantum sensor. However, at the moment, PQCs are mainly limited to process classical data. The classical data must be encoded in the initial quantum state of the circuit before execution, an operation that often scales poorly with the number of qubits. There are different ways to encode classical data in a quantum state; the most common are amplitude encoding and angle encoding.

In amplitude encoding, the classical feature vector's components are mapped into the coefficients of a quantum state:

$$\mathbf{x} \longrightarrow |x\rangle = \sum_{i=1}^{2^N} x_i |i\rangle,$$

where N is the number of qubits and $|i\rangle$ represents the ordered vector of the computational basis. With amplitude encoding, it is possible to encode 2^N classical features into N qubits. However, the circuit required to perform this encoding grows exponentially in the number of CNOT gates with respect to the number of qubits [350]. This limitation makes amplitude encoding impractical for execution on NISQ devices.

In angle encoding, every classical feature is encoded in the phase of a qubit using a rotation gate. Angle encoding requires a number of single-qubit gates that grows linearly with the number of features and does not require any two-qubit gates. However, this encoding requires a number of qubits equal to the number of features to be encoded, making it impractical for high-dimensional data. Moreover, the initial state generated this way is not entangled.

For specific types of data, it is possible to use type-specific encoding techniques. For example, in the case of images, it is possible to use the Flexible Representation of Quantum Images (FRQI) [351] to encode information about colors in RGB images and corresponding pixel positions.

Finally, when dealing with classical data, it is possible to use the data re-uploading technique [352] to compress the number of qubits required for angle encoding. With this technique, a subset of the input feature vector is encoded at the beginning of every layer of the PQC. Moreover, the input features can be uploaded multiple times during the execution of the circuit to increase the expressivity of the model. However, it is important to note that this technique is not applicable in the case of an initial unknown quantum state.

5.1.2 Training and barren plateaus

The training procedure for a PQC may vary depending if this procedure is performed in simulation, on a classical computer, or on a real quantum device.

In the case of simulation, the training is performed in a similar way as for classical neural networks. The loss is defined as a function of the output state of the PQC. As in simulation the output state is known exactly, it is possible to use gradient-based optimization algorithms to find the circuit parameters that minimize the loss. Differentiability is ensured by the fact that a quantum circuit implements a unitary transformation. For small PQCs, it is also possible to use classical gradient free optimization algorithms, such as simulated annealing or genetic algorithms [353].

Using classical devices to simulate QML algorithms is useful to test new models. However, the requirement of exponential resources scaling the number of qubits makes it impractical for possible real world applications.

When training on a real quantum device we don't have access to the full output state of the circuit. Performing a state tomography to reconstruct the output state at each iteration of the optimization process is not feasible due to the high number of measurements required. The main technique used to train a PQC on a real quantum device is the parameter shift rule [354, 355]. Consider a single parametrized Pauli rotation $R(\theta)$ in a quantum circuit, it is possible to obtain the exact value of the gradient of the loss function $\mathcal{L}(x, \theta)$ with respect to the parameter θ as:

$$\partial_{\theta}\mathcal{L}(x, \theta) = \frac{1}{2} \left(\mathcal{L}(x, \theta + \frac{\pi}{2}) - \mathcal{L}(x, \theta - \frac{\pi}{2}) \right) .$$

The demonstration of this formula can be found in Appendix K. The parameter shift rule can be generalized to a wider class of gates as shown in [355]. In this case the gradient can be evaluated with the following formula:

$$\partial_{\theta}\mathcal{L}(x, \theta) = c (\mathcal{L}(x, \theta + s) - \mathcal{L}(x, \theta - s)) ,$$

where the multiplication factor c and the shift s depend on the specific gate. The main limitation of the parameter shift rule regards the scaling to circuits with a high number of parameters. In fact, with this method, it is necessary to compute the loss function twice for each parameter to evaluate the total gradient. To reduce the computational cost it is possible to use the stochastic parameter shift rule [356]. This method allows to drastically reduce the number of evaluations of the loss function by sampling random shifts. The drawback of this technique is that it introduces a trade-off between the computational cost and the accuracy of the gradient estimation. Another important issue affecting the parameter shift rule is its sensitivity to the presence of noise. To account for this problem some noise-aware techniques have been proposed [357, 358]. These techniques try to mitigate the effect of noise on the gradient estimation by both applying error mitigation techniques and by setting the shift values s and the multiplication factor c to values depending on the noise model.

A problem that affects the training of PQCs, both in simulation and on quantum hardware, is the landscape of the loss function. As shown by McClean et al. [359], loss functions of PQCs tend to have extremely flat regions in the parameter space; this phenomenon is known as barren plateaus.

In analogy with the vanishing gradient problem in classical neural networks, barren plateaus arise in deep quantum circuits with a large number of qubits. Moreover, other significant optimization challenges exist, such as the frequent occurrence of local minima and the narrowness of global minima [360]. To mitigate these challenges, several techniques have been proposed [361]. While no single method has proven to be universally effective, we will give a brief overview of the most relevant ones.

Local loss function: as shown by Cerezo et al. [362], barren plateaus can be mitigated by using a loss function that depends independently on measurements of individual or small subsets of qubits. A loss function of this kind is called a local loss function, in contrast with a global loss function that depends on a simultaneous measurement of all the qubits of the circuit. For example, a local loss function that independently sets each qubit to the ground state

$$\mathcal{L}_{\text{Local}} = \sum_{i=1}^N (1 - p(|0\rangle_i)) , \quad (5.1)$$

will have a smoother landscape compared to a global loss function that tries to set the entire state to the ground state

$$\mathcal{L}_{\text{Global}} = 1 - p(|00\dots 0\rangle) .$$

This technique has been used in the research presented in Section 5.2, where a loss function of the kind shown in Equation 5.1 has been used.

Layerwise training: this technique consists of training the PQC layer by layer in order to reduce the depth of the trainable region of the circuit [363]. After the layerwise training, it is suggested to perform a final training of the whole circuit with a small learning rate. The layerwise training can be seen as an initialization technique that helps find a region of the parameter space close to a minimum of the loss function. It is interesting to note that layerwise training was first proposed in the context of classical neural networks as a way to mitigate the vanishing gradient problem [220].

Smart initialization: the initial values of the parameters of the PQC have an important effect in avoiding flat regions of the loss function. Initialization techniques that have been proposed in the context of classical neural networks, such as Xavier initialization, can be adapted to the quantum case [364] and have shown better performance with respect to random initialization. Another noteworthy initialization technique, which shares similarities with layerwise training, has been proposed by Grant et al. [365]. This method consists of initializing the parameters of some layers of the PQC so that they implement the identity transformation.

Ansatz design: as for classical neural networks, the architecture is important in avoiding the vanishing gradient problem, the ansatz of a PQC can help mitigate barren plateaus. In particular, symmetries or constraints that reflect the structure of the problem help reduce the number of free parameters [366, 367], thus reducing the actual depth of the circuit. An interesting recent research, published by Holmes et al. [368], shows a correlation between the expressivity of the ansatz and the presence of barren plateaus. Finally, the work of Marrero et al. [369] shows that the problem of barren plateaus is magnified by high levels of entanglement

in the circuit. Their research suggests fine-tuning the number of entangling gates in the layers of the PQC.

5.1.3 Comparison between quantum and classical machine learning

In the previous sections, we have observed many similarities between classical and quantum machine learning. In particular, PQCs can be used to solve almost all the ML tasks that are addressed by NNs. The question that naturally arises is whether there are any advantages in using QML algorithms compared to the classical ones. Before going into the details of the answer to this question, it is important to note that, at the moment, no QML algorithm has ever shown a clear advantage over its classical counterpart. In the current era, the study of QML algorithms has mainly academic interests and is not aimed at finding practical applications. There are two main reasons for this situation: the first is the lack of reliable quantum hardware, and the second is that QML is still in its infancy compared to classical ML so that the quantum counterparts are developed only after their classical analogues. It is unlikely that QML will be able to outperform classical ML in the near future, characterized by NISQ devices. Some recent research outlines that the basic requirement for observing potential advantages is the availability of quantum hardware with over 100-200 qubits [370–372]. Moreover, a recent study by Schuster et al. shows that any quantum circuit for which error mitigation is efficient must be classically simulable [297], thus adding the requirement of error correction to achieve a possible quantum advantage. The derivative nature of QML is evident from the fact that its quantum counterparts of ML models are typically developed only after their classical analogues, as emphasized by P. Wang et al. [373]. This creates a situation where QML is always behind classical ML. Taking inspiration from classical models may help in fast development but carries the risk of not exploiting the full potential of quantum computing. As shown in the work of H. Huang et al. [374], you do not automatically get advantage by just replacing classical model components with quantum ones; the structure of the problem and access to truly quantum data or processes matter. Moreover, a recent study by P. Bermejo et al. [375] shows that the quantum counterparts of CNNs, known as quantum convolutional neural networks, when randomly initialized and tested on low entangled data, are classically simulable. Hopefully, with the advent of large-scale quantum computers, it will be possible to develop new QML models that are not inspired by classical ones and that will fully exploit the quantum nature of the hardware.

Even with all the limitations that have been discussed, the study of QML may still lead to interesting results that highlight possible future advantages of these kinds of algorithms over classical ML. Some recent research has shown that QML algorithms can be trained with a smaller number of samples than classical ML algorithms [376, 377]. These results suggest possible advantages in the use of QML algorithms in the presence of limited data. Quantum computers show a speedup in some subroutines useful in finding the solution of optimization problems, such as the Harrow-Hassidim-Lloyd (HHL) algorithm [378]. QML algorithms that exploit this speedup, like the quantum K-means [379, 380], may be more efficient than their classical counterparts [36, 38, 381]. However, it is important to note that this class of QML algorithms is not based on PQCs. A similar framework, showing interesting results, is the development of hybrid quantum-classical ML algorithms [382]. In these algorithms both NNs and PQCs cooperate in solving a complex task. For example, a classical autoencoder can be used to reduce the dimensionality of the input data, while a PQC can

process the information in the latent space. Another possible advantage of QML algorithms deals with the intrinsic stochastic nature of quantum computers and their ability to manipulate complex probability distributions [383]. This property is particularly useful for the development of quantum generative models [384–386], an example of which is reported in Section 5.3. Finally, it is worth observing that, at the moment, QML algorithms are used on classical data. An interesting study by L. Schatzki et al. [387] argues that the majority of QML works are tested on classical datasets (encoded into quantum states) but that to fully reveal quantum advantage one should use quantum datasets. They propose *NTangled*, a dataset of quantum states with varying entanglement, for supervised learning classification tasks. However, this dataset still requires amplitude encoding to be loaded into a quantum computer. As the encoding procedure represents a big bottleneck, the possibility of processing quantum data directly from quantum instruments would be a game changer in the field of QML. The exponential growth of the quantum state space would make it impossible to use classical ML algorithms to process high-dimensional quantum states, while QML algorithms could directly analyze this kind of data. For example, quantum anomaly detection algorithms may be able to analyze the output of a quantum sensor as depicted in Section 5.2.

In conclusion, trying to predict the future of QML is a difficult task. The research in this field is focused on understanding potential advantages while awaiting the development of quantum hardware for practical applications.

5.2 Quantum anomaly detection for high energy physics

The background useful to understand the research presented in this section is introduced in Section 3.2.3 (anomaly detection) and Section 5.1 (parametrized quantum circuits).

5.2.1 Abstract

We investigate the possibility of applying quantum machine learning techniques for data analysis, with particular regard to an interesting use-case in high energy physics. We propose an anomaly detection algorithm based on a parametrized quantum circuit. This algorithm has been trained on a classical computer and tested with simulations as well as on real quantum hardware. Tests on NISQ devices have been performed with IBM quantum computers. For the execution on quantum hardware, specific hardware-driven adaptations have been devised and implemented. The quantum anomaly detection algorithm is able to detect simple anomalies like different characters in handwritten digits as well as more complex structures like anomalous patterns in the particle detectors produced by the decay products of long-lived particles produced at a collider experiment. For the high energy physics application, performance is estimated in simulation only, as the quantum circuit is not simple enough to be executed on the available quantum hardware. This work demonstrates that it is possible to perform anomaly detection with quantum algorithms. However, as amplitude encoding of classical data is required for the task, due to the noise level in the available quantum hardware, the current implementation cannot outperform classic anomaly detection algorithms based on deep neural networks.

5.2.2 Introduction

The classical anomaly detection algorithm is implemented with an artificial neural network, in particular an autoencoder architecture [224–226]. In quantum machine learning, the autoencoder is realized using parametrized quantum circuits.

A quantum circuit implements a unitary, thus invertible, transformation on the initial state. This represents a great advantage for the autoencoder architecture, as the decoder can be taken as simply the inverse of the encoder quantum circuit, as shown in Figure 5.2. In order to compress information, the encoder circuit has to disentangle and set a given number of qubits to the zero state [388]. The loss function is thus taken as the expected measurement values of these qubits. In this way, for the training of the circuit, only the encoder is necessary. For the simulation and training of the PQC, we have used the QIBO [309] library, which can be easily integrated with TensorFlow [171] for automatic differentiation.

First, we test a simpler version of the quantum algorithm in an easier task involving a standard benchmark dataset in machine learning, the handwritten digits MNIST dataset. We then apply the technique to a more complex and interesting use-case, the identification of anomalous signatures inside a particle detector due to the decay of long-lived particles with macroscopic lifetimes. The application of quantum machine learning to high energy physics is an interesting field that has been studied using QML simulators in some recent works [389–399]. In this paper, we present the first application of QML to the task of anomaly detection for long-lived particle identification, and also prove that the proposed variational quantum circuits could be used on actual quantum hardware. The parametrized quantum circuits developed in this work are, in fact, simple enough to be tested

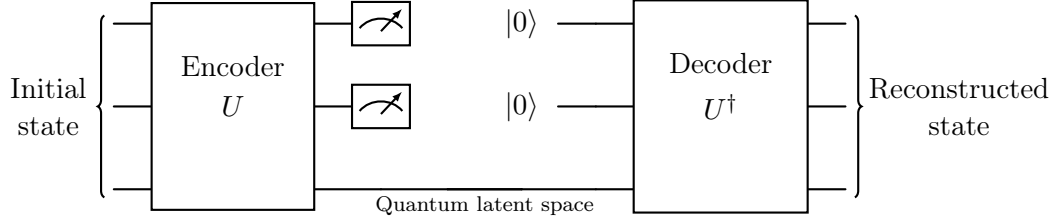


Figure 5.2: Schematic representation of a quantum autoencoder with three qubits. The encoder acts as a unitary transformation U that tries to disentangle two qubits and force them to the $|0\rangle$ state. In this way, the initial quantum information is compressed into a latent quantum space. The decoder implements the inverse transformation $U^\dagger$ to reconstruct the original data. The loss function is taken as the measurement expectation value of the qubits that are eliminated after the compression.

on NISQ computers [51, 400–402]. The tests on real quantum hardware have been implemented on IBM quantum computers [403]. We stress here that the proposed algorithm is not meant, at this stage, to outperform the classical counterpart on classical data, but only to show that it is possible to use quantum variational algorithms for anomaly detection. QML algorithms will be employed in real applications only when less noisy qubits or effective error correction procedures are available in digital quantum computers.

The proposed algorithm may show future advantages for the analysis of quantum data [372]. In fact, it is really difficult to manage quantum data with classical hardware, due to the information size growing exponentially with respect to the number of qubits. On the other hand, a QML algorithm can directly analyze quantum data, overcoming the problem of amplitude encoding (Section 5.2.4).

5.2.3 Simulation on classic hardware

In order to find the best parameters for the proposed anomaly detection algorithm, the first tests have been carried out with simulations on classical hardware. The anomaly detection task can be solved with satisfactory discriminative power by requiring a PQC with enough expressive power. This can be achieved by increasing the depth of the circuit. However, increasing the depth of the circuit may lead to difficulties in finding a suitable minimum in the loss function during the training, mainly due to the presence of barren plateaus [359, 362]. To reach an acceptable trade-off between the two effects, a detailed study of the different possible quantum gate topologies that could be used in the circuit was carried out. The algorithm has been applied at first for the recognition of anomalous handwritten digits (Section 5.2.3), and then for the high-energy physics problem (Section 5.2.3). In both cases, the quantum encoding of the classical input data has been implemented using amplitude encoding. In this way, it is possible to encode a number of features that grows exponentially with the number of qubits in the quantum circuit. A drawback with amplitude encoding, on the other hand, is in the state preparation that requires an exponential number of gates (see Section 5.1 for reference) with respect to the number of qubits. State preparation is a current and still open problem on NISQ devices.

Simple use-case: handwritten digits

Anomaly detection on handwritten digits has been carried out on the MNIST dataset. We define handwritten “zero” digits as normal data, and “one” digits as anomalous data. Examples of two im-

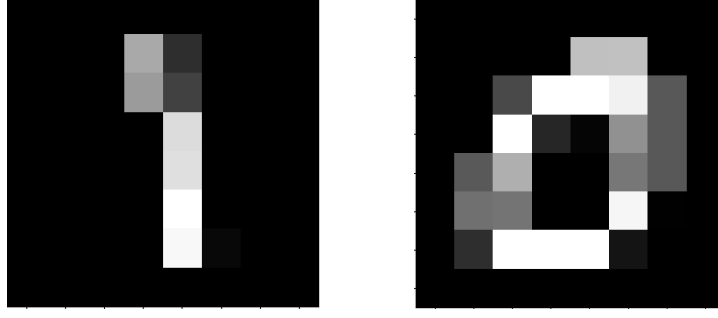


Figure 5.3: Example of images for the “zero” and “one” digits for the MNIST handwritten digits dataset. Images are compressed to 8×8 pixels.

ages from the MNIST dataset are shown in Figure 5.3. The original MNIST images are compressed down to 8×8 pixels, with each pixel being an integer number from 0 to 255 (8-bit grayscale). In order to encode the classical data, the images are flattened to obtain feature vectors composed of 64 elements. These vectors are then normalized so that they can be encoded in the state amplitudes of 6 qubits using amplitude encoding.

We tested the PQC with different numbers of layers and compressed qubits. A suitable number of layers has been empirically found to be in the interval between 4 and 8. In fact, circuits with less than 4 layers don’t have enough expressive power to solve the task, causing an abrupt drop in performance. On the other hand, we empirically observed that, when increasing the number of layers to more than 8 the training procedure becomes slow and tends to get stuck in local minima, a typical symptom of barren plateaus. The number of compressed qubits has been varied from 1 to 4. In this range, the decoder is able to reconstruct the original images with good precision. The performances of the different circuit architectures have been compared using the AUC value (see Section 3.2.3 for reference). The best configuration has been found with six layers and three compressed qubits. This procedure has been repeated in order to find the best entangling gates ansatz. We have tested different C-NOT configurations; the one that produced the best performance is reported in Figure 5.4. In order to improve the performance, rotation gates with trainable parameters were added at the end of the encoder circuit for the three compressed qubits. It is worth noting here that the chosen configuration requires only nearest neighbor connectivity for six qubits placed in a ring

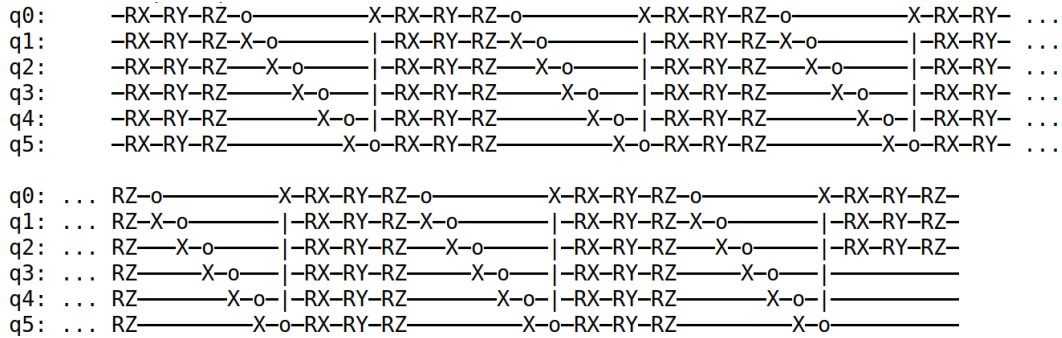


Figure 5.4: Circuit representation of the quantum encoder used for anomaly detection of handwritten digits. The encoder circuit is composed of six qubits and six layers. At the end of the quantum circuit the first three qubits are measured in order to compress information on the three remaining qubits. The loss function is computed as the sum of the probabilities of having any of the measured qubits in the ground state.

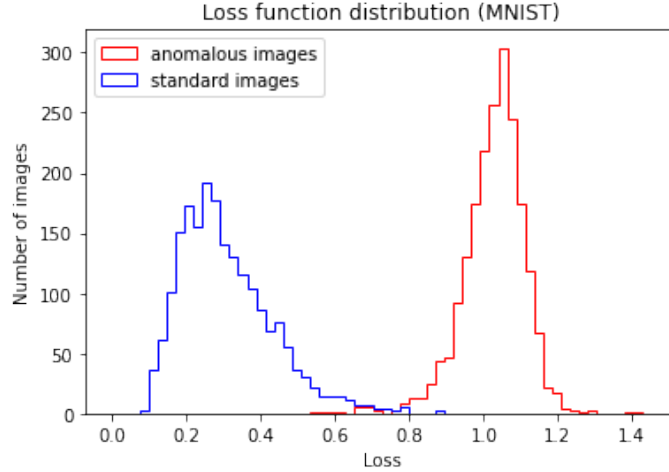


Figure 5.5: Quantum autoencoder loss function values distribution. The graph has been made using 2000 normal data images (zeros) and 2000 anomalous data images (ones) from the MNIST dataset.

topology. We also verified that it is possible to preserve good performance by reducing the number of layers from six to four. We leverage these two properties during the implementation on quantum hardware of the PQC as reported in Section 5.2.4, to minimize the disruption of the algorithm’s performance due to the noisy device.

For the training of the circuit, a dataset of 5000 images of zero handwritten digits has been employed. As the first three qubits are forced to the $|1\rangle$ state in order to apply the data compression, the loss function is the sum of the probabilities of having any of these three qubits in the ground state. Training has been performed for 20 epochs using the Adam optimizer (see Appendix H for reference), with a dynamic learning rate that spans from 0.4 in the first epochs to 0.001 in the last ones. This variable learning rate has helped reduce the problem of barren plateaus. The number of epochs is sufficient to reach a plateau in the loss function. No overfitting has been observed during the training process, thus no early stopping has been required. For the training, we have used batches composed of 20 samples (250 steps per epoch). To test the anomaly detection algorithm after the training phase, we have used 2000 normal data images not used in the training and 2000 anomalous data images. Figure 5.5 shows the loss distribution for the two test datasets. It is possible to observe a clear separation, with an average loss value for normal data of 0.307 and 1.026 for anomalous data, with an average Root Mean Square (RMS) of the two distributions of 0.110 and 0.066, respectively.

High energy physics use case

An interesting use-case for a high energy physics application of an anomaly detection algorithm is the identification of anomalous patterns in the trigger system of a collider experiment. As a benchmark scenario for this work, we investigated the specific case of long-lived exotic particles predicted in new physics models with a secluded sector, like in [404, 405]. New particles predicted by theories beyond the Standard Model can generically have lifetimes that are long compared to Standard Model particles at the weak scale. When produced at the LHC, such long-lived particles can decay far from the primary interaction vertex and possibly themselves interact with the detector material, leading to a plethora of possible detector signatures. Such signatures are distinctly different from

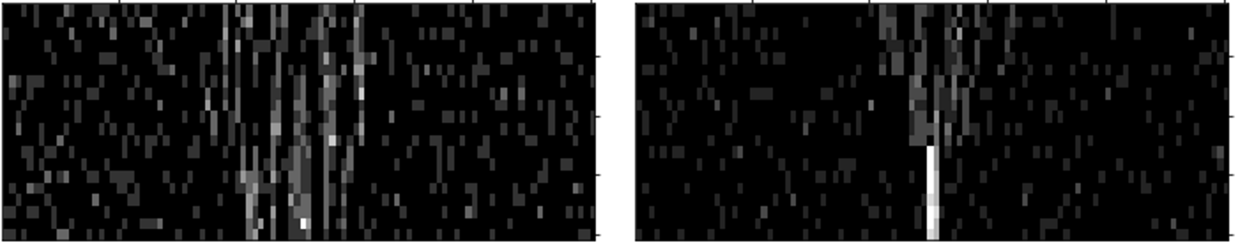


Figure 5.6: A normal (left) and an anomalous (right) event used for quantum anomaly detection applied to long-lived particles detection. Normal and anomalous data are respectively prompt and displaced decays in multi-muons (from two to ten muons). Hits produced by the muons are reconstructed in a toy simulation of the ATLAS Muon Drift Tube (MDT) chambers, that includes random hit background noise mimicking expected ATLAS phase-2 noise. Data are represented in the form of 100×20 pixels images.

those associated with traditional searches for prompt particles, and require dedicated reconstruction and identification algorithms, especially in the trigger systems of the collider detectors. As an example case, it is possible to think about the high-level muon trigger system of the ATLAS experiment [406], where the computational demands and the lack of flexibility of traditional secondary vertex reconstruction algorithms in the muon spectrometer [407], restrict their applicability only to the offline reconstruction [408]. This prevents their use in the trigger system of the experiment, limiting the discovery potential of the experiment for this kind of new physics searches. The ATLAS experiment muon system aims to collect all the particle hit information from different sub-detectors to find muon candidates in a given sector (i.e., a solid angle region of the detector). In this proof of concept study, we restrict our attention only to the barrel region of the muon spectrometer and to the MDT precision detector. The trigger algorithm tries to find patterns of hits consistent with the presence of muons originating from the same production vertex. One can think to arrange the MDT detector hits into image-like objects, to be used as input for ML algorithms particularly suitable to find patterns like the muon tracks in this test scenario. Using the published ATLAS detector geometry and resolution as well as its magnetic field map [406], it is possible to generate toy events with muon tracks, from the decays of a non-interacting neutral long-lived particle at different decay lengths from the primary proton-proton interaction vertex.

The images are produced initially with a pixel resolution that roughly corresponds to the MDT detector segmentation, each horizontal pixel corresponding to the center of an MDT tube, and each vertical pixel corresponding to one of the layers of MDT tubes in the sector. The simplified simulation is only parametric, associating a binary value to each pixel (zero or one) depending on whether one of the muons from the long-lived particle decay went through the tube, and then applying position smearing based on the published ATLAS detector performance. Random hit background emulating the typical noise rate conditions during the LHC runs has also been included in the simulation. The background noise is generated only accounting for the average hit rate in the spectrometer MDTs, therefore it does not consider correlated backgrounds. We did not aim here to perfectly reproduce the experimental conditions, but to give a proof of principle of the anomaly detection algorithm in the context of a HEP experiment.

The images obtained in the simplified simulation are too large (20×333 pixels) to be encoded as quantum input in the available quantum computers, therefore they have been compressed with a pooling operation to reduce their size to 20×100 pixels. The pixels of the compressed image contain

the integral of the original pixel content pooled and therefore the image appears as a grayscale and no longer as a binary image. Events are generated with a single particle gun generator producing the decay products of a hypothetical dark particle promptly decaying into multi-muons in the primary vertex of the detector, and then translated at different decay lengths. The momentum distribution of the particle mimics the typical expected one for a dark-photon in the Falkowski–Ruderman–Volansky–Zupan model [404], the mass of the long-lived particle has been randomly chosen in the range $[0.5, 5]$ GeV. Two datasets are generated, one corresponding to prompt to short decay length decays into multi-muons (from two to ten muons), with radial decay lengths uniformly distributed between 0.0 and 20.0 cm from the primary interaction vertex, and one corresponding to very displaced decays into multi-muons, with radial decay lengths uniformly distributed between 250.0 and 450.0 cm. Data are conveniently represented in the form of images of dimension 100×20 pixels, a sample is reported in Figure 5.6.

The encoding procedure used for this dataset is the same as the one described for the MNIST dataset in Section 5.2.3. In this case, 11 qubits are required for amplitude encoding. In order to find the best PQC ansatz, the same procedure described in Section 5.2.3 has been repeated. For this task, we have tested PQCs with a number of layers varying from 6 to 10 and a number of compressed qubits varying from 1 to 4. Choosing a number of layers within this interval represents a good trade-off between the expressive power and the problem of barren plateaus. The best performance (AUC value) has been obtained with 8 layers and 3 qubits compression. Training has been performed on a dataset of 8000 normal data images for 60 epochs. The number of epochs is sufficient to reach a plateau in the loss function. Also in this case, no overfitting has been observed during the training process, thus no early stopping has been required. For the training, we have used batches composed of 20 samples (400 steps per epoch). The optimizer and learning rate are the same as those used in Section 5.2.3. In order to mitigate barren plateaus, we have also tried to train the circuit layer by layer as suggested in [363], but the final result was worse than a single all-layers training.

In order to benchmark the performance of the quantum algorithm, we have made a comparison with a classic anomaly detection algorithm. The standard autoencoder has been implemented with CNN. The encoder is composed of three convolution layers followed by a dense layer and compresses the input features down to a latent space of dimension 5. The decoder is composed of a dense layer followed by three transpose convolution layers that restore the original input dimension. The total number of parameters of this neural network is about 7.9×10^3 . The loss function is the binary cross-entropy between the input and the reconstructed image. Training has been performed on the same dataset used in the quantum case, for 60 epochs and using the Adam optimizer. The final performance evaluation has been carried out on 2000 normal data images and 2000 images of anomalous decays for both the quantum and classic algorithms.

The loss function distributions for the normal and anomalous datasets are reported in Figure 5.7. It is possible to observe a separation in the loss function distribution for anomalous and normal data images in both cases. For the quantum algorithm, the resulting average loss for normal events is 0.870 while the average loss for anomalous events is 0.788. The RMS of the two distributions are respectively 0.038 and 0.030. For the classic algorithm, the resulting average loss for normal events is 0.409 while the average loss for anomalous events is 0.355. The RMS of both distributions

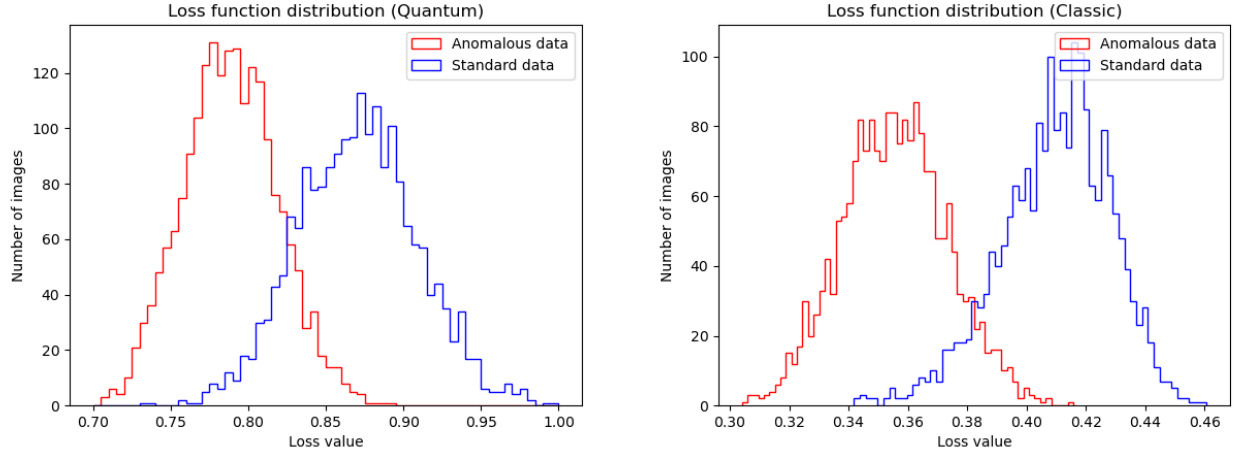


Figure 5.7: Loss function values distribution for the quantum anomaly detection algorithm (left) and the classic counterpart (right). The graphs have been made using 2000 normal data images (decays that happen just as the particle enters the detector) and 2000 anomalous data images (decays that happen after the particle has travelled inside the detector).

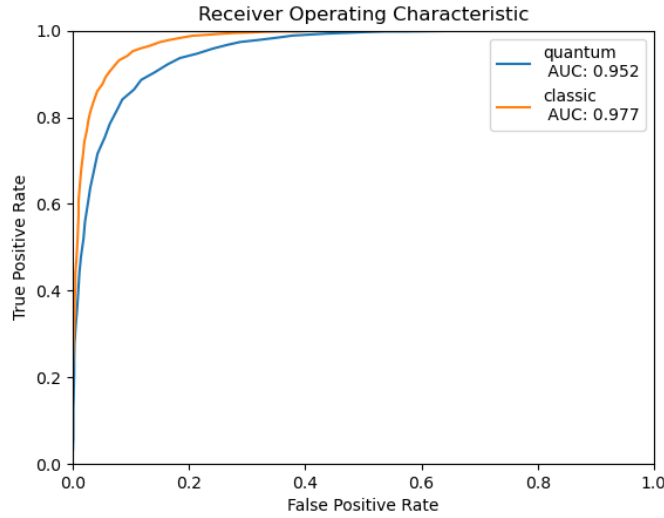


Figure 5.8: ROC curve and AUC for the quantum anomaly detection algorithm (blue) and the classical counterpart (orange).

is 0.017. Note that as the loss functions are different for the classic and quantum cases, only the relative separations can be compared.

It is worth noting here that, for both algorithms, the anomalous data show a lower average loss than normal data. This is due to the fact that anomalous images have a simpler structure with hits distributed in a narrow cone with respect to the normal images, and thus are easier to compress. For a better benchmark, Figure 5.8 shows the ROC curve and AUC comparisons between the classic autoencoder and the quantum model. The quantum algorithm does not reach the same level of performance as the classical equivalent. This is due to the reduced expressive power of the implemented quantum circuit, imposed by the constraints on the number of usable qubits and gates for the simulation. In any case, the quantum algorithm is already very close to the classical one, and is expected to significantly improve, on quantum hardware implementation, with the availability of

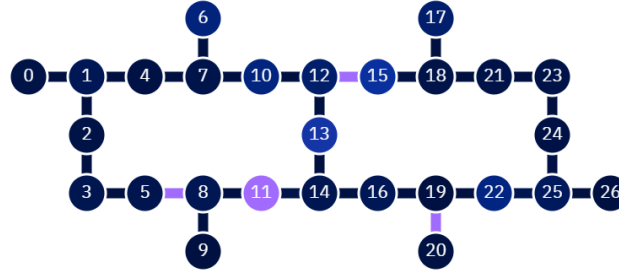


Figure 5.9: Architecture of IBM_hanoi quantum computer. The C-NOT connectivity is reported. The colours of single qubits and their connections represent respectively the single qubit readout assignment error and C-NOT error probabilities. Darker colours represent a lower error probability, in a range between 5.9×10^{-3} and 9.8×10^{-2} for readout error and 3.3×10^{-3} and 1 for C-NOT gates. Data from calibration on 19/10/2022.

the new generation of low noise quantum circuits which has been announced to be available soon¹.

5.2.4 Test on quantum hardware

The execution of quantum circuits on NISQ devices is difficult even on state-of-the-art quantum devices. The main problems come from the high error rate of quantum gates, especially C-NOT gates that are fundamental to generate entanglement between qubits. Another important limitation comes from the connectivity in the architecture of quantum computers. In fact, it is not possible to apply C-NOT gates between all possible pairs of qubits. If an interaction between non-connected qubits is required, it is necessary to use SWAP gates to swap the quantum states of two qubits. As each SWAP gate can be decomposed into three C-NOT gates, the use of noisy gates is further increased. State-of-the-art quantum computers offer connectivity only between neighboring qubits, arranged in linear or circular structures. The architecture of the quantum computer used in this work (IBM_hanoi) is reported in Figure 5.9.

Adaptation to quantum hardware

With the limitations described, it is impossible to achieve any significant performance on the IBM quantum hardware for the anomaly detection circuit we have developed. In order to make the algorithm work, some changes and careful adaptations had to be implemented to reduce the complexity and size of the PQC. Given the consequent reduction in the expressive power of the model, we decided to focus only on the simplest use-case of the handwritten digits for the quantum hardware test. We describe here the changes to the quantum circuit in order to solve the two problems discussed in the previous section, the amplitude encoding and the C-NOT connectivity.

For the connectivity problem, it is important to notice that the PQC proposed in Section 5.1 requires only neighboring qubits interaction if the qubits are arranged in a circular topology. However, in our quantum hardware (Figure 5.9) it is not possible to find six qubits arranged in a ring. By removing the last C-NOT gate on each entangling layer, only neighboring qubits interactions are required for qubits arranged in a line. This allows us to remove all SWAP gates from the circuit, thus significantly reducing the total number of C-NOT gates.

Moreover, we decided to use only four layers for the encoder circuit. With these changes, the ex-

¹See for example the IBM Roadmap to quantum advantage: <https://www.ibm.com/quantum/technology>, 2025.

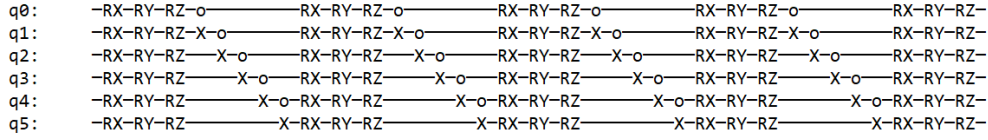


Figure 5.10: Parametrized quantum circuit used for approximated amplitude encoding. The circuit is composed of four layers made of rotation gates and five C-NOT gates plus a final layer composed only of rotation gates.

pressive power of the PQC is reduced but still sufficient to detect anomalies. In noisy simulations, the performance of this PQC outperformed the one with 6 layers employed in Section 5.2.3.

Amplitude encoding is a state preparation procedure that allows transforming the standard initial state (all qubits in the ground state) into a state that encodes the input data for the quantum algorithm. This procedure is necessary only to analyze classical data with quantum hardware. Implementing amplitude encoding on quantum hardware requires a number of C-NOT gates that grows exponentially with the number of qubits. For a circuit of 6 qubits, more than one hundred C-NOT gates are required, which adds too much noise to the final result. To overcome this problem, we developed another PQC designed to provide a good approximation of the exact amplitude encoding while using a reduced number of gates. This circuit is trained to transform the initial ground state into a state that approximates amplitude encoding. The parameters of the circuit have been chosen by minimizing the mean squared error between the output state of the circuit and the target state. For this procedure, one circuit has to be trained for each normal or anomalous data that we want to encode. The PQC ansatz is composed of four layers with the C-NOT gate disposition described before for the encoder as reported in Figure 5.10. Moreover, a final layer composed only of rotation gates has been added to increase expressive power and improve performance. Each training has been carried out for 15 epochs, with each epoch composed of 100 training steps, using the Adam optimizer and a learning rate of 0.01. In order to avoid barren plateaus, we started each training using the parameters of the previous PQC. Looking at the final loss values, we found that the encoding of anomalous data (1 digits) is easier than the encoding of normal data (0 digits). In order to avoid possible bias introduced by bad amplitude encoding, we selected only the data encoded with a loss smaller than 0.1. We have trained approximated amplitude encoding with these characteristics for about 200 normal and 200 anomalous data.

Results

The final test on quantum hardware has been carried out using a circuit composed of two parts, the approximated amplitude encoding circuit and the encoder. The parameters of the encoder have been trained with simulations on classic hardware in the same way as described in Section 5.2.3. The parameters of the approximated amplitude encoding circuit have been trained on each dataset as described in the previous section. 200 samples of normal data and 200 samples of anomalous data have been tested on the quantum hardware. Each circuit has been executed with 2048 shots. Figure 5.11 reports a comparison of the measurement counts of the circuit (only the first three qubits are measured) for a normal sample obtained in three different cases. One plot reports the simulated circuit without noise, the second plot reports the simulated circuit with noise. For the

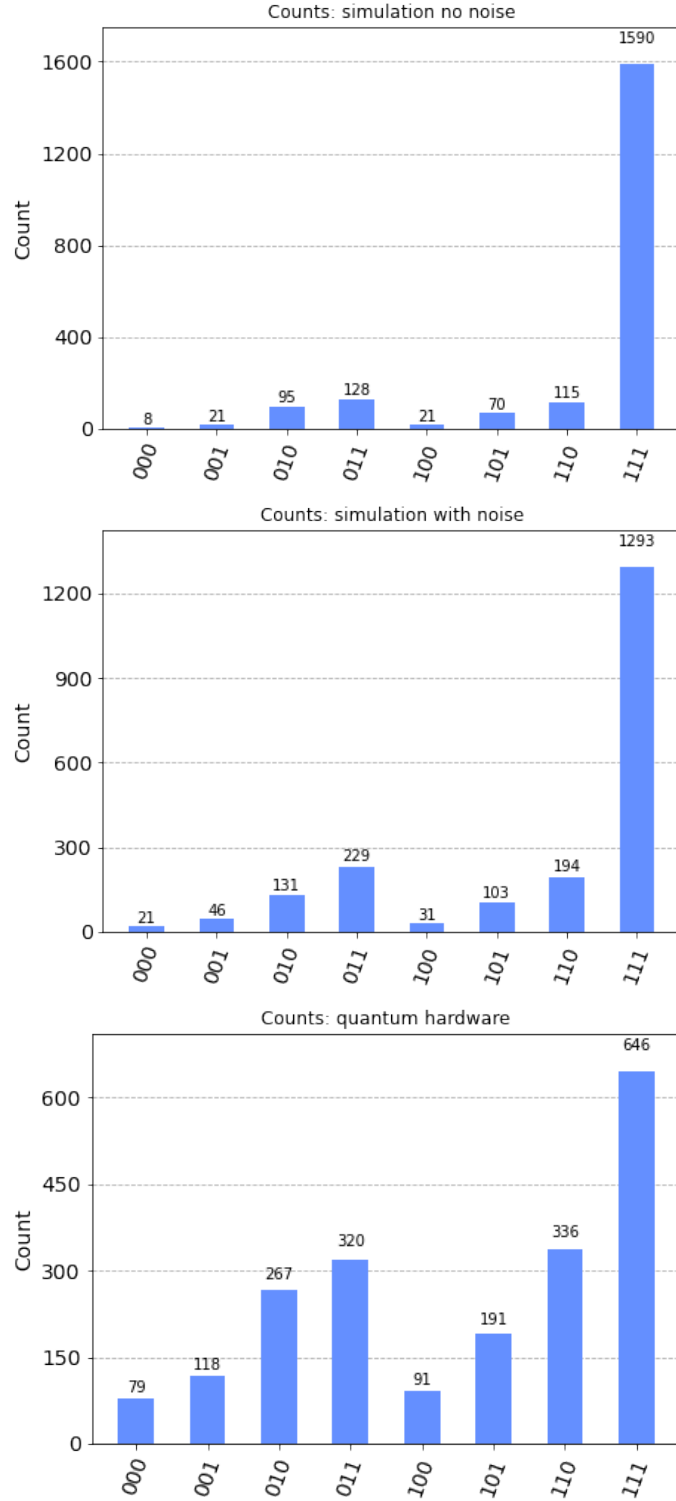


Figure 5.11: Counts distribution of 2048 shots for a simulated circuit with no noise (top) a simulated circuit with realistic noise (center) and a noisy quantum circuit (bottom).

noise model we have used the calibration values of IBM_hanoi taken on 19/10/2022². The last plot reports the counts for the circuit executed on the quantum hardware.

As explained in Section 5.2.3, the loss function, which the circuit has to minimize, measures the sum of the probabilities of the three compressed qubits to be in the ground state. Thus, if the algorithm is working correctly, we expect the measured qubits to be mainly in the $|1\rangle$ state. The counts distribution is peaked, as expected, on the $|111\rangle$ state in all cases. The real quantum circuit, however, clearly shows a higher level of noise shown by higher counts for states different from $|111\rangle$. Moreover, the noise reported by IBM's simulation is lower than the one on real quantum hardware. This is probably due to the difficulty in reproducing a realistic noise model. In a real quantum circuit there are many sources of noise, besides the readout measurement error and the gate error, so it is complicated to keep track of them all. In particular, the built-in noise model offered by Qiskit [174] only accounts for gate errors (modelled as depolarizing channels), readout errors, and relaxation and dephasing times modelled with thermal relaxation channels. Moreover, the noise parameters, obtained from calibration, change over time. This makes them no longer reliable if the circuit is not executed right after the calibration, like in this case.

The noise makes the loss function distributions for normal and anomalous data almost indistinguishable. However, by observing the counts, we noticed that it is possible to improve the performance of anomaly detection by using as a loss function one minus the probability of the $|111\rangle$ state. This loss function is different from the one previously employed. In fact, it can be easily observed that: $1 - p(|111\rangle) \neq p(|0\rangle_1) + p(|0\rangle_2) + p(|0\rangle_3)$. This new loss function gives better results in the presence of noise because the state $|111\rangle$ is the one with the highest output probability. Output state probabilities are computed as the number of counts, for that specific state, divided by the total number of counts. Thus, probabilities of states with higher counts are less affected by the noise, compared to the probabilities of states with a lower number of counts.

The distributions for normal and anomalous data for this loss function are reported in Figure 5.12 for simulated circuits with no noise (top left), simulated circuits with noise (top right), and real quantum circuits (bottom). For a better comparison, Figure 5.13 reports the ROC curves and the AUC for the three cases. It is possible to observe a significant separation between normal and anomalous data in the presence of noise, although with a clear degradation in the case of the execution on real quantum hardware (hardware vs simulation AUC: 0.896 vs 0.983).

It is interesting to notice that in the case of simulation, the ROC curves for simulated circuits with or without noise almost overlap even if the loss values for these two cases have different distributions. A low level of noise just shifts the losses to higher values but does not reduce the discriminative performance of the algorithm.

²The calibration can be accessed through IBM's Qiskit library [174]: <https://www.ibm.com/quantum/qiskit>, 2025.

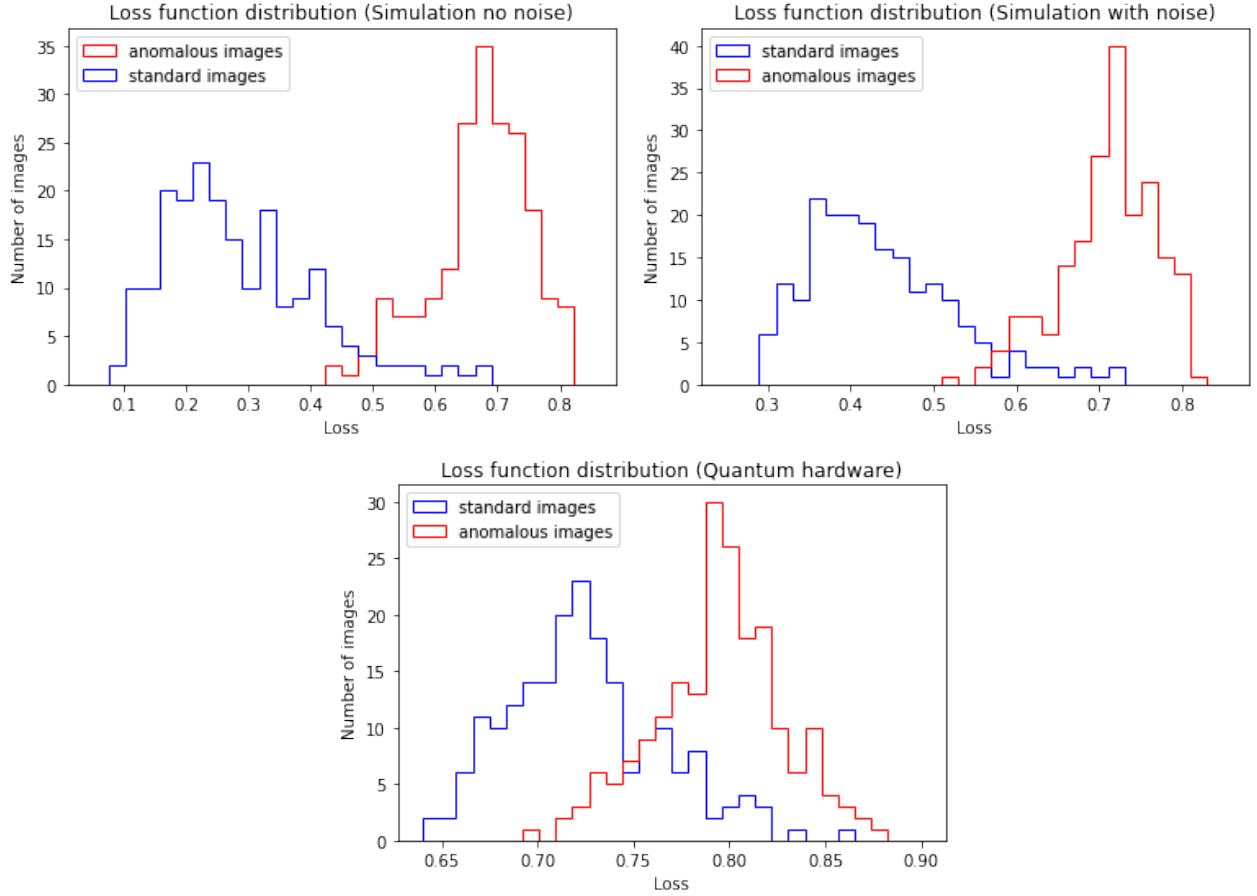


Figure 5.12: Quantum autoencoder loss function values distribution. Simulated circuits with no noise (top left), simulated circuits with noise (top right) and a noisy quantum circuits (bottom). The histogram has been realized using 200 normal data and 200 anomalous data, with 2048 shots each circuit. The loss function is one minus the probability of the $|111\rangle$ state.

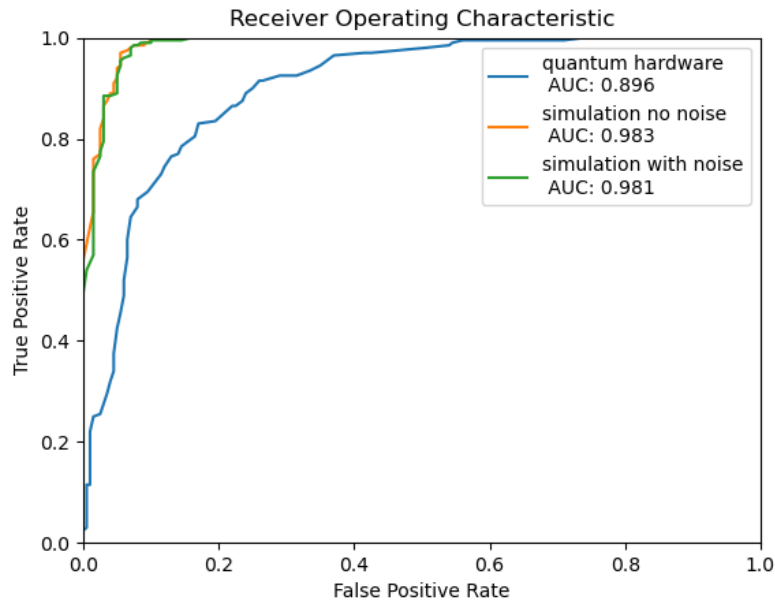


Figure 5.13: ROC curves and AUC for anomaly detection, simulated circuits with no noise (orange), simulated circuits with noise (green) and noisy quantum circuits (blue).

5.2.5 Conclusions

Quantum machine learning is a newborn topic with many possible algorithms still to be explored. In this work, we proved that it is possible to perform anomaly detection for long-lived particle searches in a HEP experiment using parametrized quantum circuits. Theoretically, without considering noise limitations, PQCs are powerful enough to distinguish anomalies in complex patterns like the ones obtained in a muon spectrometer of a hadron collider experiment. With the currently available NISQ devices, it is only possible to execute very simple circuits on quantum hardware. These circuits must be adapted to the hardware limitations and can be used only for simpler tasks like anomaly detection of handwritten digits. However, as quantum computers are improving rapidly [409, 410], we expect, in the near future, to use PQCs to solve complex tasks in particle physics. At that time, it will be possible to verify if, with fault-tolerant quantum computation, quantum machine learning will outperform classical machine learning algorithms in the analysis of classical data.

Another possible direction to explore is the possibility of feeding the quantum algorithms directly with quantum input data, thus avoiding the quantum data encoding step. Since the research and development of quantum sensing in particle detectors is a rapidly developing sector [411–413], this could lead to identifying a possible advantage from the use of quantum machine learning algorithms, of the type proposed in this work, in the not too distant future.

5.3 Quantum diffusion models

The background useful to understand the research presented in this section is introduced in Section 3.2.4 (diffusion models) and Section 5.1 (parametrized quantum circuits).

5.3.1 Abstract

Denoising diffusion probabilistic models have become increasingly important in machine learning, but their quantum counterpart has not been studied yet. In this work, we propose a quantum version of a generative diffusion model where artificial neural networks are replaced with parameterized quantum circuits. This algorithm can sample from quantum ensembles without necessitating classical-quantum data conversion. We evaluate the model’s performance on classical data by incorporating a measurement layer. We present a method for conditioning the output distribution using ancillary qubits. Given the current limitations of quantum hardware and qubit simulation capabilities, we employ a pre-trained classical autoencoder to generate complex distributions. All the models presented in this work are evaluated using quantitative metrics and qualitative assessments and compared with state-of-the-art classical diffusion models. Finally, we have tested a simplified version of our algorithm on a real NISQ device to assess the model’s performance in the presence of noise.

5.3.2 Introduction

The ability to reproduce probability distributions from a dataset and to sample from them is one of the most useful tasks of ML models; these are referred to as generative models. In this context, diffusion models are algorithms inspired by nonequilibrium thermodynamics [238, 414–416] that offer a promising alternative to the established VAEs and GANs (see Section 3.2.4 for reference), due to their superior sample quality and variability.

Parallel to these advancements in ML, the field of quantum computing has experienced rapid development in recent years. However, the current generation of quantum processors is too noisy for the implementation of quantum supremacy algorithms. On the other hand, QML algorithms are less sensitive to noisy devices [40]. Even if these algorithms have not proven to outperform their classical counterparts, many interesting results have been obtained both in simulations and on NISQ devices [302, 304–306, 417–419]. The first meaningful utilization of quantum computing may likely come from algorithms of this kind, complemented by error mitigation techniques.

The quantum versions of some established generative algorithms have already been developed, like the quantum GAN [398, 420] and the quantum VAE [421]. Initial attempts to develop a quantum version of a diffusion model have focused on simple distributions [422] or the learning of small quantum state ensembles [423]. In this work, we introduce and evaluate a Quantum Diffusion Model (QDM) capable of sampling from multi-qubit quantum state distributions by employing parameterized quantum circuits. Moreover, it is possible to employ a measurement layer to convert quantum states into classical data. The ability to sample quantum states from a well-defined distribution finds applications in many different fields [43] such as quantum cryptography [44], quantum metrology [45], fidelity benchmarking [46], and theoretical physics [47, 48]. Quantum states with defined characteristics can be generated using quantum circuits for classical-to-quantum encoding; however, this procedure is not scalable to a large number of qubits [424]. In this context,

a quantum diffusion model may be a scalable alternative for the generation of quantum states randomly extracted from a desired distribution.

Due to the limitations of current quantum hardware, generating high-dimensional data with quantum models is not feasible. This is essential for comparing these models with state-of-the-art classical diffusion models. To enable this comparison, we measured the generated quantum states, converting them into classical vectors. These vectors are derived from a lower-dimensional distribution, obtained by employing a pre-trained autoencoder on the original dataset. The quantum version of a diffusion model is not intended to outperform state-of-the-art classical generative models for the generation of classical data. However, a QDM, when used for the generation of classical data, requires a smaller number of trainable parameters with respect to its classical counterpart.

5.3.3 Model design

The quantum adaptation of a diffusion model works through direct manipulation of quantum states. This is accomplished by substituting the traditional NN with a PQC. Due to the intrinsic differences between these two models, several adjustments are necessary to obtain a working algorithm.

Algorithm adaptation

In a classical diffusion model, a forward Markov chain is constructed by adding noise through a Markov kernel. At the same time, an NN is trained to approximate the Gaussian kernel mean. This procedure is not possible in a quantum model, as it would involve decoding and encoding quantum states at each time step to perform classical operations on the vectors. This is impractical as current NISQ devices are subject to measurement errors and the encoding of classical data is inefficient [425]. For this reason, we have decided to utilize a hybrid approach, implementing the forward Markov chain classically, while utilizing a PQC in the backward process and the sampling phase. The PQC defines an operator $\hat{P}(\boldsymbol{\theta}, t)$ that performs data denoising directly:

$$\hat{P}(\boldsymbol{\theta}, t) |\mathbf{x}_t\rangle = |\mathbf{x}_{t-1}\rangle ,$$

where $|\mathbf{x}_t\rangle$ is the quantum state obtained by encoding the classical vector $\mathbf{x}_t$, which represents the t step in the forward chain. This way, the complete backward process becomes a denoising process, which can be performed by the consecutive application of T different PQCs, as detailed in Algorithm 2.

We observed that training a single circuit for all denoising steps results in only a limited performance reduction. Another important consideration is that, in the iterative application of the PQC, the coefficients of the quantum states can become complex, even if the initial state $\mathbf{x}_0$ had

Algorithm 2 QDM sampling algorithm.

```

Sample  $\mathbf{x}_T \sim \pi(\mathbf{x}_T)$ 
Quantum encoding:
   $\mathbf{x}_T \longrightarrow |\mathbf{x}_T\rangle$ 
for  $t = T, \dots, 1$  do
   $|\mathbf{x}_{t-1}\rangle = \hat{P}(\boldsymbol{\theta}, t) |\mathbf{x}_t\rangle$ 
end for
```

Algorithm 3 QDM training algorithm.

```

repeat
   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
   $t \sim \text{Uniform}(\{1, \dots, T\})$ 
   $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2 \sim \mathcal{CN}(\mathbf{0}, 2\mathbf{I})$ 
   $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_1$ 
   $\mathbf{x}_{t+1} = \sqrt{1 - \beta_t} \mathbf{x}_t + \sqrt{\beta_t} \boldsymbol{\epsilon}_2$ 

  Quantum encoding:
   $\mathbf{x}_t \longrightarrow |\mathbf{x}_t\rangle$ 
   $\mathbf{x}_{t+1} \longrightarrow |\mathbf{x}_{t+1}\rangle$ 

  Take gradient descent step on:
   $\nabla_{\boldsymbol{\theta}} [1 - \mathbb{E}[F(\hat{P}(\boldsymbol{\theta}, t) |\mathbf{x}_{t+1}\rangle, |\mathbf{x}_t\rangle)]]$ 

until converged

```

real values. For this reason, we have tested an alternative to the classical forward kernel where we used complex noise instead of real noise at every time step of the training:

$$\begin{cases} \boldsymbol{\epsilon} = \boldsymbol{\epsilon}_r + i\boldsymbol{\epsilon}_i \\ \boldsymbol{\epsilon}_r, \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \end{cases}$$

This approach led to a significant improvement in performance.

The PQC has been trained using a completely classical optimizer. However, during the algorithm's design, we have chosen a loss function that could be easily computed using a full quantum approach, namely the infidelity loss:

$$L(\boldsymbol{\theta}) = 1 - \mathbb{E}[F(\hat{P}(\boldsymbol{\theta}, t) |\mathbf{x}_{t+1}\rangle, |\mathbf{x}_t\rangle)],$$

where F indicates the quantum fidelity. The expected value is taken over all possible choices of $\mathbf{x}_t$ at every value of t and $\mathbf{x}_0$. On quantum hardware, the fidelity can be computed efficiently with the use of a multi-qubit swap test [426]. The procedure for training the quantum model is summarized in Algorithm 3.

Circuit architecture

We have tested several architectures for the PQCs. The two circuits leading to the best performance are reported in Figure 5.14. Both architectures use three unitary blocks operating on different qubits. Every block is composed of different layers organized as the ansatz reported in Figure 5.1

In the first architecture (bottleneck PQC), after a first unitary block $\hat{U}_n^{(1)}(\boldsymbol{\theta})$ acting on all qubits, the first m qubits are measured, reducing the number of qubits to $n - m$. After the application of the second unitary block $\hat{U}_{n-m}^{(2)}(\boldsymbol{\theta})$, acting as a bottleneck, m ancillary qubits are introduced in the state $|0\rangle^{\otimes m}$, and a final block $\hat{U}_n^{(3)}(\boldsymbol{\theta})$ is applied to all n qubits.

In the second architecture (reverse-bottleneck PQC), the order of the ancillary qubit introduction and the measurements is reversed. This way, the central block $\hat{U}_{n+m}^{(2)}(\boldsymbol{\theta})$ acts on $n + m$ qubits. We observed little difference in the performance of the two architectures, with the reverse-bottleneck PQC performing slightly better. We have found that the best value for the number of measured

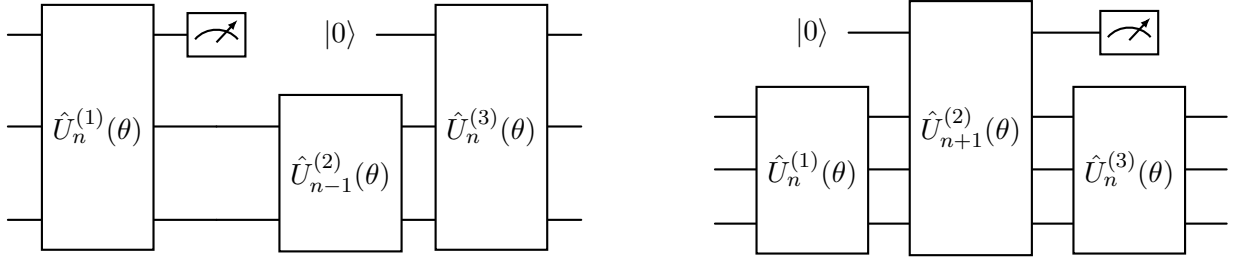


Figure 5.14: Representation of the bottleneck (left) and reverse-bottleneck (right) architectures with the measurement of one qubit. Each unitary transformation block in the image uses the layered structure reported in Figure 5.1.

qubits is $m = 1$. For these reasons, every result in the following refers to the reverse-bottleneck architecture, using one measured qubit.

Testing different PQC architectures, we have observed that the number of intermediate measurements of ancillary qubits has a strong impact on the performance of the algorithm. The introduction of intermediate measurements significantly improves the quality of generated samples. This can be explained by the fact that the measurement process acts as a non-linear map over the states' amplitudes [427]. Our numerical simulations have confirmed the non-linearity of these maps when using our ansatzes with trained parameters.

Another remarkable finding is that measurements reduce the samples' variability. In particular, when their overall number exceeds a certain threshold (dependent on the architecture and number of qubits), no sensible variability is observed in the generated samples, resulting in a model collapse. One potential explanation for this phenomenon is that, every time a qubit is measured, it is reset to a fixed state. This process erases part of the information contained in the initial noise [428]. Consequently, by increasing the number of measurements, the initial information is gradually lost, resulting in a loss of variance in the output states.

Measurements of the ancillary qubits can be performed with two methods, namely with or without branch selection [427]. With branch selection, we indicate the process of measuring the ancillary qubits and accepting the remainder of the state only if the outcome coincides with a predetermined choice. If the state is rejected, the circuit is executed again from the beginning. By measurement without branch selection, we indicate the process of accepting the remainder of the quantum state regardless of the measurement outcome.

We have empirically observed that utilizing measurements with branch selection leads to better results. However, when performing branch selection on quantum hardware, each measurement has an average success rate of 0.5. Hence, this process is not suitable for deep circuits, as the probability of accepting the circuit execution diminishes exponentially with the number of measurements. For this reason, throughout this work, we have used measurements without branch selection for the ancillary qubits. Another important remark is that, when performing measurements without branch selection, the output state is nondeterministic. This characteristic aligns our model more closely with the classical diffusion model.

Model variations

We present two independent variations on the algorithm described so far.

Latent models: To increase the expressive power of the QDM, it is possible to use a hybrid approach by employing a pre-trained classical autoencoder [429]. With this approach, the QDM is trained on a low-dimensional representation of data. The lower dimensionality of the problem, along with the strong non-linearity introduced by the classical autoencoder, has improved the quality of samples and allowed the use of smaller PQC. Moreover, the consequent simplification of the PQC has allowed the implementation of the algorithm on real quantum hardware, as explained in Section 5.3.5.

Conditioned models: Both the full quantum model and the latent model can be easily conditioned by increasing the dimension of the Hilbert space. This is done by taking the tensor product between the input quantum state and an additional state that encodes the label (label qubits):

$$|\mathbf{x}_t\rangle \longrightarrow |k\rangle \otimes |\mathbf{x}_t\rangle ,$$

where $|k\rangle$ is the k^{th} state of the computational basis. In order to encode N labels, it is necessary to add $\lceil \log_2(N) \rceil$ qubits. We emphasize that, as for the ancillary qubits, it is possible to measure label qubits with or without branch selection. We have chosen to use measurements without branch selection in our hardware implementation, and measurements with branch selection in our simulations.

Model evaluation

To assess the performance of the proposed models, we used a combination of qualitative evaluation and quantitative metrics. To visually compare the distributions of the original and generated samples and enhance our evaluation of image quality, we used dimensionality reduction techniques such as Principal Component Analysis (PCA) [430] and t-SNE [431]. For the quantitative assessment, we used the AUC [432] to evaluate the conditioning performance, while for the generation performance, we used both the Fréchet Inception Distance (FID) [433] and the 2-Wasserstein distance for Gaussian mixture models (WaM) [434].

To evaluate conditioned models, the ROC-AUC is computed using 10 binary classifiers, each specialized in distinguishing a single digit within the MNIST dataset. In this context, we assume the classifiers to be perfect, with each one achieving an accuracy well above 0.99 on the original dataset. This assumption enables the trained classifiers to serve as a reliable ground truth against which the conditioned samples can be evaluated. For the FashionMNIST dataset, we haven't used this method due to the lower classification accuracy, which does not suffice as a reliable ground truth. To evaluate the generation performance, we used both the FID and the WaM. We chose this second metric as the FID suffers major issues, as pointed out in [433], particularly with small and grayscale images. For this metric, we extracted image features utilizing the intermediate layers of two pre-trained classifiers, specialized on the MNIST and FashionMNIST datasets. To calculate both metrics, for the conditioned and unconditioned latent model, we used a total of $7 \cdot 10^4$ samples, 10^4 at a time to evaluate statistics. For the fully quantum model, we used a total of 10^4 samples, 10^3 at a time.

Table 5.1: PQC characteristics of different models.

Feature	Full quantum (MNIST)	Unconditioned latent (MNIST)	Conditioned latent (MNIST)	Unconditioned latent (F.MNIST)	Conditioned latent (F.MNIST)
Layers	100	50	50	80	80
Qubits	8	3	3+4	3	3+4
Parameters	2520	510	1110	810	1770
Denoising steps	15	8	8	8	8
Images size	16x16	28x28	28x28	28x28	28x28
Latent space	N.A.	8	8	8	8
Labels	“0”/“1”	All	All	All	All

5.3.4 Simulation results

In this section, we present the results obtained in simulation for the full quantum model without conditioning, and for the latent model with and without conditioning. We have omitted the results obtained from the conditioned full quantum model, as conditioning it on only two labels did not yield significant differences in the results. The training datasets used in this work are the MNIST and FashionMNIST datasets. For the full quantum model, we have used 2048 samples from the MNIST dataset, downsized from 28x28 pixels to 16x16 pixels. These can be encoded into 8 qubits using amplitude encoding. Moreover, only zeros and ones have been used to further simplify the problem. For the latent models, we have chosen to use latent vectors of dimension 8, encoded using 3 qubits. In this case, we used 8192 samples and all ten labels for both the MNIST and FashionMNIST datasets. The conditioned version of the latent model required 4 additional qubits to encode the 10 labels, for a total of 7 qubits.

We have tested different numbers of layers for the unitary blocks of the reverse-bottleneck circuit ansatz (Figure 5.14). For the latent models, as the PQC acts on a smaller Hilbert space, it is possible to obtain optimal performance with a total number of 50 layers (distributed as 15+20+15) for MNIST and 80 layers (distributed as 25+30+25) for FashionMNIST. The full quantum model requires a total number of 100 layers (distributed as 30+40+30) to reach sufficient expressive power. Even though these circuits are composed of a high number of layers, we have not experienced any issues related to barren plateaus [435] during training. A summary of the characteristics of the models is reported in Table 5.1.

Full quantum model

Samples generated by the full quantum model are shown in Figure 5.15. The characteristics of the model used for the generation of these samples are reported in the first column of Table 5.1. We highlight how the number of time steps T is significantly smaller with respect to classical diffusion models. Two examples of the denoising process are reported in Figure 5.16. The samples show that the model is not capable of generating high-quality and diverse samples. In particular, many samples appear to be combinations of “zero” and “one” digits not clearly belonging to either of the two classes. This is likely due to the typical linearity of quantum circuits. The FID and WaM values between the original and generated datasets can be found in the first column of Table 5.2.

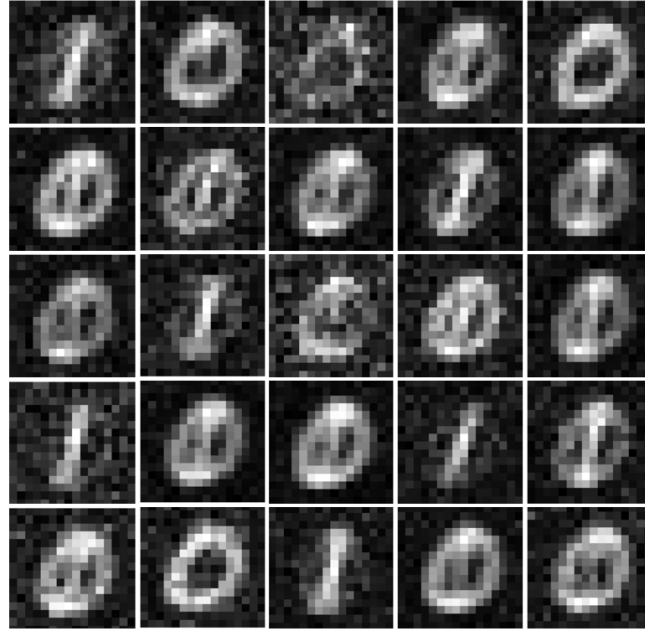


Figure 5.15: Full quantum model samples.

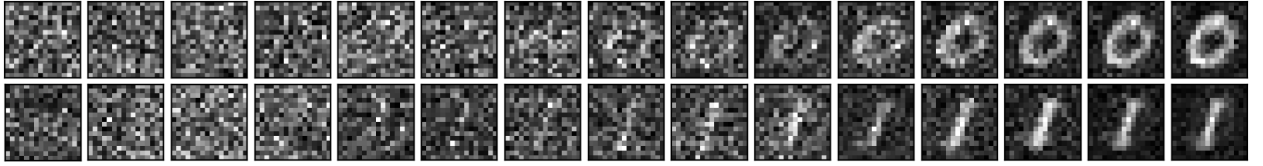


Figure 5.16: Detail on the denoising process of a “zero” and a “one” digit for the full quantum model.

Table 5.2: Evaluation metrics for QDMs on different datasets.

	Full quantum (MNIST)	Latent unconditioned (MNIST)	Latent unconditioned (F.MNIST)	Latent conditioned (MNIST)	Latent conditioned (F.MNIST)
FID	256.6 ± 2.0	41.4 ± 0.3	50.5 ± 0.5	38.2 ± 2.7	57.9 ± 0.3
WaM	120.2 ± 3.2	59.1 ± 5.7	283 ± 19	105.2 ± 1.3	267.3 ± 7.5

Table 5.3: Evaluation metrics for classical diffusion models.

	Unconditioned (MNIST)	Unconditioned (F.MNIST)	Conditioned (MNIST)	Conditioned (F.MNIST)
FID	2.98 ± 0.11	8.50 ± 0.18	2.270 ± 0.054	5.40 ± 0.11
WaM	27.0 ± 1.1	259 ± 11	18 ± 11	166 ± 33

Table 5.4: ROC-AUC values for each digit obtained with the latent conditioned model. Lowest and highest values are highlighted.

Digit	0	1	2	3	4	5	6	7	8	9
AUC	0.951	0.977	0.983	0.989	0.857	0.899	0.975	0.928	0.908	0.933

Unconditioned latent model

Samples generated with the unconditioned latent model are shown in Figure 5.17 and 5.18 for the MNIST and F.MNIST datasets respectively. Each image has been obtained by sampling from the QDM and decoding each sample with a pre-trained decoder. The characteristics of the model used to generate these samples are reported in the second column of Table 5.1. The generated images shows a better quality with respect to the full quantum model, showing a higher number of details and more variability. The non-linearities introduced by the classical autoencoder allows the model to remove the combinations of the digits observed in the full quantum model.

The FID and WaM values between the original and generated datasets can be found in the second and third columns of Table 5.2, referring to the downscaled MNIST and FashionMNIST respectively. For comparison, we report the values of FID and WaM obtained using a classical state-of-the-art denoising diffusion probabilistic model, in the first and second columns of Table 5.3.

Conditioned latent model

Samples generated with the conditioned latent model are shown in Figure 5.19 and 5.20. Conditioning the model generally improves the quality of the samples. This improvement can be attributed to the fact that conditioning increases the differentiation of samples with different labels. Moreover, adding four qubits increases the dimension of the Hilbert space of the circuit and the number of trainable parameters, thus giving more expressive power to the PQC. The characteristics of the model used to generate these samples are reported in the fourth and fifth columns of Table 5.1, referring to the downscaled MNIST and FashionMNIST respectively. A comparison between the original and generated latent data points for the downscaled MNIST dataset is reported in Figure 5.21, where we have used t-SNE to generate a 2D representation.

To assess conditioning, we have used the AUC as detailed in Section 5.3.3. The area under the curve (AUC) values for the ROC curves, specific to each digit, are presented in Table 5.4. The digits “one” and “zero” exhibit the highest AUC values, which can be attributed to their simplicity and minimal overlap with other digits. The FID and WaM values between the original and generated datasets can be found in the fourth and fifth columns of Table 5.2, referring to the downscaled MNIST and FashionMNIST respectively. These quantities are computed between the overall distributions, hence not considering the separation of labels. For comparison, we report the values of FID and WaM obtained using a classical state-of-the-art denoising diffusion probabilistic model, in the third and fourth columns of Table 5.3.

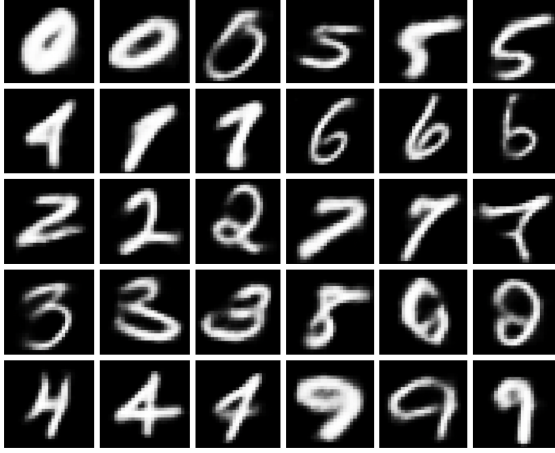


Figure 5.17: Unconditioned latent model (MNIST).

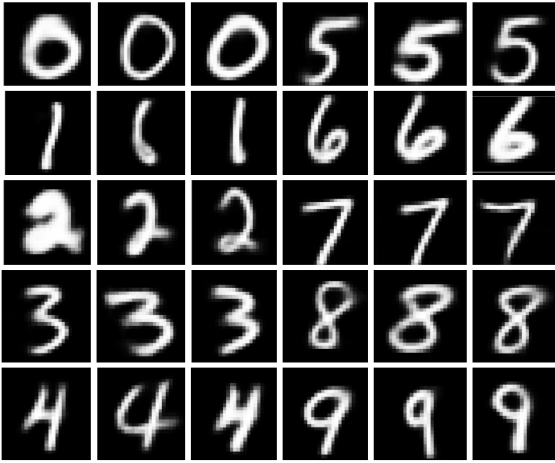


Figure 5.19: Conditioned latent model (MNIST).

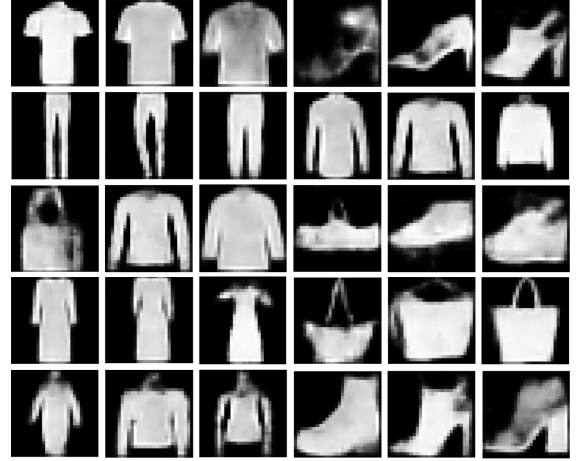


Figure 5.18: Unconditioned latent model (F.MNIST).

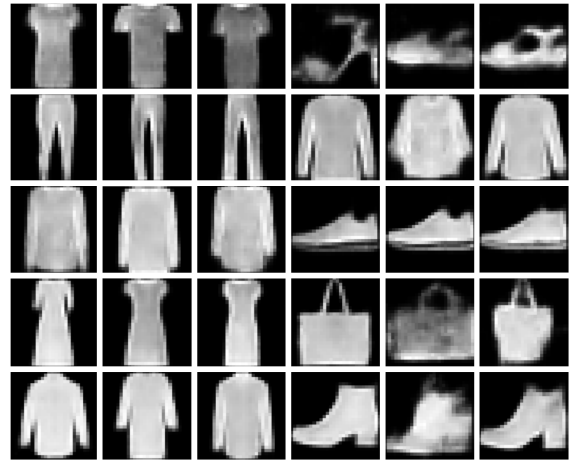


Figure 5.20: Conditioned latent model (F.MNIST).

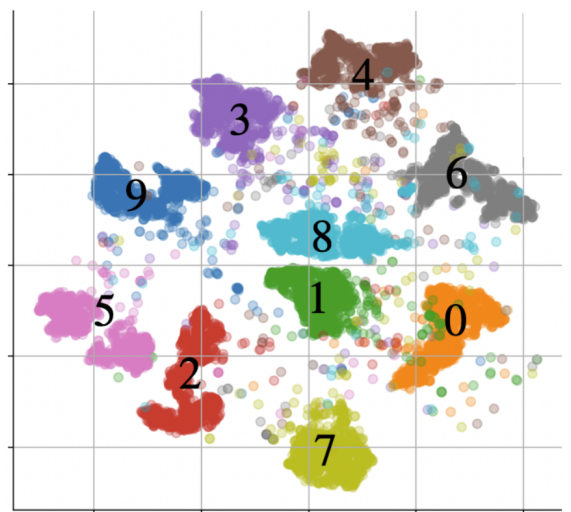
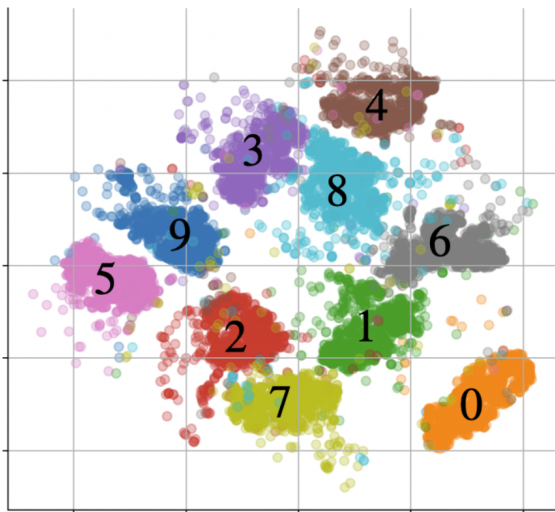


Figure 5.21: 2D representation (t-SNE) of the original distribution (left) and generated distribution (right) in the latent space (8D). Each digit is generated separately by conditioning the model. The same t-SNE has been simultaneously performed on both the original and generated data points.

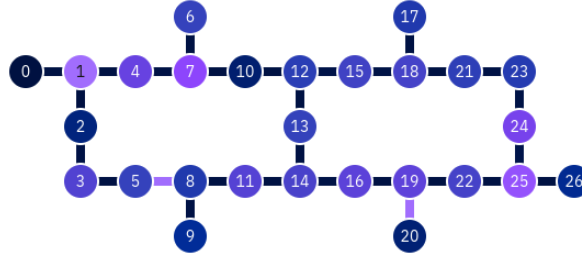


Figure 5.22: Architecture of IBM_hanoi quantum computer. The C-NOT connectivity is reported. The colors of single qubits and their connections represent respectively the T_1 and C-NOT error probabilities. Darker colours represent a lower T_1 , in the range $[47, 265]\mu s$. For the C-NOT errors, darker colors represent a lower error rate, in the range $[3.2 \cdot 10^{-3}, 1]$. Calibration data are referred to 19/10/2023.

5.3.5 Quantum hardware execution

Executing quantum circuits on current quantum devices poses significant challenges even when utilizing cutting-edge quantum hardware. The primary obstacles stem from the substantial error rates associated with quantum gates, particularly the C-NOT gates required for entangling qubits.

A second problem comes from the short qubit relaxation time T_1 [436]. This value, also known as qubit lifetime, provides an estimate of how long a qubit can be used effectively in quantum computations before it becomes too prone to errors caused by decoherence.

Another critical issue regards the quantum computer's architectural connectivity, since it's impossible to apply C-NOT gates between all qubit pairs. Contemporary NISQ devices provide connectivity exclusively between neighboring qubits, typically organized in linear or circular configurations.

Figure 5.22 illustrates the architecture of the quantum chip IBM_hanoi³ employed in this study. When it's necessary to interact qubits that are not directly connected, SWAP gates must be introduced in the circuit to exchange the quantum states of two qubits. It's worth noting that each SWAP gate requires three C-NOT gates. Given these constraints, achieving significant results using the circuit described in Section 5.3.3 is not possible. For the implementation of a QDM on quantum hardware, several modifications are essential to reduce the circuit's complexity.

Quantum circuit adaptation

In this section, we outline the modifications made to the quantum circuit to address the issues introduced before. Regarding the circuit's connectivity, it's worth noting that the PQC ansatz proposed in Section 5.3.3 necessitates only interactions between neighboring qubits when they are arranged in a circular topology. However, in our quantum hardware configuration (Figure 5.9), arranging six qubits in a circular pattern is not feasible. To mitigate this, we removed the last C-NOT gate from each entangling layer, thereby requiring interactions solely between neighboring qubits arranged linearly. This adjustment eliminated the need for SWAP gates within the circuit, significantly reducing the overall count of C-NOT gates.

To reduce the circuit depth, we have applied several changes. The first important change pertains to the number of denoising steps. In fact, the complete quantum circuit is composed of a repetition of the denoising circuit for each denoising step. We found that it is still possible to generate samples using just three denoising steps. The second major change involves using a smaller

³https://quantum-computing.ibm.com/services/resources?system=ibm_hanoi, 2021.

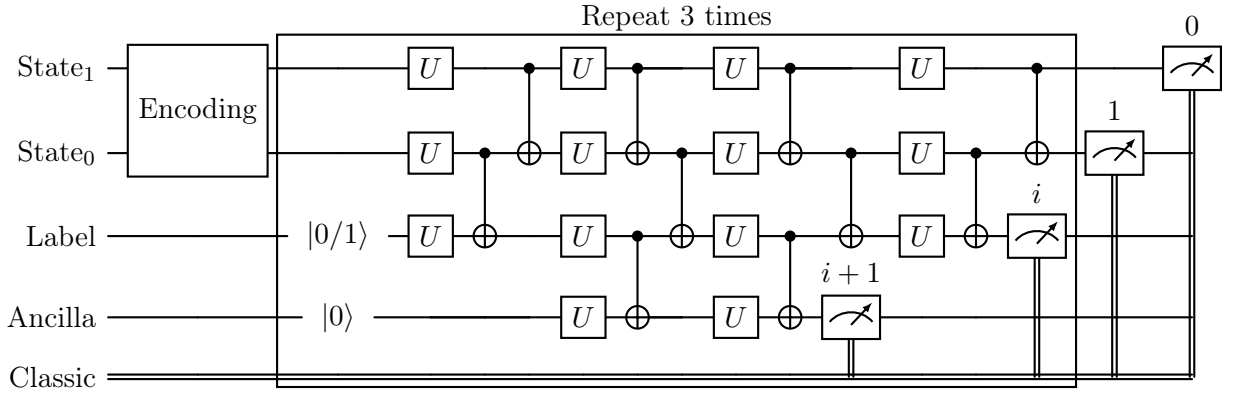


Figure 5.23: Quantum hardware adapted PQC. The central part of the circuit is repeated three times in total. State preparation is necessary only to encode the initial noise. Similarly, the state qubits are measured only at the end.

denoising circuit by reducing the number of layers. In particular, we have used a reverse-bottleneck ansatz (Section 5.3.3), reported in Figure 5.23. To restore the label qubit after its measurement, a conditional X gate is applied. To reduce the circuit depth further, we have also rearranged the C-NOT gate disposition in the entangling layers. In particular, we first apply all the C-NOT gates with an even control qubit and then the ones with an odd control qubit. This way, only two circuit moments are required for each entangling layer.

The expressive power of the quantum circuit is highly reduced by all these adaptations; therefore, we have reduced the complexity of the task. We have restricted the problem to the conditioned generation of only zeros and ones; this way, only one qubit is required to encode the label. We have also reduced the latent space dimension from 8 to 4; this way, only two qubits are required to encode the state. The final circuit is thus composed of a total of four qubits and 37 C-NOT gates (36 for the PQC plus one for the amplitude encoding of the initial noise).

Results

The hardware-adapted PQC has been trained in simulation. Training on hardware devices is unfeasible on NISQ devices because of the high level of noise, which makes the computation of gradients too imprecise. The training has been performed in the same way as described in Section 5.3.4 using “zero” and “one” handwritten digits (2048 samples). The trained weights can be used for sampling on the real device. Before the hardware execution, we have tested the trained algorithm in noiseless simulations. We have generated $5 \cdot 10^3$ latent vectors of both “zero” and “one” digits in order to compare their distribution with respect to the latent vectors of the training set. Figure 5.24 shows the two-dimensional PCA representation of the latent space for original and generated latent vectors. It is possible to observe that the model is able to correctly generate latent vectors with the distribution of the “one” digits. On the other hand, for the “zero” digits, there is not a full overlap between the two distributions. The model can generate samples that belong exclusively to a specific region within the original distribution, although the generated samples show good variability. We have observed similar results in different training executions.

In order to study the effects of noise, we have generated the latent vector of a “zero” digit in four different ways: state vector simulation, shots simulation, realistic noise simulation, and quantum

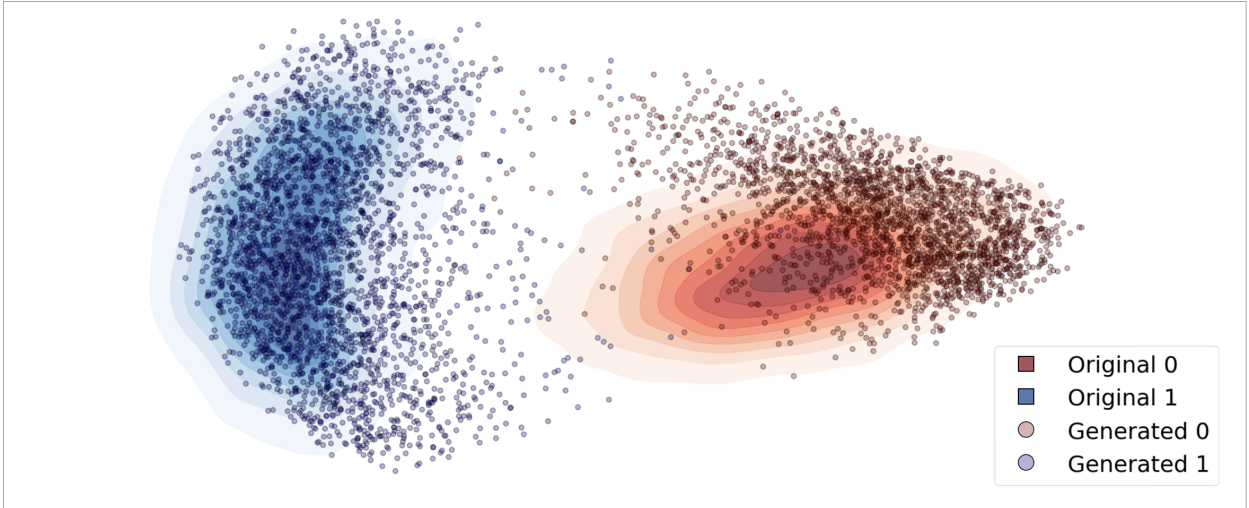


Figure 5.24: 2D representation (PCA) of the latent vectors of the original distribution (KDE plots) and generated distribution obtained in simulation (circles). The original distribution is composed by all the “zeros” and “ones” handwritten digits of the MNIST dataset. The generated distribution contains $5 \cdot 10^3$ samples for each class.

hardware. The realistic noise simulation has been performed with the Qiskit [174] built-in noise model, which uses the calibration data to perform a shots simulation with realistic noise. Apart from the state vector simulation, all other methods require measuring the output states. The need to perform these measurements creates some difficulties, since the final state depends on the results of the measurement of the ancillary qubits. Six measurements are performed before the final state, three for the ancillary qubit and three for the label qubit. This means that, starting from the same initial noise, it is possible to obtain 64 possible final states. In order to sample them all, it is necessary to execute the circuit over a large number of shots. In this study, we have performed $3.2 \cdot 10^5$ shots for every circuit, obtaining, on average, 500 shots for each possible final state. Figure 5.25 reports the measurement frequency for each of the methods described before. It is possible to observe how the shot noise obtained with the shots simulation is negligible, with no significant difference with respect to the state vector simulation. The difference between the hardware execution and the state vector simulation is more relevant. The noise changes the amplitudes of the states 01 and 11 from about 0.1 to 0.2. This is an expected result, as decoherence, due to unwanted interactions with the environment, tends to bring the output state closer to the maximally mixed state. The realistic noise simulation is in good accordance with the real result, with the differences being in the range of 5 – 10%. The effects of noise are relevant but do not influence the quality of the generated samples. Figure 5.26 shows the comparison of a “zero” and a “one” digit, generated with state vector simulation and on quantum hardware. The latent vectors have been decoded with the classical decoder. It is possible to observe how, in the presence of noise, the generated images change slightly, but still show all the features of the correct class.

We have used the results obtained on quantum hardware to study the effects of noise on the variability of the generated samples. Figure 5.27 reports a two-dimensional PCA representation of the latent space for the original and generated dataset, both in simulation and on quantum hardware. In both cases, the generated latent vectors have been obtained by executing $3.2 \cdot 10^5$ shots starting from the same initial noise. With this process, as previously explained, it is possible

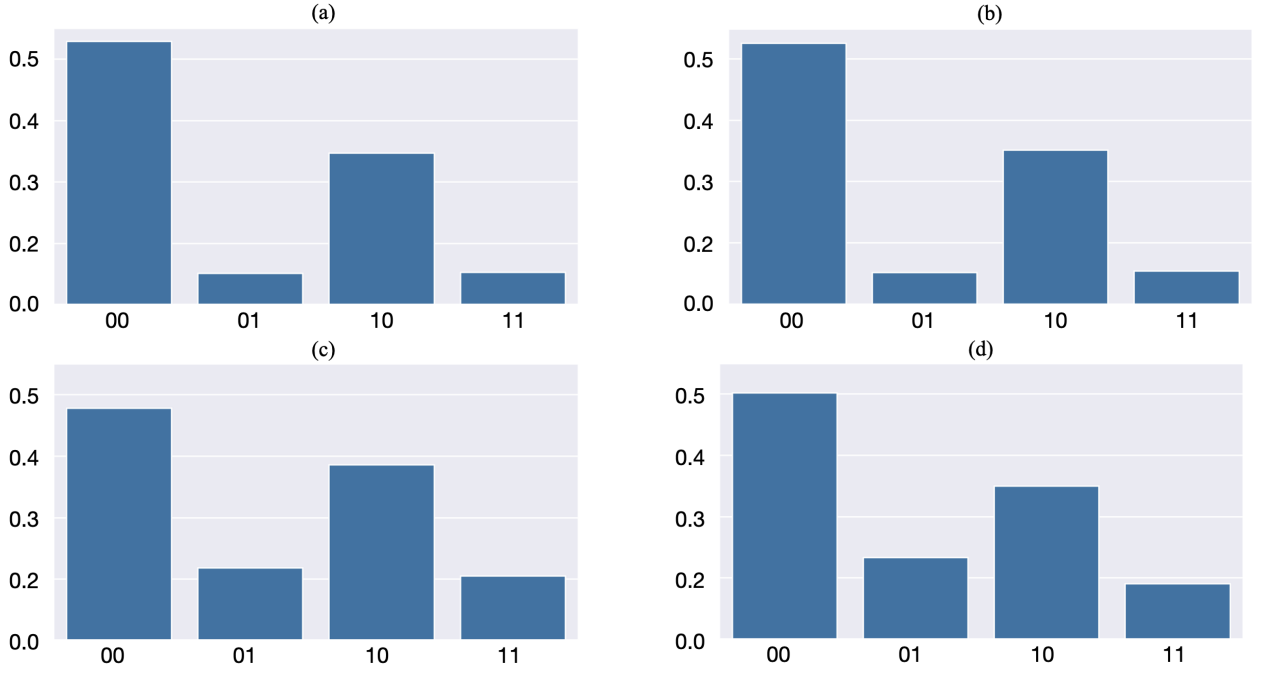


Figure 5.25: State amplitudes of a generated latent vector obtained with: state vector simulation (a), shots simulation (b), realistic noise simulation (c) and hardware execution (d).

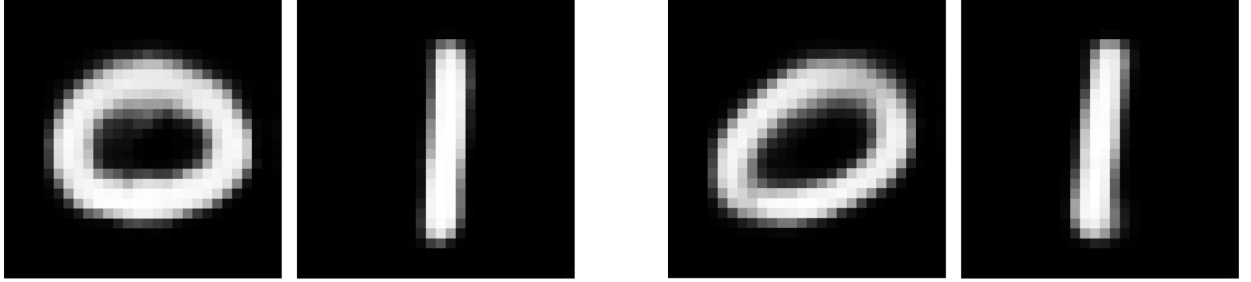


Figure 5.26: Generated images of a “zero” and a “one” handwritten digits. The latent vector has been generated with both a noiseless state vector simulation (left) and with real quantum hardware (right) in order to study the effects of noise.

to obtain 64 samples for each class. In Figure 5.27, it is possible to observe that noise reduces the variability of the generated samples. This is an opposite result with respect to the classical case, where introducing noise after each denoising step helps increase the sample variability [415]. The reduced variability with noise can be attributed to decoherence, which converges all final states towards the maximally mixed state. Moreover, measurement errors tend to mix different samples, thus averaging them. This result highlights how the intrinsic differences between classical and quantum noise lead to differences in the behavior of classical and quantum generative algorithms.

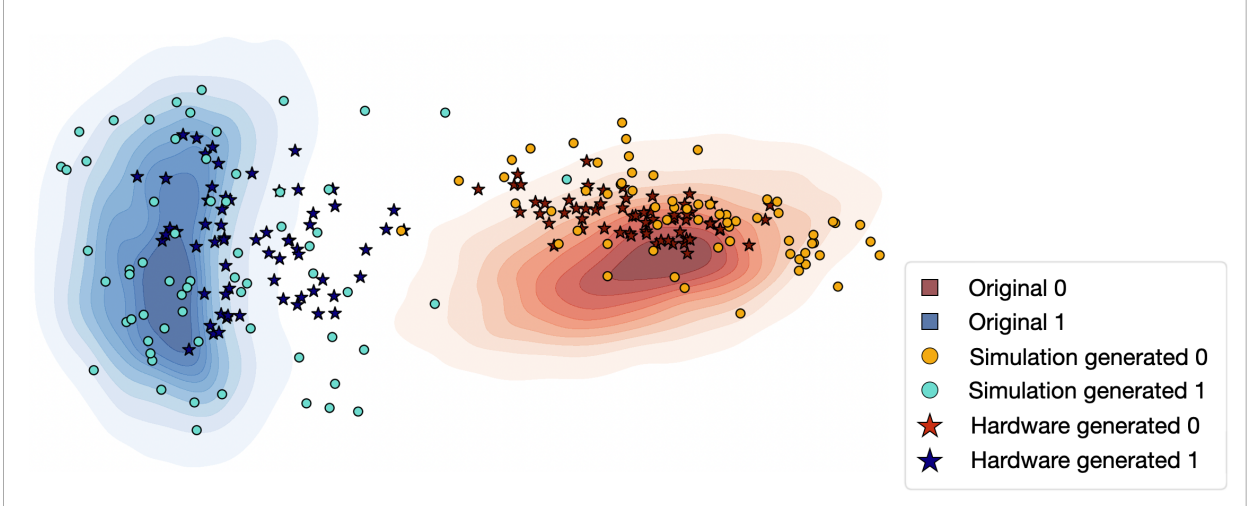


Figure 5.27: 2D representation obtained with PCA of the latent vectors distribution obtained from the original MNIST dataset (KDE plots), data points generated with quantum hardware (stars) and with noiseless shots simulation (circles). The generated distributions contain 64 samples generated with $3.2 \cdot 10^5$ shots and starting from the same initial noise.

5.3.6 Conclusions

In this work, we have introduced and analyzed a novel quantum diffusion model, consisting of two distinct variants. The first is a comprehensive quantum model designed to generate classical or quantum data directly. The second is a latent classical-quantum model intended for the generation of classical data. Both models can be conditioned using additional qubits to encode labels, thereby enhancing their practical utility. In our experiments, we have demonstrated that it is possible to execute simple versions of these algorithms on current NISQ devices. The results obtained show that quantum diffusion models require fewer parameters with respect to their classical counterparts. For the latent model, this allows generating samples of quality comparable to those generated by classical algorithms. In the full quantum case, it is possible to manipulate distributions with a number of features that scales exponentially with the number of qubits. Moreover, this model allows the direct generation of quantum states that can be employed for different applications like quantum cryptography, quantum metrology, and benchmarking.

We are currently working to determine if it is possible to define a diffusion process directly in the Hilbert space, in order to implement a fully quantum pipeline that doesn't require classical computations, even in the training phase. This would allow training the algorithm directly on quantum hardware in the hypothesis that low noise devices will become available. This development could enable the processing of quantum datasets currently beyond the capability of classical hardware.

Chapter 6

Conclusions

In this thesis, we have presented different examples of how machine learning and quantum computing can benefit from each other. In particular, it is possible to observe that machine learning can be used to address some of the most challenging problems that affect quantum computing. On the other hand, quantum computing hardware can be used to execute the quantum counterpart of machine learning algorithms in order to process quantum data. The main difficulties encountered during the work are mainly related to the current status of quantum computing hardware. In fact, NISQ devices are a limitation for the execution of quantum machine learning algorithms. Moreover, the scarce availability of quantum resources creates some difficulties in testing machine learning algorithms for the improvement of quantum computing. Despite these limitations, it has been possible to find interesting results that can be used to improve the current status of both quantum computing and machine learning. In this chapter, we will review the main results that have been obtained in the original contributions and discuss possible future perspectives and open challenges for the research. A summary of the original contributions presented in this thesis is as follows.

Convolutional neural network based decoders for surface codes [15] has been published in 2023 in *Quantum Information Processing* (Springer). The short paper *Quantum circuit noise simulation with reinforcement learning* [22] has been published in the proceedings of AIQxQIA 2023. The preprint *Quantum noise modeling through reinforcement learning* is available on arXiv [23] and is currently under review. This work has been presented at several conferences and workshops, including AIQxQIA 2024 in Bolzano, QC-INFN 2024 in Padova, and ICSC-Spoke10 2024 in Milan. *Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits* [41] has been published in 2023 in *Particles* (MDPI) and presented at several conferences including QC-INFN 2022 in Bologna, QAI 2023 in Naples, and EuCAIFCon 2024 in Amsterdam. *Quantum Diffusion Models* [42] is a preprint available on arXiv and has been presented at AIQxQIA 2023 in Rome, QC-INFN 2024 in Padova, and EuCAIFCon 2024 in Amsterdam. A short version of the paper has been accepted in the proceedings of the EuCAIFCon 2025 conference and will be published soon. *Qibolab: an open-source hybrid quantum operating system* has been published in 2024 in *Quantum* [35] and presented in several national and international conferences.

The first original contribution presented in this thesis regards the application of machine learning to the problem of quantum error correction. We have tested different neural network architectures

(FFNN and CNN) for decoding surface codes of different sizes. The results, in agreement with the literature, show that the CNN architecture allows for fast decoding times and high decoding accuracy. We have tested different strategies to improve the performance of CNN-based decoders, such as the correct choice of the training set and the use of dilated convolutions. The most interesting results have been obtained by using explainable ML techniques to gain insights into the decoder, with particular attention to the causes of failure in the decoding process. In particular, we have shown that it is possible to use this information to improve the performance of the algorithm. This performance enhancement with data augmentation driven by model explainability is a novel contribution that allows for improving the training process without the need to increase the size of the training dataset.

The biggest limitation of this work regards the scaling of the decoder to large surface codes. This scaling would require a large amount of computational resources both for the training of the algorithm and for the generation of the training dataset. Moreover, it has not been possible to test the decoder on a real surface code due to the lack of this kind of hardware.

We hope that this research will be useful for the future development of machine learning-based decoders and that this error correction technique will become available for the next generation of quantum computers.

The second original contribution is also related to the application of machine learning to quantum computing. In this case, we have used reinforcement learning to characterize and simulate the noise impacting a quantum chip. We have shown that it is possible to train a RL agent to learn how to introduce noise channels in a quantum circuit in order to simulate the noise that would be obtained in the execution on a specific hardware device. The results show that this approach reduces heuristic assumptions about the noise model, enhancing generalization properties. Tests on both simulated and real noisy devices have shown that the RL agent outperforms standard noise modeling techniques. We have also applied the agent to the noisy simulation of renowned quantum algorithms in order to test the performance of the algorithm on structured circuits with characteristics different from the training set.

The main limitation of this work is related to the scalability to chips with a large number of qubits. In fact, the method relies on quantum state tomography which requires an exponential number of measurements with the number of qubits. Moreover, the action space of the agent grows linearly with the number of qubits, making the training process more difficult. We have proposed some possible strategies to overcome these limitations and we will try to implement them in future works. The novel idea of training a RL agent for noise simulations is not only limited to this application; possible future applications include error mitigation and transpiling optimization.

We hope that this research will be useful in bridging the gap between quantum hardware and simulations, by making the simulation of quantum algorithms in the NISQ era more accurate and reliable.

The third original contribution, presented in this thesis, regards the implementation of a standard machine learning algorithm on a quantum computer. In particular, we have used a parametrized quantum circuit, in place of a classical neural network, to perform anomaly detection. The objective of this work was not to outperform the classical version of the algorithm, but to show that it is possible to use this kind of QML algorithms to process quantum data. The algorithm has been applied

to a real use-case scenario in the field of high energy physics, for the identification of anomalous decay patterns. The results show that the quantum version of the algorithm is able to identify the anomalies with a performance comparable to the classical version. The ability to directly process quantum data may become important if instruments based on quantum sensing will be applied to high energy physics. We have also implemented a reduced version of the algorithm, applied to an easier task, on a real quantum computer. This result allows us to benchmark the current status of quantum hardware in the deployment of this kind of QML algorithms.

The main limitations of this work are related to the high levels of noise in the quantum devices and the unavailability of quantum data. In fact, we have observed that it is possible to use NISQ devices only to perform simple tasks. Increasing the difficulty of the task would require more expressive power of the parametrized quantum circuit, which results in a higher number of gates, thus increasing the sensitivity to noise. Moreover, the lack of quantum data requires encoding classical data using amplitude encoding, which is not scalable to a large number of qubits.

We hope that this research will pave the way for the implementation of more complex QML algorithms on the next generations of quantum computers, for the processing of quantum data.

The final original contribution presented in this thesis is a feasibility study on the realization of the quantum version of a generative diffusion model. In this work, we have shown that it is possible to use quantum circuits to manipulate probability distributions for the generative task. Due to the intrinsic differences between parametrized quantum circuits and classical neural networks, we have proposed some adaptations of the original model. In particular, we have used complex noise in the diffusion process, which is implemented classically, and we have used a full quantum pipeline only in the denoising process. We have also proposed different variations of the model, such as a conditioned version and a hybrid classical-quantum version where the generative process is performed in the latent space of a classical autoencoder. A reduced version of the algorithm has also been executed on a real quantum device for the generation of two-qubit states.

The results show that quantum circuits, even in simulations, can be used to manipulate probability distributions but only for simple tasks. We have compared the quantum diffusion model with the classical version using some metrics from the literature; however, the latter version outperforms the quantum counterpart. Moreover, the quantum diffusion model is very sensitive to the hyperparameters and tends to collapse to a single mode. We are currently working on improving the performance of the quantum diffusion model and solving the issue of mode collapse. The main limitation comes from the expressive power of the quantum circuits, which are only able to perform unitary transformations, in contrast with classical neural networks that can perform highly non-linear transformations.

It is unlikely that quantum diffusion models will outperform classical generative models in the near future. However, the generation of quantum states sampled from a well-defined distribution is an interesting task that may find application in fields such as quantum cryptography, quantum metrology, and benchmarking.

Several challenges are open in the field of quantum computing. The development of quantum hardware is progressing rapidly but is still not sufficient to run complex quantum algorithms. In parallel with the development of quantum hardware, there is a need for the development of the soft-

ware stack that allows the execution of quantum algorithms. The creation of a “quantum operating system” is the target of the Qibolab project, and more generally of the Qibo framework, which I have presented in this thesis. The collaboration with this project has allowed me to think about possible future advancements in the field. In particular, possible future research directions that I would like to explore are the optimization of the transpiling process and the development of analog quantum machine learning algorithms.

The transpiling process, which I had the possibility to explore during my collaboration with the Qibolab project, is an important open problem in the field of quantum computing. The main issues are related to the trade-off between the optimization level and the computational resources required. An interesting approach to this problem is the use of machine learning techniques to solve the optimization problem, with particular regard to the use of reinforcement learning. Some new studies have already been proposed in this direction [192, 437, 438], and possible future advancements could come from the realization of chip-specific noise-sensitive transpilers. Also, the use of new neural network architectures and explainable ML techniques could lead to a performance improvement of the transpiling process. The work on the noise simulation RL agent could be a starting point for the development of this kind of algorithms. Scaling the algorithm to a large number of qubits is a challenge that can be addressed in future works.

The development of analog quantum machine learning algorithms is another interesting research direction. In particular, it is possible to tune the interactions between qubits in a chip in order to implement a specific transfer function [439]. This approach finds application in the field of quantum sensing [440, 441]. For example, superconductive qubits can be used to detect photons, and implementing a transfer function embedded in the detector system may be used to extract useful information from the detected photon [442]. The application of superconductive qubits to photon detection is particularly interesting in a completely different topic: the research of dark matter [443, 444]. Several projects are currently working in this field, such as the *Qub-it* group [445, 446] at the INFN laboratories in Frascati (Rome), with which I am starting a collaboration. My contribution to the project aims at the application of classical and quantum machine learning techniques to improve the performance of the detectors.

The fields of quantum computing and machine learning will undoubtedly be two of the most interesting ones in the next decades. With the works presented in this project and my future projects, I hope to contribute to the development of these fields for the benefit of everyone.

Bibliography

- [1] Johannes Bausch, Andrew Senior, Francisco Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Yuezhen Niu, Sam Blackwell, George Holland, Dvir Kafri, Juan Atalaya, Craig Gidney, Demis Hassabis, Sergio Boixo, Hartmut Neven, and Pushmeet Kohli. Learning high-accuracy error decoding for quantum processors. *Nature*, 635:834–840, 11 2024.
- [2] Savvas Varsamopoulos, Koen Bertels, and Carmen G. Almudever. Decoding surface code with a distributed neural network based decoder, 2019.
- [3] Paul Baireuther, Thomas E. O’Brien, Brian Tarasinski, and Carlo W. J. Beenakker. Machine-learning-assisted correction of correlated qubit errors in a topological code. *Quantum*, 2:48, Jan 2018.
- [4] Giacomo Torlai and Roger G. Melko. Neural decoder for topological codes. *Physical Review Letters*, 119(3), Jul 2017.
- [5] Kai Meinerz, Chae-Yeun Park, and Simon Trebst. Scalable neural decoder for topological surface codes, 2021.
- [6] Xiaotong Ni. Neural network decoders for large-distance 2d toric codes. *Quantum*, 4:310, Aug 2020.
- [7] Amarsanaa Davaasuren, Yasunari Suzuki, Keisuke Fujii, and Masato Koashi. General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes. *Physical Review Research*, 2(3), Sep 2020.
- [8] Milap Sheth, Sara Zafar Jafarzadeh, and Vlad Gheorghiu. Neural ensemble decoding for topological quantum error-correcting codes. *Physical Review A*, 101(3), Mar 2020.
- [9] Debasmita Bhounik, Pinaki Sen, Ritajit Majumdar, Susmita Sur-Kolay, Latesh Kumar K J, and Sundaraja Sitharama Iyengar. Efficient decoding of surface code syndromes for error correction in quantum computing, 2021.
- [10] Haoran Liao, Derek S. Wang, Iskandar Sitdikov, Ciro Salcedo, Alireza Seif, and Zlatko K. Mineev. Machine learning for practical quantum error mitigation. *Nature Machine Intelligence*, 6(12):1478–1486, November 2024.
- [11] Jihye Kim, Byungdu Oh, Yonuk Chong, Euyheon Hwang, and Daniel K Park. Quantum readout error mitigation via deep learning. *New Journal of Physics*, 24(7):073009, July 2022.

- [12] Andrey Zhukov and Walter Pogosov. Quantum error mitigation in the regime of high noise using deep neural network: Trotterized dynamics. *Quantum Information Processing*, 23(3), February 2024.
- [13] Temitope Adeniyi and Sathish Kumar. Adaptive neural network for quantum error mitigation. *Quantum Machine Intelligence*, 7, 01 2025.
- [14] M. Liao, Y. Zhu, G. Chiribella, and Y. Yang. Noise-agnostic quantum error mitigation with data augmented neural models. *npj Quantum Information*, 11(1):8, 2025. Epub 2025 Jan 18.
- [15] Bordoni Simone and Giagu Stefano. Convolutional neural network based decoders for surface codes. *Quantum Information Processing*, 22(3):151, 3 2023.
- [16] Hannah Lange, Anka Van de Walle, Atiye Abedinnia, and Annabelle Bohrdt. From architectures to applications: A review of neural quantum states, 2024.
- [17] David R. Vivas, Javier Madroñero, Victor Bucheli, Luis O. Gómez, and John H. Reina. Neural-network quantum states: A systematic review, 2022.
- [18] Alexander Zlokapa and Alexandru Gheorghiu. A deep learning model for noise prediction on near-term quantum devices, 2020.
- [19] Christopher J. Wood. Special session: Noise characterization and error mitigation in near-term quantum computers. In *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pages 13–16, 2020.
- [20] Konstantinos Georgopoulos, Clive Emary, and Paolo Zuliani. Modeling and simulating the noisy behavior of near-term quantum computers. *Phys. Rev. A*, 104:062432, 12 2021.
- [21] Robin Harper, Steven T. Flammia, and Joel J. Wallman. Efficient learning of quantum noise. *Nature Physics*, 16(12):1184–1188, 8 2020.
- [22] Simone Bordoni, Andrea Papaluca, Piergiorgio Buttarini, Alejandro Sopena, Stefano Giagu, and Stefano Carrazza. Quantum circuit noise simulation with reinforcement learning. In *AI for Quantum and Quantum for AI 2023*, 2023.
- [23] Simone Bordoni, Andrea Papaluca, Piergiorgio Buttarini, Alejandro Sopena, Stefano Giagu, and Stefano Carrazza. Quantum noise modeling through reinforcement learning, 2024.
- [24] Meng-Yun Mao, Zheng Cheng, Yan Xia, Andrzej M. Oleś, and Wen-Long You. Machine-learning-inspired quantum optimal control of nonadiabatic geometric quantum computation via reverse engineering. *Physical Review A*, 108(3), September 2023.
- [25] Bikrant Bhattacharyya, Fredy An, Dominik Kozbiel, Andy J. Goldschmidt, and Fred-eric T. Chong. Using optimal control to guide neural-network interpolation of continuously-parameterized gates, 2024.
- [26] Murphy Yuezhen Niu, Sergio Boixo, Vadim Smelyanskiy, and Hartmut Neven. Universal quantum control through deep reinforcement learning, 2018.

- [27] Emily Wright and Rogério De Sousa. Fast quantum gate design with deep reinforcement learning using real-time feedback on readout signals. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, page 1295–1303. IEEE, September 2023.
- [28] Omar Shindi, Qi Yu, Parth Girdhar, and Daoyi Dong. Model-free quantum gate design and calibration using deep reinforcement learning, 2023.
- [29] Shouliang Hu, Chunlin Chen, and Daoyi Dong. Deep reinforcement learning for control design of quantum gates. In *2022 13th Asian Control Conference (ASCC)*, pages 2367–2372, 2022.
- [30] Giuseppe Di Guglielmo, Botao Du, Javier Campos, Alexandra Boltasseva, Akash V. Dixit, Farah Fahim, Zhaxylyk Kudyshev, Santiago Lopez, Ruichao Ma, Gabriel N. Perdue, Nhan Tran, Omer Yesilyurt, and Daniel Bowring. End-to-end workflow for machine learning-based qubit readout with qick and hls4ml, 2025.
- [31] Jihye Kim, Byungdu Oh, Yonuk Chong, Euyheon Hwang, and Daniel Park. Quantum readout error mitigation via deep learning. *New Journal of Physics*, 24:073009, 07 2022.
- [32] Claudio Bonizzoni, Mirco Tincani, Fabio Santanni, and Marco Affronte. Machine-learning-assisted manipulation and readout of molecular spin qubits. *Phys. Rev. Appl.*, 18:064074, Dec 2022.
- [33] Élie Genois, Jonathan A. Gross, Agustin Di Paolo, Noah J. Stevenson, Gerwin Koolstra, Akel Hashim, Irfan Siddiqi, and Alexandre Blais. Quantum-tailored machine-learning characterization of a superconducting qubit. *PRX Quantum*, 2:040355, Dec 2021.
- [34] Yuval Baum, Mirko Amico, Sean Howell, Michael Hush, Maggie Liuzzi, Pranav Mundada, Thomas Merkh, Andre R.R. Carvalho, and Michael J. Biercuk. Experimental deep reinforcement learning for error-robust gate-set design on a superconducting quantum computer. *PRX Quantum*, 2:040324, Nov 2021.
- [35] Stavros Efthymiou, Alvaro Orgaz-Fuertes, Rodolfo Carobene, Juan Cereijo, Andrea Pasquale, Sergi Ramos-Calderer, Simone Bordoni, David Fuentes-Ruiz, Alessandro Candido, Edoardo Pedicillo, Matteo Robbiati, Yuanzheng Paul Tan, Jadwiga Wilkens, Ingo Roth, José Ignacio Latorre, and Stefano Carrazza. Qibolab: an open-source hybrid quantum operating system. *Quantum*, 8:1247, February 2024.
- [36] Armando Bellante, Alessandro Luongo, and Stefano Zanero. Quantum algorithms for svd-based data representation and analysis. *Quantum Machine Intelligence*, 4(2), August 2022.
- [37] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum speed-up for unsupervised learning. *Machine Learning*, 90(2):261–287, February 2013.
- [38] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, January 2018.

-
- [39] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, July 2021.
 - [40] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 9 2017.
 - [41] Simone Bordoni, Denis Stanev, Tommaso Santantonio, and Stefano Giagu. Long-lived particles anomaly detection with parametrized quantum circuits. *Particles*, 6(1):297–311, 2023.
 - [42] Andrea Cacioppo, Lorenzo Colantonio, Simone Bordoni, and Stefano Giagu. Quantum diffusion models, 2023.
 - [43] Jonas Maziero. Random sampling of quantum states: a survey of methods: And some issues regarding the overparametrized method. *Brazilian Journal of Physics*, 45(6):575–583, September 2015.
 - [44] Prabhanjan Ananth, Luowen Qian, and Henry Yuen. Cryptography from pseudorandom quantum states, 2022.
 - [45] M. Oszmaniec, R. Augusiak, C. Gogolin, J. Kołodyński, A. Acín, and M. Lewenstein. Random bosonic states for robust quantum metrology. *Phys. Rev. X*, 6:041044, Dec 2016.
 - [46] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80(1), jul 2009.
 - [47] Patrick Hayden and John Preskill. Black holes as mirrors: quantum information in random subsystems. *Journal of High Energy Physics*, 2007(09):120–120, September 2007.
 - [48] Jonah Kudler-Flam, Vladimir Narovlansky, and Shinsei Ryu. Negativity spectra in random tensor networks and holography. *Journal of High Energy Physics*, 2022(2), February 2022.
 - [49] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 10 1997.
 - [50] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.
 - [51] Frank Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.
 - [52] Philip Ball. Physicists in China challenge Google’s ‘quantum advantage’. *Nature*, 588(7838):380–380, December 2020.
 - [53] Han-Sen Zhong, Yu-Hao Deng, Jian Qin, Hui Wang, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Dian Wu, Si-Qiu Gong, Hao Su, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Jelmer J. Renema, Chao-Yang Lu, and Jian-Wei Pan. Phase-programmable gaussian boson sampling using stimulated squeezed light. *Phys. Rev. Lett.*, 127:180502, Oct 2021.

- [54] Lars Madsen, Fabian Laudenbach, Mohsen Askarani, Fabien Rortais, Trevor Vincent, Jacob Bulmer, Filippo Miatto, Leonhard Neuhaus, Lukas Helt, Matthew Collins, Adriana Lita, Thomas Gerrits, Sae Nam, Varun Vaidya, Matteo Menotti, Ish Dhand, Zachary Vernon, Nicolas Quesada, and Jonathan Lavoie. Quantum computational advantage with a programmable photonic processor. *Nature*, 606:75–81, 06 2022.
- [55] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.
- [56] Peter Nimbe, Benjamin Asubam Weyori, and Adebayo Felix Adekoya. Models in quantum computing: a systematic review. *Quantum Information Processing*, 20(2):80, 2021.
- [57] Atanu Rajak, Sei Suzuki, Amit Dutta, and Bikas K. Chakrabarti. Quantum annealing: an overview. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2241), December 2022.
- [58] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, January 2009.
- [59] Philipp Hauke, Helmut G Katzgraber, Wolfgang Lechner, Hidetoshi Nishimori, and William D Oliver. Perspectives of quantum annealing: Methods and implementations. *Reports on Progress in Physics*, 83(5):054401, 2020.
- [60] David C. McKay, Stefan Filipp, Antonio Mezzacapo, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. Universal gate for fixed-frequency qubits via a tunable bus. *Physical Review Applied*, 6(6), December 2016.
- [61] Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of quantum computation and information-volume I: Basic concepts*. World scientific, 2004.
- [62] John E. Hopcroft. Turing machines. *Scientific American*, 250(5):86–E9, 1984.
- [63] Arjen K Lenstra, W Hendrik Jr, et al. *The development of the number field sieve*, volume 1554. Springer Science & Business Media, 1993.
- [64] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9–11):771–783, September 2000.
- [65] V. V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. L. Brock, A. Z. Ding, L. Frunzio, S. M. Girvin, R. J. Schoelkopf, and M. H. Devoret. Real-time quantum error correction beyond break-even. *Nature*, 616(7955):50–55, March 2023.
- [66] Eliot Kapit. Hardware-efficient and fully autonomous quantum error correction in superconducting circuits. *Phys. Rev. Lett.*, 116:150501, Apr 2016.
- [67] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz. Realization of real-time fault-tolerant quantum error correction. *Phys. Rev. X*, 11:041058, Dec 2021.

-
- [68] Alexander P. M. Place, Lila V. H. Rodgers, Pranav Mundada, Basil M. Smitham, Mattias Fitzpatrick, Zhaoqi Leng, Anjali Premkumar, Jacob Bryon, Andrei Vrajitoarea, Sara Sussman, Guangming Cheng, Trisha Madhavan, Harshvardhan K. Babla, Xuan Hoang Le, Youqi Gang, Berthold Jäck, András Gyenis, Nan Yao, Robert J. Cava, Nathalie P. de Leon, and Andrew A. Houck. New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds. *Nature Communications*, 12(1), March 2021.
 - [69] Aaron Somoroff, Quentin Ficheux, Raymond A. Mencia, Haonan Xiong, Roman Kuzmin, and Vladimir E. Manucharyan. Millisecond coherence in a superconducting qubit. *Phys. Rev. Lett.*, 130:267001, Jun 2023.
 - [70] Irfan Siddiqi. Engineering high-coherence superconducting qubits. *Nature Reviews. Materials*, 6(10), 9 2021.
 - [71] Paul D. Nation, Hwajung Kang, Neereja Sundaresan, and Jay M. Gambetta. Scalable mitigation of measurement errors on quantum computers. *PRX Quantum*, 2:040326, 11 2021.
 - [72] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, Paolo Stornati, and Xiaoyang Wang. Measurement error mitigation in quantum computers through classical bit-flip correction. *Phys. Rev. A*, 105:062404, 6 2022.
 - [73] Alejandro Sopena, Max Hunter Gordon, Germán Sierra, and Esperanza López. Simulating quench dynamics on a digital quantum computer with data-driven error mitigation. *Quantum Science and Technology*, 6(4):045003, 7 2021.
 - [74] Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C. Benjamin, and Ying Li. Learning-based quantum error mitigation. *PRX Quantum*, 2:040330, 11 2021.
 - [75] Masatoshi Sato and Yoichi Ando. Topological superconductors: a review. *Reports on Progress in Physics*, 80(7):076501, may 2017.
 - [76] Daniel Loss and David P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120–126, Jan 1998.
 - [77] Karen Wintersperger, Florian Dommert, Thomas Ehmer, Andrey Housanov, Johannes Klepsch, Wolfgang Mauere, Georg Reuber, Thomas Strohm, Ming Yin, and Sebastian Luber. Neutral atom quantum computing hardware: performance and end-user perspective. *EPJ Quantum Technology*, 10(1), August 2023.
 - [78] Bryan T. Gard, Keith R. Motes, Jonathan P. Olson, Peter P. Rohde, and Jonathan P. Dowling. *An Introduction to Boson-Sampling*, page 167–192. WORLD SCIENTIFIC, June 2015.
 - [79] Alexander S. Delliös, Margaret D. Reid, and Peter D. Drummond. Simulating gaussian boson sampling quantum computers, 2023.
 - [80] Changhun Oh, Liang Jiang, and Bill Fefferman. Spoofing cross-entropy measure in boson sampling. *Physical Review Letters*, 131(1), July 2023.
-

- [81] Ali Ibnun Nurhadi and Nana Rachmana Syambas. Quantum key distribution (qkd) protocols: A survey. In *2018 4th International Conference on Wireless and Telematics (ICWT)*, pages 1–5, 2018.
- [82] Mehul Malik, Manuel Erhard, Marcus Huber, Mario Krenn, Robert Fickler, and Anton Zeilinger. Multi-photon entanglement in high dimensions. *Nature Photonics*, 10(4):248–252, February 2016.
- [83] Xi-Lin Wang, Xin-Dong Cai, Zu-En Su, Ming-Cheng Chen, Dian Wu, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum teleportation of multiple properties of a single quantum particle, 2014.
- [84] G. J. Milburn. Quantum optical fredkin gate. *Phys. Rev. Lett.*, 62:2124–2127, May 1989.
- [85] Darrick E. Chang, Vladan Vuletić, and Mikhail D. Lukin. Quantum nonlinear optics — photon by photon. *Nature Photonics*, 8:685 – 694, 2014.
- [86] E. Knill, R. Laflamme, and G. Milburn. Efficient linear optics quantum computation, 2000.
- [87] C. R. Myers and R. Laflamme. Linear optics quantum computation: an overview, 2005.
- [88] Joshua W. Silverstone, Damien Bonneau, Jeremy L. O’Brien, and Mark G. Thompson. Silicon quantum photonics. *IEEE Journal of Selected Topics in Quantum Electronics*, 22(6):390–402, November 2016.
- [89] Sergei Slussarenko and Geoff J. Pryde. Photonic quantum information processing: A concise review. *Applied Physics Reviews*, 6(4), October 2019.
- [90] G. N. Gol’tsman, O. Okunev, G. Chulkova, A. Lipatov, A. Semenov, K. Smirnov, B. Voronov, A. Dzardanov, C. Williams, and Roman Sobolewski. Picosecond superconducting single-photon optical detector. *Applied Physics Letters*, 79(6):705–707, 08 2001.
- [91] Dileep V. Reddy, Robert R. Nerem, Adriana E. Lita, Sae Woo Nam, Richard P. Mirin, and Varun B. Verma. Exceeding 95 In *2019 Conference on Lasers and Electro-Optics (CLEO)*, pages 1–2, 2019.
- [92] Blas Cabrera, Roland Clarke, P. Colling, Aaron Miller, Siri Nam, and R. Romani. Detection of single infrared, optical, and ultraviolet photons using superconducting transition edge sensors. *Applied Physics Letters*, 73:735–737, 08 1998.
- [93] Daiji Fukuda, Go Fujii, Takayuki Numata, Kuniaki Amemiya, Akio Yoshizawa, Hidemi Tsuchida, Hidetoshi Fujino, Hiroyuki Ishii, Taro Itatani, Shuichiro Inoue, and Tatsuya Zama. Titanium-based transition-edge photon number resolving detector with 98% detection efficiency with index-matched small-gap fiber coupling. *Opt. Express*, 19(2):870–875, Jan 2011.
- [94] Andreas Christ, Alessandro Fedrizzi, Hannes Hübel, Thomas Jennewein, and Christine Silberhorn. Chapter 11 - parametric down-conversion. In Alan Migdall, Sergey V. Polyakov, Jingyun Fan, and Joshua C. Bienfang, editors, *Single-Photon Generation and Detection*, volume 45 of *Experimental Methods in the Physical Sciences*, pages 351–410. Academic Press, 2013.

-
- [95] David S. Hum and Martin M. Fejer. Quasi-phasematching. *Comptes Rendus Physique*, 8(2):180–198, 2007. Recent advances in crystal optics.
 - [96] W. P. Grice, A. B. U’Ren, and I. A. Walmsley. Eliminating frequency and space-time correlations in multiphoton states. *Phys. Rev. A*, 64:063815, Nov 2001.
 - [97] Sergei Slussarenko, Morgan M. Weston, Helen M. Chrzanowski, Lynden K. Shalm, Varun B. Verma, Sae Woo Nam, and Geoff J. Pryde. Unconditional violation of the shot-noise limit in photonic quantum metrology. *Nature Photonics*, 11(11):700–703, October 2017.
 - [98] Nora Tischler, Farzad Ghafari, Travis J. Baker, Sergei Slussarenko, Raj B. Patel, Morgan M. Weston, Sabine Wollmann, Lynden K. Shalm, Varun B. Verma, Sae Woo Nam, H. Chau Nguyen, Howard M. Wiseman, and Geoff J. Pryde. Conclusive experimental demonstration of one-way einstein-podolsky-rosen steering. *Phys. Rev. Lett.*, 121:100401, Sep 2018.
 - [99] A. L. Migdall, D. Branning, and S. Castelletto. Tailoring single-photon and multiphoton probabilities of a single-photon on-demand source. *Phys. Rev. A*, 66:053805, Nov 2002.
 - [100] Fumihiro Kaneda, Feihu Xu, Joseph Chapman, and Paul G. Kwiat. Quantum-memory-assisted multi-photon generation for efficient quantum information processing. *Optica*, 4(9):1034–1037, Sep 2017.
 - [101] A. I. Lvovsky, B. C. Sanders, and W. Tittel. Optical quantum memory, 2010.
 - [102] N. Somaschi, V. Giesz, L. De Santis, J. C. Loredó, M. P. Almeida, G. Hornecker, S. L. Portalupi, T. Grange, C. Antón, J. Demory, C. Gómez, I. Sagnes, N. D. Lanzillotti-Kimura, A. Lemaître, A. Auffeves, A. G. White, L. Lancio, and P. Senellart. Near-optimal single-photon sources in the solid state. *Nature Photonics*, 10(5):340–345, March 2016.
 - [103] Fulvio Flamini, Nicolò Spagnolo, and Fabio Sciarrino. Photonic quantum information processing: a review. *Reports on Progress in Physics*, 82(1):016001, November 2018.
 - [104] Jeremy L. O’Brien, Akira Furusawa, and Jelena Vučković. Photonic quantum technologies. *Nature Photonics*, 3(12):687–695, December 2009.
 - [105] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. Demonstration of a fundamental quantum logic gate. *Phys. Rev. Lett.*, 75:4714–4717, Dec 1995.
 - [106] Justin G. Bohnet, Brian C. Sawyer, Joseph W. Britton, Michael L. Wall, Ana Maria Rey, Michael Foss-Feig, and John J. Bollinger. Quantum spin dynamics and entanglement generation with hundreds of trapped ions. *Science*, 352(6291):1297–1301, June 2016.
 - [107] Jwo-Sy Chen, Erik Nielsen, Matthew Ebert, Volkan Inlek, Kenneth Wright, Vandiver Chaplin, Andrii Maksymov, Eduardo Pérez, Amrit Poudel, Peter Maunz, and John Gamble. Benchmarking a trapped-ion quantum computer with 29 algorithmic qubits, 2023.
 - [108] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, J. G. Bohnet, N. C. Brown, N. Q. Burdick, W. C. Burton, S. L. Campbell, J. P. Campora, C. Carron, J. Chambers, J. W. Chan, Y. H. Chen,

- A. Chernoguzov, E. Chertkov, J. Colina, J. P. Curtis, R. Daniel, M. DeCross, D. Deen, C. Delaney, J. M. Dreiling, C. T. Ertsgaard, J. Esposito, B. Estey, M. Fabrikant, C. Figgatt, C. Foltz, M. Foss-Feig, D. Francois, J. P. Gaebler, T. M. Gatterman, C. N. Gilbreth, J. Giles, E. Glynn, A. Hall, A. M. Hankin, A. Hansen, D. Hayes, B. Higashi, I. M. Hoffman, B. Horning, J. J. Hout, R. Jacobs, J. Johansen, L. Jones, J. Karcz, T. Klein, P. Lauria, P. Lee, D. Liefer, S. T. Lu, D. Lucchetti, C. Lytle, A. Malm, M. Matheny, B. Mathewson, K. Mayer, D. B. Miller, M. Mills, B. Neyenhuis, L. Nugent, S. Olson, J. Parks, G. N. Price, Z. Price, M. Pugh, A. Ransford, A. P. Reed, C. Roman, M. Rowe, C. Ryan-Anderson, S. Sanders, J. Sedlacek, P. Shevchuk, P. Siegfried, T. Skripka, B. Spaun, R. T. Sprengle, R. P. Stutz, M. Swallows, R. I. Tobey, A. Tran, T. Tran, E. Vogt, C. Volin, J. Walker, A. M. Zolot, and J. M. Pino. A race-track trapped-ion quantum processor. *Phys. Rev. X*, 13:041052, Dec 2023.
- [109] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof. Experimental issues in coherent quantum-state manipulation of trapped atomic ions, 1998.
- [110] G Pagano, P W Hess, H B Kaplan, W L Tan, P Richerme, P Becker, A Kyprianidis, J Zhang, E Birkelbaw, M R Hernandez, Y Wu, and C Monroe. Cryogenic trapped-ion system for large scale quantum simulation. *Quantum Science and Technology*, 4(1):014004, October 2018.
- [111] Robert F. Spivey, Ismail V. Inlek, Zhubing Jia, Stephen Crain, Ke Sun, Junki Kim, Geert Vrijsen, Chao Fang, Colin Fitzgerald, Steffen Kross, Tom Noel, and Jungsang Kim. High stability cryogenic system for quantum computing with compact packaged ion traps, 2021.
- [112] Jurgen Eschner, Giovanna Morigi, Ferdinand Schmidt-Kaler, and Rainer Blatt. Laser cooling of trapped ions. *J. Opt. Soc. Am. B*, 20(5):1003–1015, May 2003.
- [113] K. G. Johnson, J. D. Wong-Campos, A. Restelli, K. A. Landsman, B. Neyenhuis, J. Mizrahi, and C. Monroe. Active stabilization of ion trap radiofrequency potentials. *Review of Scientific Instruments*, 87(5), May 2016.
- [114] Colin D. Bruzewicz, Robert McConnell, John Chiaverini, and Jeremy M. Sage. Scalable loading of a two-dimensional trapped-ion array. *Nature Communications*, 7(1), September 2016.
- [115] Jürgen H. Gross. *Electron Ionization*, pages 193–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [116] M. Johanning, A. Braun, D. Eiteneuer, C. Paape, C. Balzer, W. Neuhauser, and C. Wunderlich. Resonance-enhanced isotope-selective photoionization of ybi for ion trap loading. *Applied Physics B*, 103(2):327–338, April 2011.
- [117] J. B. Naber, L. Torralbo-Campo, T. Hubert, and R. J. C. Spreeuw. Raman transitions between hyperfine clock states in a magnetic trap. *Physical Review A*, 94(1), July 2016.
- [118] Nitzan Akerman, Nir Navon, Shlomi Kotler, Yinnon Glickman, and Roei Ozeri. Universal gate-set for trapped-ion qubits using a narrow linewidth diode laser. *New Journal of Physics*, 17(11):113060, November 2015.

-
- [119] A. Bermudez and X. et al. Xu. Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation. *Physical Review X*, 7(4), December 2017.
 - [120] Ye Wang, Mark Um, Junhua Zhang, Shuoming An, Ming Lyu, Jing-Ning Zhang, L.-M. Duan, Dahyun Yum, and Kihwan Kim. Single-qubit quantum memory exceeding ten-minute coherence time. *Nature Photonics*, 11(10):646–650, September 2017.
 - [121] T.P. Harty, D.T.C. Allcock, C.J. Ballance, L. Guidoni, H.A. Janacek, N.M. Linke, D.N. Stacey, and D.M. Lucas. High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit. *Physical Review Letters*, 113(22), November 2014.
 - [122] T. Ruster, C. T. Schmiegelow, H. Kaufmann, C. Warschburger, F. Schmidt-Kaler, and U. G. Poschinger. A long-lived zeeman trapped-ion qubit. *Applied Physics B*, 122(10), September 2016.
 - [123] A Keselman, Y Glickman, N Akerman, S Kotler, and R Ozeri. High-fidelity state detection and tomography of a single-ion zeeman qubit. *New Journal of Physics*, 13(7):073027, July 2011.
 - [124] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74:4091–4094, May 1995.
 - [125] Anders Sørensen and Klaus Mølmer. Entanglement and quantum computation with ions in thermal motion. *Physical Review A*, 62(2), July 2000.
 - [126] Alexander Erhard, Joel J. Wallman, Lukas Postler, Michael Meth, Roman Stricker, Esteban A. Martinez, Philipp Schindler, Thomas Monz, Joseph Emerson, and Rainer Blatt. Characterizing large-scale quantum computers via cycle benchmarking. *Nature Communications*, 10(1), nov 2019.
 - [127] J.P. Gaebler, T.R. Tan, Y. Lin, Y. Wan, R. Bowler, A.C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D.J. Wineland. High-fidelity universal gate set for be^{3+} ion qubits. *Physical Review Letters*, 117(6), aug 2016.
 - [128] S. L. Todaro, V. B. Verma, K. C. McCormick, D. T. C. Allcock, R. P. Mirin, D. J. Wineland, S. W. Nam, A. C. Wilson, D. Leibfried, and D. H. Slichter. State readout of a trapped ion qubit using a trap-integrated superconducting photon detector. *Phys. Rev. Lett.*, 126:010501, Jan 2021.
 - [129] Stephen Crain, Clinton Cahall, Geert Vrijsen, Emma E. Wollman, Matthew D. Shaw, Varun B. Verma, Sae Woo Nam, and Jungsang Kim. High-speed low-crosstalk detection of a $^{171}\text{yb}^+$ qubit using superconducting nanowire single photon detectors, 2019.
 - [130] A. H. Burrell, D. J. Szwer, S. C. Webster, and D. M. Lucas. Scalable simultaneous multiqubit readout with 99.99% single-shot fidelity. *Phys. Rev. A*, 81:040302, Apr 2010.
 - [131] C. L. Degen, F. Reinhard, and P. Cappellaro. Quantum sensing. *Rev. Mod. Phys.*, 89:035002, Jul 2017.
-

- [132] John M. Martinis, Michel H. Devoret, and John Clarke. Energy-level quantization in the zero-voltage state of a current-biased josephson junction. *Phys. Rev. Lett.*, 55:1543–1546, Oct 1985.
- [133] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Phys. Rev. A*, 76:042319, Oct 2007.
- [134] Olivier Ezratty. Is there a moore’s law for quantum computing?, 2023.
- [135] Yu A Pashkin, O Astafiev, T Yamamoto, Y Nakamura, and JS Tsai. Josephson charge qubits: a brief review. *Quantum Information Processing*, 8:55–80, 2009.
- [136] A Yu Dmitriev and OV Astafiev. A perspective on superconducting flux qubits. *Applied Physics Letters*, 119(8), 2021.
- [137] John M Martinis. Superconducting phase qubits. *Quantum information processing*, 8:81–103, 2009.
- [138] Feng Bao, Hao Deng, Dawei Ding, Ran Gao, Xun Gao, Cupjin Huang, Xun Jiang, Hsiang-Sheng Ku, Zhisheng Li, Xizheng Ma, et al. Fluxonium: an alternative qubit platform for high-fidelity operations. *Physical review letters*, 129(1):010502, 2022.
- [139] Andrea Pasquale, Edoardo Pedicillo, Juan Cereijo, Sergi Ramos-Calderer, Alessandro Candido, Gabriele Palazzo, Rodolfo Carobene, Marco Gobbo, Stavros Efthymiou, Yuanzheng Paul Tan, Ingo Roth, Matteo Robbiati, Jadwiga Wilkens, Alvaro Orgaz-Fuertes, David Fuentes-Ruiz, Andrea Giachero, Frederico Brito, José Ignacio Latorre, and Stefano Carrazza. Qibocal: an open-source framework for calibration of self-hosted quantum devices, 2024.
- [140] Daniel Thomas Sank. Fast, accurate state measurement in superconducting qubits, 2014.
- [141] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer’s guide to superconducting qubits. *Applied Physics Reviews*, 6(2), June 2019.
- [142] J. M. Chow, L. DiCarlo, J. M. Gambetta, F. Motzoi, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Optimized driving of superconducting artificial atoms for improved single-qubit gates. *Physical Review A*, 82(4), October 2010.
- [143] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple pulses for elimination of leakage in weakly nonlinear qubits. *Physical Review Letters*, 103(11), September 2009.
- [144] Ross Shillito, Florian Hopfmueller, Bohdan Kulchytskyy, and Pooya Ronagh. Compact pulse schedules for high-fidelity single-flux quantum qubit control, 2023.
- [145] J C Gallop and B W Petley. Squids and their applications. *Journal of Physics E: Scientific Instruments*, 9(6):417, jun 1976.

-
- [146] J. Majer, J. M. Chow, J. M. Gambetta, Jens Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, A. A. Houck, A. Wallraff, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Coupling superconducting qubits via a cavity bus. *Nature*, 449(7161):443–447, September 2007.
 - [147] Youngkyu Sung, Leon Ding, Jochen Braumüller, Antti Vepsäläinen, Bharath Kannan, Morten Kjaergaard, Ami Greene, Gabriel O. Samach, Chris McNally, David Kim, Alexander Melville, Bethany M. Niedzielski, Mollie E. Schwartz, Jonilyn L. Yoder, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. Realization of high-fidelity cz and zz-free iswap gates with a tunable coupler. *Phys. Rev. X*, 11:021058, Jun 2021.
 - [148] Rui Li, Kentaro Kubo, Yinghao Ho, Zhiguang Yan, Yasunobu Nakamura, and Hayato Goto. Realization of high-fidelity cz gate based on a double-transmon coupler, 2024.
 - [149] Yuan Xu, Ji Chu, Jiahao Yuan, Jiawei Qiu, Yuxuan Zhou, Libo Zhang, Xinsheng Tan, Yang Yu, Song Liu, Jian Li, Fei Yan, and Dapeng Yu. High-fidelity, high-scalability two-qubit gate scheme for superconducting qubits. *Physical Review Letters*, 125(24), December 2020.
 - [150] Chenlu Wang, Xuegang Li, Huikai Xu, Zhiyuan Li, Junhua Wang, Zhen Yang, Zhenyu Mi, Xuehui Liang, Tang Su, Chuhong Yang, Guangyue Wang, Wenyan Wang, Yongchao Li, Mo Chen, Chengyao Li, Kehuan Linghu, Jiaxiu Han, Yingshan Zhang, Yulong Feng, Yu Song, Teng Ma, Jingning Zhang, Ruixia Wang, Peng Zhao, Weiyang Liu, Guangming Xue, Yirong Jin, and Haifeng Yu. Towards practical quantum computers: transmon qubit with a lifetime approaching 0.5 milliseconds. *npj Quantum Information*, 8(1), January 2022.
 - [151] E.T. Jaynes and F.W. Cummings. Comparison of quantum and semiclassical radiation theories with application to the beam maser. *Proceedings of the IEEE*, 51(1):89–109, 1963.
 - [152] Claude Cohen Tannoudji, Jacques Dupont Roc, and Gilbert Grynberg. *A Survey of Some Interaction Processes Between Photons and Atoms*, chapter 2, pages 67–163. John Wiley and Sons, Ltd, 1998.
 - [153] Rukhsan Ul Haq and Keshav Singh. A systematic method for schrieffer-wolff transformation and its generalizations, 2020.
 - [154] Alexandre Blais, Ren-Shou Huang, Andreas Wallraff, S. M. Girvin, and R. J. Schoelkopf. Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation. *Physical Review A*, 69(6), June 2004.
 - [155] François Swiadek, Ross Shillito, Paul Magnard, Ants Remm, Christoph Hellings, Nathan Lacroix, Quentin Ficheux, Dante Colao Zanuz, Graham J. Norris, Alexandre Blais, Sebastian Krinner, and Andreas Wallraff. Enhancing dispersive readout of superconducting qubits through dynamic control of the dispersive shift: Experiment and theory, 2023.
 - [156] T. Walter, P. Kurpiers, S. Gasparinetti, P. Magnard, A. Potočník, Y. Salathé, M. Pechal, M. Mondal, M. Oppliger, C. Eichler, and A. Wallraff. Rapid high-fidelity single-shot dispersive readout of superconducting qubits. *Physical Review Applied*, 7(5), May 2017.
-

- [157] Daniel Sank, Zijun Chen, Mostafa Khezri, J. Kelly, R. Barends, B. Campbell, Y. Chen, B. Chiaro, A. Dunsworth, A. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Mutus, M. Neeley, C. Neill, P.J.J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, T. White, J. Wenner, Alexander N. Korotkov, and John M. Martinis. Measurement-induced state transitions in a superconducting qubit: Beyond the rotating wave approximation. *Physical Review Letters*, 117(19), November 2016.
- [158] Suman Kundu, Nicolas Gheeraert, Sumeru Hazra, Tanay Roy, Kishor V Salunkhe, Meghan P Patankar, and R Vijay. Multiplexed readout of four qubits in 3d circuit qed architecture using a broadband josephson parametric amplifier. *Applied Physics Letters*, 114(17), 2019.
- [159] Hampus Renberg Nilsson. Characterisation of a travelling-wave parametric amplifier for improved qubit measurements, 2019.
- [160] Seungchan Seo, Jiheon Seong, and Joonwoo Bae. Mitigation of crosstalk errors in a quantum measurement and its applications, 2021.
- [161] Gabriele Cenedese, Giuliano Benenti, and Maria Bondani. Correcting coherent errors by random operation on actual quantum hardware. *Entropy*, 25(2):324, 2 2023.
- [162] Daniel Greenbaum and Zachary Dutton. Modeling coherent errors in quantum error correction. *Quantum Science and Technology*, 3(1):015007, 12 2017.
- [163] Florian Venn, Jan Behrends, and Benjamin Béri. Coherent-error threshold for surface codes from majorana delocalization. *Physical Review Letters*, 131(6), 8 2023.
- [164] O. Kern, G. Alber, and D. L. Shepelyansky. Quantum error correction of coherent errors by randomization. *The European Physical Journal D*, 32(1):153–156, 1 2005.
- [165] Sergey Bravyi, Matthias Englbrecht, Robert König, and Nolan Peard. Correcting coherent errors with surface codes. *npj Quantum Information*, 4(1), 10 2018.
- [166] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.
- [167] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correction code, 1996.
- [168] John Preskill. Lecture notes for physics 219: Quantum computation, 01 1999.
- [169] Rene Brun and Fons Rademakers. Root — an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81–86, 1997. New Computing Techniques in Physics Research V.
- [170] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *Journal of High Energy Physics*, 2014(7):79, 2014.

- [171] Martín Abadi, Ashish Agarwal, and Paul Barham et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [172] Cirq, a python framework for creating, editing, and invoking noisy intermediate scale quantum (nisq) circuits, 2018.
- [173] M. Broughton and et al. Tensorflow quantum: A software framework for quantum machine learning, 2020.
- [174] H. Abraham and et al. Qiskit: An open-source framework for quantum computing, 2019.
- [175] Robert S. Smith, Michael J. Curtis, and William J. Zeng. A practical quantum instruction set architecture, 2017.
- [176] Gian Giacomo Guerreschi, Justin Hogaboam, Fabio Baruffa, and Nicolas P D Sawaya. Intel quantum simulator: a cloud-ready high-performance simulator of quantum circuits. *Quantum Science and Technology*, 5(3):034007, May 2020.
- [177] A. Kelly. Simulating quantum computers using opencl, 2018.
- [178] Damian S. Steiger, Thomas Häner, and Matthias Troyer. Projectq: an open source software framework for quantum computing. *Quantum*, 2:49, January 2018.
- [179] V. Bergholm and et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2018.
- [180] J. H. Nielsen, M. Astafev, W. H. Nielsen, D. Vogel, lakhotiahharshit, A. Johnson, A. Hardal, Akshita, sohail chatoor, F. Bonabi, Liang, G. Ungaretti, S. Pauka, T. Morgan, Adriaan, P. Eendebak, B. Nijholt, qSaevan, P. Eendebak, S. Droege, Samantha, J. Darulova, R. van Gulik, N. Pearson, ThorvaldLarsen, and A. Corna. Qcodes/qcodes: Qcodes 0.43.0, 2024.
- [181] M. Rol, C. Dickel, S. Asaad, N. Langford, C. Bultink, R. Sagastizabal, N. Langford, G. de Lange, X. Fu, S. de Jong, F. Luthi, and W. Vlothuizen. Dicarloblab-delft/pycqed_py3: Initial public release, 2016.
- [182] Keysight. Labber, 2022.
- [183] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995.
- [184] T. Ito, N. Kakimura, N. Kamiyama, Y. Kobayashi, and Y. Okamoto. Algorithmic theory of qubit routing, 2023.
- [185] Sengthai Heng, Dongmin Kim, Sovanmonynuth Heng, and Youngsun Han. Decomposition analysis of quantum native gates on various quantum computers. In *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pages 1–3, 2022.
- [186] P. Zhu, S. Zheng, L. Wei, C. Xueyun, Z. Guan, and S. Feng. The complexity of quantum circuit mapping with fixed parameters. *Quantum Information Processing*, 21, 2022.

- [187] T. Itoko, R. Raymond, T. Imamichi, and A. Matsuo. Optimization of quantum circuit mapping using gate transformation and commutation, 2019.
- [188] G. Vidal and C. M. Dawson. Universal quantum circuit for two-qubit transformations with three controlled-not gates. *Phys. Rev. A*, 69:010301, Jan 2004.
- [189] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li. Quantum circuit optimization with deep reinforcement learning, 2021.
- [190] G. Li, Y. Ding, and Y. Xie. Tackling the qubit mapping problem for nisq-era quantum devices, 2019.
- [191] Y. Kharkov, A. Ivanova, E. Mikhantiev, and A. Kotelnikov. Arline benchmarks: Automated benchmarking platform for quantum compilers, 2022.
- [192] Matteo G. Pozzi, Steven J. Herbert, Akash Sengupta, and Robert D. Mullins. Using reinforcement learning to perform qubit routing in quantum compilers, 2020.
- [193] Konstantin F. Pilz, James Sanders, Robi Rahman, and Lennart Heim. Trends in ai supercomputers, 2025.
- [194] S. Watanabe. *Knowing and Guessing: A Quantitative Study of Inference and Information*. Quantitative Study of Inference and Information. Wiley, 1969.
- [195] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [196] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019. Special Issue: Deep Learning in Medical Physics.
- [197] Celia Fernández Madrazo, Ignacio Heredia Cacha, Lara Lloret Iglesias, and Jesús Marco de Lucas. Application of a convolutional neural network for image classification to the analysis of collisions in high energy physics, 2017.
- [198] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih. Machine learning in the search for new fundamental physics, 2021.
- [199] Vasilis Belis, Patrick Odagiu, and Thea Klæboe Aarrestad. Machine learning for anomaly detection in particle physics. *Reviews in Physics*, 12:100091, 2024.
- [200] S. Velliangiri, S. Alagumuthukrishnan, and S Iwin Thankumar josph. A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165:104–111, 2019. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC-DISRUPT - TIV INNOVATION , 2019 November 11-12, 2019.
- [201] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*, 34(6, Part A):2515–2528, 2022.

- [202] Raghav Kansal, Anni Li, Javier Duarte, Nadezda Chernyavskaya, Maurizio Pierini, Breno Orzari, and Thiago Tomei. Evaluating generative models in high energy physics. *Physical Review D*, 107(7), April 2023.
- [203] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning, 2018.
- [204] Alexandre Chenu, Olivier Serris, Olivier Sigaud, and Nicolas Perrin-Gilbert. Single-reset divide and conquer imitation learning, 2024.
- [205] Ying-Chao Hung. A review of monte carlo and quasi-monte carlo sampling techniques algorithms and computational methods > numerical methods statistical and graphical methods of data analysis > sampling. *WIREs Computational Statistics*, 16, 11 2023.
- [206] Vijay Chahar, Sourabh Katoch, and Sumit Chauhan. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 02 2021.
- [207] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson. *The Theory and Practice of Simulated Annealing*, pages 287–319. Springer US, Boston, MA, 2003.
- [208] Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16:100258, 2022.
- [209] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.
- [210] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
- [211] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.
- [212] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width, 2017.
- [213] Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD dissertation, Harvard University, 1974.
- [214] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Blaisdell Publishing Company, Waltham, MA, 1969.
- [215] Stuart E. Dreyfus. The computational solution of optimal control problems with time lag. *IEEE Transactions on Automatic Control*, 18(4):383–385, 1973.
- [216] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [217] A.G. Ivakhnenko, V.G. Lapa, and PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING. *Cybernetic Predicting Devices*. JPRS 37, 803. Joint Publications Research Service [available from the Clearinghouse for Federal Scientific and Technical Information], 1965.

-
- [218] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 18–36, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
 - [219] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization, 2018.
 - [220] Y. Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, and U. Montreal. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, volume 19, 01 2007.
 - [221] Yann Lecun, Bernhard Boser, John Denker, Don Henderson, R. Howard, W. Hubbard, and Larry Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 12 1989.
 - [222] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.
 - [223] Rajdeep Kaur, Rakesh Kumar, and Meenu Gupta. Review on transfer learning for convolutional neural network. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 922–926, 2021.
 - [224] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2020.
 - [225] Ganggang Dong, Guisheng Liao, Hongwei Liu, and Gangyao Kuang. A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images. *IEEE Geoscience and Remote Sensing Magazine*, 6(3):44–68, 2018.
 - [226] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
 - [227] Tilmann Gneiting and Peter Vogel. Receiver operating characteristic (roc) curves, 2018.
 - [228] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589, 2021.
 - [229] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. *Parallel Distributed Process*, 1, 01 1986.
 - [230] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
 - [231] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β -vae, 2018.
-

- [232] Nicolas Vercheval and Aleksandra Pizurica. Hierarchical variational autoencoder for visual counterfactuals, 2021.
- [233] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016.
- [234] Monireh Mohebbi Moghadam, Bahar Boroomand, Mohammad Jalali, Arman Zareian, Alireza DaeiJavad, Mohammad Hossein Manshaei, and Marwan Krunz. Game of gans: Game-theoretical models for generative adversarial networks, 2022.
- [235] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets, 2016.
- [236] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
- [237] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.
- [238] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [239] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.
- [240] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.
- [241] Zike Wu, Pan Zhou, Kenji Kawaguchi, and Hanwang Zhang. Fast diffusion model, 2023.
- [242] Martin L. Puterman. Chapter 8 markov decision processes. In *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, pages 331–434. Elsevier, 1990.
- [243] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7:133653–133667, 2019.
- [244] G. Rummery and Mahesan Niranjana. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.
- [245] Matthias Lehmann. The definitive guide to policy gradients in deep reinforcement learning: Theory, algorithms and implementations, 2024.
- [246] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.
- [247] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

- [248] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020.
- [249] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning, 2023.
- [250] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017.
- [251] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.
- [252] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham M. Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity, 2022.
- [253] Kamil Ciosek. Imitation learning by reinforcement learning, 2022.
- [254] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018.
- [255] Maximilian Schlosshauer. Quantum decoherence. *Physics Reports*, 831:1–57, 2019.
- [256] G. Benenti, G. Casati, and G. Strini. *Principles Of Quantum Computation And Information - Volume II: Basic Tools And Special Topics*. World Scientific Publishing Company, Italy, 2007.
- [257] Joschka Roffe. Quantum error correction: an introductory guide. *Contemporary Physics*, 60(3):226–245, Jul 2019.
- [258] Emanuel Knill, Raymond Laflamme, and Lorenza Viola. Theory of quantum error correction for general noise. *Physical Review Letters*, 84(11):2525–2528, Mar 2000.
- [259] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, Jan 2003.
- [260] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, 9 2012.
- [261] Savvas Varsamopoulos, Koen Bertels, and Carmen Garcia Almudever. Comparing neural network based decoders for the surface code. *IEEE Transactions on Computers*, 69(2):300–311, Feb 2020.
- [262] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, Dec 2012.

-
- [263] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, Mar 2019.
 - [264] Barbara M. Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2):307–346, Apr 2015.
 - [265] Austin G. Fowler. Analytic asymptotic performance of topological codes. *Physical Review A*, 87(4), Apr 2013.
 - [266] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg. Threshold error rates for the toric and surface codes, 2009.
 - [267] Yu Tomita and Krysta M. Svore. Low-distance surface codes under realistic quantum noise. *Physical Review A*, 90(6), Dec 2014.
 - [268] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997.
 - [269] Stephen T. Spitz, Brian Tarasinski, Carlo W. J. Beenakker, and Thomas E. O’Brien. Adaptive weight estimator for quantum error correction in a time-dependent environment. *Advanced Quantum Technologies*, 1(1), Jul 2018.
 - [270] Austin G. Fowler. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $o(1)$ parallel time, 2014.
 - [271] William Cook and André Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.
 - [272] Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Physical Review A*, 90(3), sep 2014.
 - [273] Guillaume Duclos-Cianci and David Poulin. A renormalization group decoding algorithm for topological quantum codes, 2010.
 - [274] Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. *Physical Review Letters*, 104(5), Feb 2010.
 - [275] Michael Herold, Earl T Campbell, Jens Eisert, and Michael J Kastoryano. Cellular-automaton decoders for topological quantum memories. *npj Quantum Information*, 1(1), Oct 2015.
 - [276] Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Physical Review A*, 90(3), Sep 2014.
 - [277] Adrian Hutter, James R. Wootton, and Daniel Loss. Efficient markov chain monte carlo algorithm for the surface code. *Physical Review A*, 89(2), Feb 2014.
 - [278] Savvas Varsamopoulos, Ben Criger, and Koen Bertels. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology*, 3(1), Nov 2017.
 - [279] Austin G. Fowler. Optimal complexity correction of correlated errors in the surface code, 2013.
-

-
- [280] Nicolas Delfosse and Jean-Pierre Tillich. A decoding algorithm for css codes using the x/z correlations. *2014 IEEE International Symposium on Information Theory*, Jun 2014.
 - [281] Stefan Krastanov and Liang Jiang. Deep neural network probabilistic decoder for stabilizer codes. *Scientific Reports*, 7, 05 2017.
 - [282] Johannes Bausch, Andrew Senior, Francisco Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Yuezheng Niu, Sam Blackwell, George Holland, Dvir Kafri, Juan Atalaya, Craig Gidney, Demis Hassabis, Sergio Boixo, Hartmut Neven, and Pushmeet Kohli. Learning high-accuracy error decoding for quantum processors. *Nature*, 635:834–840, 11 2024.
 - [283] Austin G. Fowler, Adam C. Whiteside, Angus L. McInnes, and Alimohammad Rabbani. Topological code autotune. *Physical Review X*, 2(4), oct 2012.
 - [284] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016.
 - [285] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns, 2015.
 - [286] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019.
 - [287] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.
 - [288] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 8 2018.
 - [289] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 12 2020.
 - [290] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Yu Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, T. Huang, E. Jeffrey, Erik Lucero, J. Y. Mutus, O. Naaman, C. Neill, C. Quintana, P. Roushan, Daniel Sank, A. Vainsencher, J. Wenner, T. C. White, S. Boixo, R. Babbush, V. N. Smelyanskiy, H. Neven, and John M. Martinis. Fluctuations of energy-relaxation times in superconducting qubits. *Phys. Rev. Lett.*, 121:090502, 8 2018.
 - [291] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, 2020.
 - [292] Peng Zhao, Kehuan Linghu, Zhiyuan Li, Peng Xu, Ruixia Wang, Guangming Xue, Yirong Jin, and Haifeng Yu. Quantum crosstalk analysis for simultaneous gate operations on superconducting qubits. *PRX Quantum*, 3:020301, 4 2022.

-
- [293] Irina Heinz and Guido Burkard. Crosstalk analysis for single-qubit and two-qubit gates in spin qubit arrays. *Phys. Rev. B*, 104:045420, 7 2021.
 - [294] Tirthak Patel, Abhay Potharaju, Baolin Li, Rohan Basu Roy, and Devesh Tiwari. Experimental evaluation of nisq quantum computers: Error measurement, characterization, and implications. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2020.
 - [295] Swamit S. Tannu and Moinuddin K. Qureshi. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '52*, page 279–290, New York, NY, USA, 2019. Association for Computing Machinery.
 - [296] Bo Yang, Rudy Raymond, and Shumpei Uno. Efficient quantum readout-error mitigation for sparse measurement outcomes of near-term quantum devices. *Phys. Rev. A*, 106:012423, Jul 2022.
 - [297] Thomas Schuster, Chao Yin, Xun Gao, and Norman Y. Yao. A polynomial-time classical algorithm for noisy quantum circuits, 2024.
 - [298] E Knill. Quantum computing with realistically noisy devices. *Nature*, 434:39–44, 04 2005.
 - [299] Kosuke Fukui, Akihisa Tomita, Atsushi Okamoto, and Keisuke Fujii. High-threshold fault-tolerant quantum computation with analog quantum error correction. *Phys. Rev. X*, 8:021054, 5 2018.
 - [300] Savvas Varsamopoulos, Ben Criger, and Koen Bertels. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology*, 3(1):015004, 11 2017.
 - [301] Jaeho Choi and Joongheon Kim. A tutorial on quantum approximate optimization algorithm (qaoa): Fundamentals and applications. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 138–142, 2019.
 - [302] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, and Lukasz Cincio. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 8 2021.
 - [303] Daniel J. Egger, Chiara Capecchi, Bibek Pokharel, Panagiotis Kl. Barkoutsos, Laurin E. Fischer, Leonardo Guidoni, and Ivano Tavernelli. Pulse variational quantum eigensolver on cross-resonance-based hardware. *Physical Review Research*, 5(3):033159, September 2023.
 - [304] Matteo Robbiati, Juan M. Cruz-Martinez, and Stefano Carrazza. Determining probability density functions with adiabatic quantum computing, 2023. 7 pages, 3 figures.
 - [305] Matteo Robbiati, Stavros Efthymiou, Andrea Pasquale, and Stefano Carrazza. A quantum analytical adam descent through parameter shift rule using qibo, 2022.
 - [306] Juan M Cruz-Martinez, Matteo Robbiati, and Stefano Carrazza. Multi-variable integration with a variational quantum circuit. *Quantum Science and Technology*, 9(3):035053, July 2024.
-

- [307] G. Ravi, K. N. Smith, P. Gokhale, and F. T. Chong. Quantum computing in the cloud: Analyzing job and machine characteristics. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*, pages 39–50, Los Alamitos, CA, USA, nov 2021. IEEE Computer Society.
- [308] qibolab_platforms_qrc, 2023.
- [309] Stavros Efthymiou, Sergi Ramos-Calderer, Carlos Bravo-Prieto, Adrián Pérez-Salinas, Diego García-Martín, Artur Garcia-Saez, José Ignacio Latorre, and Stefano Carrazza. Qibo: a framework for quantum simulation with hardware acceleration. *Quantum Science and Technology*, 7(1):015018, dec 2021.
- [310] Stavros Efthymiou, Marco Lazzarin, Andrea Pasquale, and Stefano Carrazza. Quantum simulation with just-in-time compilation. *Quantum*, 6:814, sep 2022.
- [311] Edoardo Pedicillo, Alessandro Candido, Stavros Efthymiou, Hayk Sargsyan, Yuanzheng Paul Tan, Juan Cereijo, Jun Yong Khoo, Andrea Pasquale, Matteo Robbiati, and Stefano Carrazza. An open-source framework for quantum hardware control, 2024.
- [312] Andrea Pasquale, Stavros Efthymiou, Sergi Ramos-Calderer, Jadwiga Wilkens, Ingo Roth, and Stefano Carrazza. Towards an open-source framework to perform quantum calibration and characterization, 2023.
- [313] Antoine Brilliant, Peter Groszkowski, Alireza Seif, Jens Koch, and Aashish Clerk. Randomized benchmarking with non-markovian noise and realistic finite-time gates, 2025.
- [314] Emilio Onorati, Tamara Kohler, and Toby S. Cubitt. Fitting quantum noise models to tomography data. *Quantum*, 7:1197, December 2023.
- [315] Kaixin Huang, Dmitry Farfurnik, Alireza Seif, Mohammad Hafezi, and Yi-Kai Liu. Random pulse sequences for qubit noise spectroscopy, 2025.
- [316] Senrui Chen, Yunchao Liu, Matthew Otten, Alireza Seif, Bill Fefferman, and Liang Jiang. The learnability of pauli noise. *Nature Communications*, 14(1), January 2023.
- [317] Moein Malekakhlagh, Alireza Seif, Daniel Puzzuoli, Luke C. G. Govia, and Ewout van den Berg. Efficient lindblad synthesis for noise model construction, 2025.
- [318] Ewout van den Berg and Pawel Wocjan. Techniques for learning sparse pauli-lindblad noise models. *Quantum*, 8:1556, December 2024.
- [319] Stefano Mangini, Marco Cattaneo, Daniel Cavalcanti, Sergei Filippov, Matteo A. C. Rossi, and Guillermo García-Pérez. Tensor network noise characterization for near-term quantum computers. *Physical Review Research*, 6(3), August 2024.
- [320] Ettore Canonici, Stefano Martina, Riccardo Mengoni, Daniele Ottaviani, and Filippo Caruso. Machine learning based noise characterization and correction on neutral atoms nisc devices. *Advanced Quantum Technologies*, 7(1), November 2023.

-
- [321] Cole Hoffer. *Superconducting qubit readout pulse optimization using deep reinforcement learning*. PhD thesis, 01 2021.
 - [322] Aniket Chatterjee, Jonathan Schwinger, and Yvonne Y. Gao. Enhanced qubit readout via reinforcement learning. *Phys. Rev. Appl.*, 23:054057, May 2025.
 - [323] T. Crosta, L. Rebón, F. Vilarino, J. M. Matera, and M. Bilkis. Automatic re-calibration of quantum devices by reinforcement learning, 2024.
 - [324] Kevin Young. Reinforcement learning for calibrating quantum processors. In *APS March Meeting Abstracts*, volume 2023 of *APS Meeting Abstracts*, page B72.005, January 2023.
 - [325] Ho Nam Nguyen, Felix Motzoi, Mekena Metcalf, K Birgitta Whaley, Marin Bukov, and Markus Schmitt. Reinforcement learning pulses for transmon qubit entangling gates. *Machine Learning: Science and Technology*, 5(2):025066, June 2024.
 - [326] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J. Briegel, and Nicolai Friis. Optimizing quantum error correction codes with reinforcement learning. *Quantum*, 3:215, December 2019.
 - [327] Jan Olle, Remmy Zen, Matteo Puviani, and Florian Marquardt. Simultaneous discovery of quantum error correction codes and encoders with a noise-aware reinforcement learning agent, 2024.
 - [328] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.
 - [329] Yeong-Cherng Liang, Yu-Hao Yeh, Paulo E M F Mendonça, Run Yan Teh, Margaret D Reid, and Peter D Drummond. Quantum fidelity measures for mixed states. *Reports on Progress in Physics*, 82(7):076001, June 2019.
 - [330] Qin Huang and Marco Merkli. Qubit dynamics with classical noise. *Physics Open*, 5:100043, 2020.
 - [331] Matthias Christandl and Renato Renner. Reliable quantum state tomography. *Phys. Rev. Lett.*, 109:120403, 9 2012.
 - [332] G.I. Struchalin, Ya. A. Zagorovskii, E.V. Kovlakov, S.S. Straupe, and S.P. Kulik. Experimental estimation of quantum state properties from classical shadows. *PRX Quantum*, 2:010307, 1 2021.
 - [333] Jens Eisert, Dominik Hangleiter, Nathan Walk, Ingo Roth, Damian Markham, Rhea Parekh, Ulysse Chabaud, and Elham Kashefi. Quantum certification and benchmarking. *Nature Reviews Physics*, 2(7):382–390, 6 2020.
 - [334] Senrui Chen, Wenjun Yu, Pei Zeng, and Steven T. Flammia. Robust shadow estimation. *PRX Quantum*, 2:030348, 9 2021.
-

- [335] Alan C. Santos. O computador quântico da IBM e o IBM quantum experience. *Revista Brasileira de Ensino de Física*, 39(1), 9 2016.
- [336] Jonas Helsen, Xiao Xue, Lieven M. K. Vandersypen, and Stephanie Wehner. A new class of efficient randomized benchmarking protocols. *npj Quantum Information*, 5:71, 8 2019.
- [337] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Randomized benchmarking of quantum gates. *Phys. Rev. A*, 77:012307, 1 2008.
- [338] Sergey Bravyi and Dmitri Maslov. Hadamard-Free Circuits Expose the Structure of the Clifford Group. *IEEE Transactions on Information Theory*, 67(7):4546–4563, July 2021.
- [339] M. H. Devoret and R. J. Schoelkopf. Superconducting circuits for quantum information: An outlook. *Science*, 339(6124):1169–1174, 2013.
- [340] Benjamin Nachman, Miroslav Urbanek, Wibe A. De Jong, and Christian W. Bauer. Unfolding quantum computer readout noise. *npj Quantum Information*, 6(1):84, September 2020.
- [341] Ellis Wilson, Sudhakar Singh, and Frank Mueller. Just-in-time quantum circuit transpilation reduces noise. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 345–355, 2020.
- [342] Nouhaila Innan, Owais Ishtiaq Siddiqui, Shivang Arora, Tamojit Ghosh, Yasemin Poyraz Koçak, Dominic Paragas, Abdullah Al Omar Galib, Muhammad Al-Zafar Khan, and Mohamed Bennai. Quantum state tomography using quantum machine learning. *Quantum Machine Intelligence*, 6(1), May 2024.
- [343] Tobias Schmale, Moritz Reh, and Martin Gärttner. Efficient quantum state tomography with convolutional neural networks. *npj Quantum Information*, 8(1), September 2022.
- [344] Abhinav Anand, Philipp Schleich, Sumner Alperin-Lea, Phillip W. K. Jensen, Sukin Sim, Manuel Díaz-Tinoco, Jakob S. Kottmann, Matthias Degroote, Artur F. Izmaylov, and Alán Aspuru-Guzik. A quantum computing view on unitary coupled cluster theory. *Chemical Society Reviews*, 51(5):1659–1684, 2022.
- [345] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O’Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), July 2014.
- [346] Davide Materia, Leonardo Ratini, Celestino Angeli, and Leonardo Guidoni. Quantum information driven ansatz (qida): shallow-depth empirical quantum circuits from quantum chemistry, 2023.
- [347] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.
- [348] Harper R. Grimsley, Sophia E. Economou, Edwin Barnes, and Nicholas J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10(1), July 2019.

-
- [349] Gloria Turati, Maurizio Ferrari Dacrema, and Paolo Cremonesi. Benchmarking adaptative variational quantum algorithms on qubo instances, 2023.
- [350] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A*, 103:032430, 3 2021.
- [351] Phuc Le, Abdullah Ilyasu, Fangyan Dong, and Kaoru Hirota. A flexible representation of quantum images for polynomial preparation, image compression and processing operations, quantum inf. *Quantum Information Processing*, 10:63–84, 02 2011.
- [352] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, February 2020.
- [353] Leo Sünkel, Darya Martyniuk, Denny Mattern, Johannes Jung, and Adrian Paschke. Ga4qco: Genetic algorithm for quantum circuit optimization, 2023.
- [354] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3), September 2018.
- [355] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), March 2019.
- [356] Leonardo Banchi and Gavin E. Crooks. Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule. *Quantum*, 5:386, January 2021.
- [357] Hanrui Wang, Jiaqi Gu, Yongshan Ding, Zirui Li, Frederic T. Chong, David Z. Pan, and Song Han. Quantumnat: quantum noise-aware training with noise injection, quantization and normalization. In *Proceedings of the 59th ACM/IEEE Design Automation Conference, DAC '22*, page 1–6, New York, NY, USA, 2022. Association for Computing Machinery.
- [358] Hanrui Wang, Yilian Liu, Pengyu Liu, Jiaqi Gu, Zirui Li, Zhiding Liang, Jinglei Cheng, Yongshan Ding, Xuehai Qian, Yiyu Shi, David Z. Pan, Frederic T. Chong, and Song Han. Robuststate: Boosting fidelity of quantum state preparation via noise-aware variational training, 2023.
- [359] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), November 2018.
- [360] Javier Rivera-Dean, Patrick Huembeli, Antonio Acín, and Joseph Bowles. Avoiding local minima in variational quantum algorithms with neural networks, 2021.
- [361] Jack Cunningham and Jun Zhuang. Investigating and mitigating barren plateaus in variational quantum circuits: A survey, 2024.
- [362] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), March 2021.

- [363] Andrea Skolik, Jarrod R. McClean, Masoud Mohseni, Patrick van der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3(1), January 2021.
- [364] Muhammad Kashif, Muhammad Rashid, Saif Al-Kuwari, and Muhammad Shafique. Alleviating barren plateaus in parameterized quantum machine learning circuits: Investigating advanced parameter initialization strategies, 2023.
- [365] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, December 2019.
- [366] Xia Liu, Geng Liu, Hao-Kai Zhang, Jiabin Huang, and Xin Wang. Mitigating barren plateaus of variational quantum eigensolvers. *IEEE Transactions on Quantum Engineering*, 5:1–19, 2024.
- [367] Chae-Yeun Park, Minhyeok Kang, and Joonsuk Huh. Hardware-efficient ansatz without barren plateaus in any depth, 2024.
- [368] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J. Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1), January 2022.
- [369] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. Entanglement-induced barren plateaus. *PRX Quantum*, 2:040316, Oct 2021.
- [370] Christopher Havenstein, Damarcus Thomas, and Swami Chandrasekaran. Comparisons of performance between quantum and classical machine learning. *SMU Data Science Review*, 1(4):11, 2018.
- [371] Sofiene Jerbi, Lukas J. Fiderer, Hendrik Poulsen Nautrup, Jonas M. Kübler, Hans J. Briegel, and Vedran Dunjko. Quantum machine learning beyond kernel methods. *Nature Communications*, 14(1), January 2023.
- [372] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean. Power of data in quantum machine learning. *Nature Communications*, 12(1), May 2021.
- [373] Peiyong Wang, Casey. R. Myers, Lloyd C. L. Hollenberg, and Udaya Parampalli. Let the quantum creep in: Designing quantum neural network models by gradually swapping out classical components, 2024.
- [374] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Information-theoretic bounds on quantum advantage in machine learning. *Physical Review Letters*, 126(19), May 2021.
- [375] Pablo Bermejo, Paolo Braccia, Manuel S. Rudolph, Zoë Holmes, Lukasz Cincio, and M. Cerezo. Quantum convolutional neural networks are (effectively) classically simulable, 2024.

-
- [376] Matthias C. Caro, Hsin-Yuan Huang, M. Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J. Coles. Generalization in quantum machine learning from few training data. *Nature Communications*, 13(1), August 2022.
- [377] Barry D. Reese, Marek Kowalik, Christian Metzl, Christian Bauckhage, and Eldar Sultanow. Predict better with less training data using a qnn, 2022.
- [378] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), October 2009.
- [379] Hiroshi Ohno. A quantum algorithm of k-means toward practical use. *Quantum Information Processing*, 21, 04 2022.
- [380] Stephen DiAdamo, Corey O’Meara, Giorgio Cortiana, and Juan Bernabé-Moreno. Practical quantum k-means clustering: Performance analysis and applications in energy grid classification. *IEEE Transactions on Quantum Engineering*, 3:1–16, 2022.
- [381] Samuel Bosch. *Quantum Algorithms for Linear Algebra and Optimization*. PhD thesis, 08 2020.
- [382] Adam Callison and Nicholas Chancellor. Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Physical Review A*, 106(1), July 2022.
- [383] Jinkai Tian, Xiaoyu Sun, Yuxuan Du, Shanshan Zhao, Qing Liu, Kaining Zhang, Wei Yi, Wanrong Huang, Chaoyue Wang, Xingyao Wu, Min-Hsiu Hsieh, Tongliang Liu, Wenjing Yang, and Dacheng Tao. Recent advances for quantum neural networks in generative learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12321–12340, 2023.
- [384] Yuxuan Du, Zhuozhuo Tu, Bujiao Wu, Xiao Yuan, and Dacheng Tao. Power of quantum generative learning, 2022.
- [385] Xun Gao, Eric R. Anschuetz, Sheng-Tao Wang, J. Ignacio Cirac, and Mikhail D. Lukin. Enhancing generative models via quantum correlations. *Physical Review X*, 12(2), May 2022.
- [386] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. Protocols for trainable and differentiable quantum generative modeling. *Phys. Rev. Res.*, 6:033291, Sep 2024.
- [387] Louis Schatzki, Andrew Arrasmith, Patrick J. Coles, and M. Cerezo. Entangled datasets for quantum machine learning, 2021.
- [388] Carlos Bravo-Prieto. Quantum autoencoders with enhanced data encoding. *Machine Learning: Science and Technology*, 2(3):035028, 7 2021.
- [389] Alessio Gianelle, Patrick Koppenburg, Donatella Lucchesi, Davide Nicotra, Eduardo Rodrigues, Lorenzo Sestini, Jacco de Vries, and Davide Zuliani. Quantum machine learning for b-jet charge identification. *Journal of High Energy Physics*, 2022(8), 8 2022.
- [390] Sulaiman Alvi, Christian Bauer, and Benjamin Nachman. Quantum anomaly detection for collider physics, 2022.

- [391] Christian W. Bauer, Zohreh Davoudi, A. Baha Balantekin, Tanmoy Bhattacharya, Marcela Carena, Wibe A. de Jong, Patrick Draper, Aida El-Khadra, Nate Gemelke, Masanori Hanada, Dmitri Kharzeev, Henry Lamm, Ying-Ying Li, Junyu Liu, Mikhail Lukin, Yannick Meurice, Christopher Monroe, Benjamin Nachman, Guido Pagano, John Preskill, Enrico Rinaldi, Alessandro Roggero, David I. Santiago, Martin J. Savage, Irfan Siddiqi, George Siopsis, David Van Zanten, Nathan Wiebe, Yukari Yamauchi, Kübra Yeter-Aydeniz, and Silvia Zorzetti. Quantum simulation for high energy physics, 2022.
- [392] A. Mott, J. Job, J. Vlimant, D. Lidar, and M. Spiropulu. Solving a Higgs optimization problem with quantum annealing for machine learning. *Nature*, 550(7676):375–379, 10 2017.
- [393] A. Blance and M. Spannowsky. Quantum machine learning for particle physics using a variational quantum classifier. *Journal of High Energy Physics*, 2021(2), 2 2021.
- [394] K. Terashi, M. Kaneda, T. Kishimoto, M. Saito, R. Sawada, and J. Tanaka. Event classification with quantum machine learning in high-energy physics. *Computing and Software for Big Science*, 5(1), 1 2021.
- [395] Samuel Yen-Chi Chen, Tzu-Chieh Wei, Chao Zhang, Haiwang Yu, and Shinjae Yoo. Quantum convolutional neural networks for high energy physics data analysis, 2020.
- [396] Sau Lan Wu, J. Chan, W. Guan, S. Sun, A. Wang, C. Zhou, M. Livny, F. Carminati, A. Di Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, Y. Chen, S., S. Yoo, and T. Wei. Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits. *Journal of Physics G: Nuclear and Particle Physics*, 48(12):125003, 10 2021.
- [397] Sau Lan Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang, M. Livny, J. Glick, P. Kl. Barkoutsos, S. Woerner, I Tavernelli, F. Carminati, A. Di Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, S. Y. Chen, S. Yoo, and T. Wei. Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC. *Physical Review Research*, 3(3), 9 2021.
- [398] Carlos Bravo-Prieto, Julien Baglio, Marco Cè , Anthony Francis, Dorota M. Grabowska, and Stefano Carrazza. Style-based quantum generative adversarial networks for monte carlo events. *Quantum*, 6:777, 8 2022.
- [399] Vishal S. Ngairangbam, Michael Spannowsky, and Michihisa Takeuchi. Anomaly detection in high-energy physics using a quantum autoencoder. *Physical Review D*, 105(9), 5 2022.
- [400] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.*, 94:015004, 2 2022.
- [401] Brian Coyle. Machine learning applications for noisy intermediate-scale quantum computers, 2022.

-
- [402] Gennaro De Luca. A survey of nisq era hybrid quantum-classical machine learning research. *Journal of Artificial Intelligence and Technology*, 2(1):9–15, 12 2021.
 - [403] <https://quantum-computing.ibm.com/>, 2021.
 - [404] A. Falkowski, J. A. T. Ruderman, V. T. Volansky, and J. Zupan. Hidden higgs decaying to lepton jets. *J. High Energ. Phys.*, 2010(77), 2010.
 - [405] M. J. Strassler and K. M. Zurek. Echoes of a hidden valley at hadron colliders. *J. High Energ. Phys.*, 651(5-6), 2007.
 - [406] ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003, 2008.
 - [407] ATLAS Collaboration. Standalone vertex finding in the ATLAS muon spectrometer. *JINST*, 9:P02001, 2014.
 - [408] ATLAS Collaboration. Search for long-lived neutral particles produced in pp collisions at $\sqrt{s} = 13$ TeV decaying into displaced hadronic jets in the atlas inner detector and muon spectrometer. *Phys. Rev. D*, 101:052013, 3 2020.
 - [409] Antonio Acín, Immanuel Bloch, Harry Buhrman, Tommaso Calarco, Christopher Eichler, Jens Eisert, Daniel Esteve, Nicolas Gisin, Steffen J Glaser, Fedor Jelezko, Stefan Kuhr, Maciej Lewenstein, Max F Riedel, Piet O Schmidt, Rob Thew, Andreas Wallraff, Ian Walmsley, and Frank K Wilhelm. The quantum technologies roadmap: a european community view. *New Journal of Physics*, 20(8):080201, 8 2018.
 - [410] Sukhpal Singh Gill, Adarsh Kumar, Harvinder Singh, Manmeet Singh, Kamalpreet Kaur, Muhammad Usman, and Rajkumar Buyya. Quantum computing: A taxonomy, systematic review and future directions, 2020.
 - [411] Rainer Kaltenbaek, Antonio Acin, Laszlo Bacsardi, and Paolo Bianco et al. Quantum technologies in space. *Experimental Astronomy*, 51(3):1677–1694, 6 2021.
 - [412] Bass Steven D. and Zohar Erez. Quantum technologies in particle physics. *Phil. Trans. R. Soc.*, A(380), 12 2021.
 - [413] M. Doser, E. Auffray, F.M. Brunbauer, I. Frank, H. Hillemanns, G. Orlandini, and G. Koronakov. Quantum systems for enhanced high energy particle physics detectors. *Frontiers in Physics*, 10, 2022.
 - [414] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.
 - [415] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.
 - [416] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

- [417] Sau Lan Wu and Shinjae Yoo. Challenges and opportunities in quantum machine learning for high-energy physics. *Nature Reviews Physics*, 4(3):143–144, 3 2022.
- [418] Daniel J. Egger, Chiara Capecchi, Bibek Pokharel, Panagiotis Kl. Barkoutsos, Laurin E. Fischer, Leonardo Guidoni, and Ivano Tavernelli. A study of the pulse-based variational quantum eigensolver on cross-resonance based hardware, 2023.
- [419] ShiJie Wei, YanHu Chen, ZengRong Zhou, and GuiLu Long. A quantum convolutional neural network on nisq devices, 2021.
- [420] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.
- [421] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, Hossein Sadeghi, and Mohammad H Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1):014001, 2018.
- [422] Marco Parigi, Stefano Martina, and Filippo Caruso. Quantum-noise-driven generative diffusion models. *arXiv preprint arXiv:2308.12013*, 2023.
- [423] Bingzhi Zhang, Peng Xu, Xiaohui Chen, and Quntao Zhuang. Generative quantum machine learning via denoising diffusion probabilistic models. *arXiv preprint arXiv:2310.05866*, 2023.
- [424] Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov. Synthesis of quantum logic circuits. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference, ASP-DAC '05*, page 272–275, New York, NY, USA, 2005. Association for Computing Machinery.
- [425] John A. Cortese and Timothy M. Braje. Loading classical data into a quantum computer, 2018.
- [426] Adriano Barenco, Andre Berthiaume, David Deutsch, Artur Ekert, Richard Jozsa, and Chiara Macchiavello. Stabilization of quantum computations by symmetrization. *SIAM Journal on Computing*, 26(5):1541–1557, 1997.
- [427] Maria Schuld and Francesco Petruccione. *Machine learning with quantum computers*. Springer, 2021.
- [428] Michael A Nielsen. The entanglement fidelity and quantum error correction. *arXiv preprint quant-ph/9606012*, 1996.
- [429] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [430] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.
- [431] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

-
- [432] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.
 - [433] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
 - [434] Lorenzo Luzi, Carlos Ortiz Marrero, Nile Wymar, Richard G Baraniuk, and Michael J Henry. Evaluating generative networks using gaussian mixtures of image features. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 279–288, 2023.
 - [435] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.
 - [436] Malcolm Carroll, Sami Rosenblatt, Petar Jurcevic, Isaac Lauer, and Abhinav Kandala. Dynamics of superconducting qubit relaxation times, 2022.
 - [437] David Kremer, Victor Villar, Hanhee Paik, Ivan Duran, Ismael Faro, and Juan Cruz-Benito. Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning, 2024.
 - [438] Giovanni Acampora and Roberto Schiattarella. Deep neural networks for quantum circuit mapping. *Neural Computing and Applications*, 33, 10 2021.
 - [439] M. Kounalakis, C. Dickel, A. Bruno, N. K. Langford, and G. A. Steele. Tuneable hopping and nonlinear cross-kerr interactions in a high-coherence superconducting circuit. *npj Quantum Information*, 4(1), August 2018.
 - [440] S. Danilin and M. Weides. Quantum sensing with superconducting circuits, 2021.
 - [441] Sergey Danilin, Nicholas Nugent, and Martin Weides. Quantum sensing with tunable superconducting qubits: optimization and speed-up, 2024.
 - [442] Alen Senanian, Sridhar Prabhu, Vladimir Kremenetski, Saswata Roy, Yingkang Cao, Jeremy Kline, Tatsuhiro Onodera, Logan G. Wright, Xiaodi Wu, Valla Fatemi, and Peter L. McMahon. Microwave signal processing using an analog quantum reservoir computer. *Nature Communications*, 15(1), August 2024.
 - [443] Akash V. Dixit, Srivatsan Chakram, Kevin He, Ankur Agrawal, Ravi K. Naik, David I. Schuster, and Aaron Chou. Searching for dark matter with a superconducting qubit. *Phys. Rev. Lett.*, 126:141302, Apr 2021.
 - [444] C. Braggio, L. Balembois, R. Di Vora, Z. Wang, J. Travesedo, L. Pallegoix, G. Carugno, A. Ortolan, G. Ruoso, U. Gambardella, D. D’Agostino, P. Bertet, and E. Flurin. Quantum-enhanced sensing of axion dark matter with a transmon-based single microwave photon counter. *Phys. Rev. X*, 15:021031, Apr 2025.
-

- [445] R. Moretti, H. Atse Corti, D. Labranca, F. Ahrens, G. Avallone, D. Babusci, L. Banchi, C. Barone, M. M. Beretta, M. Borghesi, B. Buonomo, E. Calore, G. Carapella, F. Chiarello, A. Cian, A. Cidronali, F. Costa, A. Cuccoli, A. D’Elia, D. Di Gioacchino, S. Di Pascoli, P. Falferi, M. Fanciulli, M. Faverzani, G. Felici, E. Ferri, G. Filatrella, L. G. Foggetta, C. Gatti, A. Giachero, F. Giazotto, D. Giubertoni, V. Granata, C. Guarcello, G. Lamanna, C. Ligi, G. Maccarrone, M. Macucci, G. Manara, F. Mantegazzini, P. Marconcini, B. Margesin, F. Mattioli, A. Miola, A. Nucciotti, L. Origo, S. Pagano, F. Paolucci, L. Piersanti, A. Rettaroli, S. Sanguinetti, S. F. Schifano, P. Spagnolo, S. Tocci, A. Toncelli, G. Torrioli, and A. Vinante. Design and simulation of a transmon qubit chip for axion detection. *IEEE Transactions on Applied Superconductivity*, 34(3):1–5, 2024.
- [446] Danilo Labranca, Hervé Atsè Corti, Leonardo Banchi, Alessandro Cidronali, Simone Felicetti, Claudio Gatti, Andrea Giachero, and Angelo Nucciotti. First design of a superconducting qubit for the qub-it experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1046:167716, January 2023.
- [447] B.D. Josephson. Possible new effects in superconductive tunnelling. *Physics Letters*, 1(7):251–253, 1962.
- [448] C.A. Hamilton, C.J. Burroughs, and S.P. Benz. Josephson voltage standard-a review. *IEEE Transactions on Applied Superconductivity*, 7(2):3756–3761, 1997.
- [449] John Clarke. The josephson effect and e/h . *American Journal of Physics*, 38(9):1071–1095, 1970.
- [450] Joseph Emerson, Robert Alicki, and Karol Życzkowski. Scalable noise estimation with random unitary operators. *Journal of Optics B: Quantum and Semiclassical Optics*, 7(10):S347, 9 2005.
- [451] Markus Heinrich, Martin Kliesch, and Ingo Roth. Randomized benchmarking with random quantum circuits, 2023.
- [452] Andrew Steane. A tutorial on quantum error correction, 01 2006.
- [453] Imrus Salehin and Dae-Ki Kang. A review on dropout regularization approaches for deep neural networks within the scholarly domain. *Electronics*, 12(14), 2023.

Disclaimer

This thesis is the original work of the author. However, it should be noted that AI-based tools were utilized in certain sections for grammar correction and readability improvement. The use of these tools was intended to enhance the clarity and coherence of the text, without altering the originality and integrity of the research presented. All findings, interpretations, and conclusions expressed in this thesis are those of the author.

Appendix A

No Cloning Theorem

The *no cloning theorem* states that it is not possible to duplicate an a-priori unknown quantum state. The proof follows from the fact that all transformations of a quantum state are unitary. Suppose absurdly, that a unitary transformation U exists such that, given an unknown state $|\psi\rangle_a$, copies it to an auxiliary state $|e\rangle_b$:

$$U(|\psi\rangle_a |e\rangle_b) = |\psi\rangle_a |\psi\rangle_b$$

For every $|\psi\rangle$ and $|\phi\rangle$, from the unitarity of U , follows that:

$$\langle\phi|\psi\rangle_a \langle e|e\rangle_b = (\langle\phi|_a \langle e|_b) U^\dagger U (|\psi\rangle_a |e\rangle_b) = \langle\phi|\psi\rangle^2$$

As $\langle e|e\rangle = 1$, the following equality is obtained:

$$|\langle\phi|\psi\rangle| = |\langle\phi|\psi\rangle|^2$$

This equality is only satisfied if ψ and ϕ are identical or orthogonal, but is not valid for arbitrary states. This contradiction concludes the proof.

Appendix B

Josephson Junction

A Josephson junction is a system composed of two superconducting regions, typically made of niobium or aluminum, separated by a thin insulating layer, commonly aluminum oxide, approximately 1nm thick. Within the superconducting regions, Cooper pairs form a Bose-Einstein condensate, allowing them to be described by a single wavefunction. A schematic of this system, showing the wavefunctions of the Cooper pairs, is presented in Figure B.1.

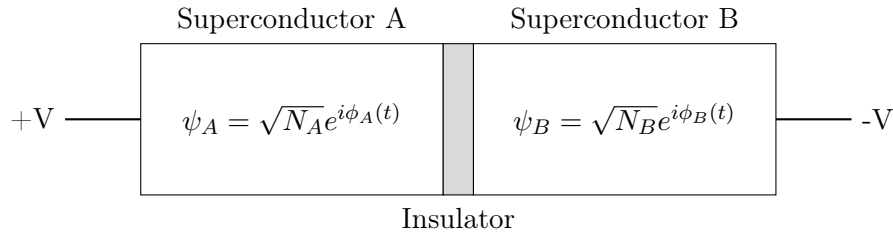


Figure B.1: Schematic representation of a Josephson junction. Two superconductors, labeled S_A and S_B , are separated by a thin insulating barrier. The wavefunctions $\psi_A = \sqrt{N_A}e^{i\phi_A(t)}$ and $\psi_B = \sqrt{N_B}e^{i\phi_B(t)}$ describe the state of the Cooper pairs within each superconductor, where N_A and N_B are the numbers of electron Cooper pairs in each superconductive region. An applied voltage difference across the junction, indicated by $+V$ and $-V$, influences the phase difference between the superconductors.

When a voltage difference is applied across the junction, macroscopic quantum phenomena, known as the Josephson effect, can be observed. The prediction of this phenomenon by B. Josephson in 1962 [447] earned him the Nobel Prize in 1973. In the presence of an applied voltage difference of $2V$, as shown in Figure B.1, the time evolution of the wavefunctions is described by the Schrödinger equation:

$$\begin{aligned} i\hbar \frac{\partial \psi_A}{\partial t} &= eV\psi_A + K\psi_B, \\ i\hbar \frac{\partial \psi_B}{\partial t} &= -eV\psi_B + K\psi_A, \end{aligned}$$

where K , the coupling energy, quantifies the strength of the tunneling of Cooper pairs between the two superconductors. This parameter decreases exponentially with the thickness of the insulating barrier. By substituting the wavefunctions depicted in Figure B.1 into the previous equations, in the limit of balanced superconductors $N_A \simeq N_B$, it is possible to describe the current passing through the junction I as a function of the relative phase of the electrons' wavefunction $\phi = \phi_A - \phi_B$. Moreover, ϕ depends on the voltage difference applied at the edges of the junction. These

dependencies are known as the Josephson equations:

$$I(t) = I_c \sin(\phi(t)), \quad (\text{B.1})$$

$$\frac{\partial \phi}{\partial t} = \frac{2eV(t)}{\hbar}. \quad (\text{B.2})$$

In Equation B.1, we have introduced the critical current I_c , which is the maximum current that can pass through the junction in the superconducting regime (without any resistive loss). This parameter depends, in good approximation, on the coupling energy K :

$$I_c = \frac{2e}{\hbar} K.$$

The Josephson equations, as mentioned previously, give rise to macroscopic phenomena that cannot be explained by classical mechanics. In particular, we will examine the cases of a zero voltage applied to the junction and a constant voltage. If $V = 0$, a non-zero current is observed in the junction:

$$I(t) = I_c \sin(\phi_0).$$

This current arises from the tunneling effect of Cooper pairs through the insulator. The current intensity depends on the initial relative phase of the electrons' wavefunction, ϕ_0 . In the case of a constant voltage, we obtain from the Josephson equations:

$$I(t) = I_c \sin\left(\phi_0 + \frac{2eV}{\hbar}t\right).$$

Thus, a rapidly oscillating current will be observed in the junction. This effect allows for a precise definition of the voltage by relating it to the frequency of the alternating current in the junction [448] and for the measurement of $\frac{e}{h}$ [449]. We will now derive two important parameters of the Josephson junction, the inductance and the energy. To compute the inductance we introduce the reduced magnetic flux quantum $\phi_0 = \frac{\hbar}{2e}$ into Equation B.2, yielding:

$$V = \phi_0 \frac{\partial \phi}{\partial t}.$$

This relation is formally similar to Faraday's law of induction ($V = -\frac{\partial \Phi}{\partial t}$). The electromotive force generated in the junction is equivalent to that of a classical inductor, where $\Phi = \phi_0 \phi$. The electromotive force that arises in the junction does not originate from magnetic energy, as there is no magnetic field within the superconductor. Rather, it stems from the kinetic energy of the charge carriers (Cooper pairs); hence, it is referred to as kinetic inductance. In the context of superconductive circuits, it is convenient to use the variable $\Phi = \phi_0 \phi$, as it emerges as the conjugate variable of the charge in the Hamiltonian formalism. From the time derivative of Equation B.1, we can obtain:

$$V = \frac{\phi_0}{I_c \cos(\phi)} \frac{\partial I}{\partial t}.$$

This equation is formally equivalent to the characteristic expression for an inductor, $V = L \frac{\partial I}{\partial t}$.

Thus, we can identify the inductance associated with the Josephson junction as:

$$L = \frac{L_J}{\cos(\phi)}, \quad \text{where} \quad L_J = \frac{\phi_0}{I_c}. \quad (\text{B.3})$$

Here, the Josephson inductance L_J has been defined. Finally, to compute the energy associated with a Josephson junction:

$$E = \int V dt = \int \phi_0 I_c \sin(\phi) \frac{d\phi}{dt} dt = E_J (1 - \cos(\phi)), \quad (\text{B.4})$$

where we have introduced the Josephson energy $E_J = \phi_0 I_c$ and assumed that the phase difference at the beginning of the integration interval is zero.

Appendix C

Randomized Benchmarking

To benchmark the average gate fidelity of a quantum chip, and to get a simplified noise modeling, it is possible to employ a technique known as RB [336, 450, 451]. RB enables efficient estimation of the average error magnitude across a set of quantum gates, with resource requirements scaling polynomially with the number of qubits. A RB protocol employs sequences of varying lengths of randomly chosen Clifford gates, where the ideal composite operation of the sequence is the identity. To produce such a sequence of depth $m + 1$, each of the first m gates in the sequence are picked randomly from the Clifford group. The last gate is then uniquely determined as the Clifford element which inverts the composition of the previous m gates. In the presence of noise, the actual sequence of Clifford gates does not represent an identity operation. Instead, there is a certain survival probability (P) of measuring the initial state after performing the sequence. By estimating this survival probability across multiple independent random sequences of increasing depth (d), we can fit an exponential decay of the form:

$$P(d) = Ap^d + B, \quad (\text{C.1})$$

as shown in Figure C.1.

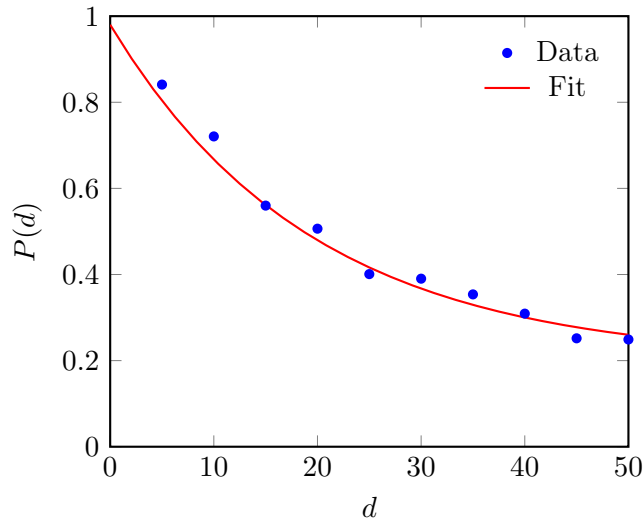


Figure C.1: Example of randomized benchmarking experiment. The survival probability P decays with the depth of the circuit d in the form: $P(d) = Ap^d + B$.

The average error map, therefore, behaves like a completely depolarizing model with an average depolarizing parameter p . This parameter can be used to estimate the average gate fidelity of quantum gates.

Employing RB as a noise predictor involves extracting the decay parameter and introducing a depolarizing channel with a depolarizing parameter equal to the decay parameter after each gate. Although this noise model, which projects all noise sources onto the depolarizing channel, may seem unrealistic, it provides a useful baseline for comparison with other more sophisticated algorithms.

Appendix D

Unitary Representation Theorem

Given a set of Kraus operators $\{E_k\}$, that describe the evolution of a non-isolated quantum system ($\mathcal{H}_1$), it is always possible to introduce an auxiliary system ($\mathcal{H}_2$) such that the evolution of the total system is described by a unitary operator (U). Given the state of the system $|\psi\rangle_1$, U is defined on states of the form $|\psi\rangle_1 |0\rangle_2$ in the following way:

$$U |\psi\rangle_1 |0\rangle_2 = \sum_k E_k |\psi\rangle_1 |k\rangle_2,$$

where $\{|k\rangle_2\}$ is an orthonormal basis for the auxiliary subsystem. The dimension of the auxiliary Hilbert space is defined by the number of Kraus operators. The operator U preserves the scalar product when acting on states of the form $|\psi\rangle_1 |0\rangle_2$; given two arbitrary states $|\psi\rangle_1$ and $|\phi\rangle_1$:

$$\langle\phi|_1 \langle 0|_2 U^\dagger U |\psi\rangle_1 |0\rangle_2 = \sum_j \langle\phi|_1 \langle j|_2 E_j^\dagger \sum_k E_k |\psi\rangle_1 |k\rangle_2 = \sum_k \langle\phi|_1 E_k^\dagger E_k |\psi\rangle_1 = \langle\phi|_1 \langle 0|_2 |\psi\rangle_1 |0\rangle_2.$$

U can be extended to the whole Hilbert space preserving unitarity. This is the unitary representation of the superoperator, in fact, given a density operator of the first system ρ_1 :

$$\begin{aligned} \mathcal{S}(\rho_1) &= \text{Tr}_2(U \rho_1 \otimes |0\rangle_2 \langle 0|_2 U^\dagger) = \sum_l \langle l|_2 \left[\sum_j |j\rangle_2 E_j \rho_1 \sum_k E_k^\dagger |\psi\rangle_1 \langle k|_2 \right] |l\rangle_2 = \\ &= \sum_j \sum_k \sum_l E_j \rho_1 E_k^\dagger \langle k|_2 |l\rangle_2 \langle l|_2 |j\rangle_2 = \sum_l E_l \rho_1 E_l^\dagger. \end{aligned}$$

This result is part of the Choi-Kraus theorem. This theorem states that given a map ($\mathcal{S}$) that:

1. Is linear: $\mathcal{S}(\alpha\rho_1 + \beta\rho_2) = \alpha\mathcal{S}(\rho_1) + \beta\mathcal{S}(\rho_2)$.
2. Preserves Hermiticity.
3. Preserves trace.
4. Is completely positive: every extension, from $\mathcal{H}$ to $\mathcal{H} \otimes \mathcal{H}_E$, of the form $\mathcal{S} \otimes \mathcal{I}_E$ is positive.

Then $\mathcal{S}$ admits a Kraus representation and a unitary representation.

Appendix E

Classical and Quantum Hamming Bound

The standard notation to identify an error correction code, both classical or quantum, is the triplet of numbers (n, k, d) where:

k is the number of logical bits or qubits, it is the logical information stored in the code.

n is the number of physical bits or qubits used in the code to store the logical information.

d is the minimal distance of the code, that means the minimum number of errors necessary to change the logical information encoded by the code.

Sometimes it is useful to specify the number of maximum errors that can be corrected by the code (t), this number is bounded by the inequality: $t \leq (d - 1)/2$ [452]. The classical Hamming bound puts an upper limit on the minimal distance of the code, given the number of physical and logical bits. If j errors are present in a physical word, the number of possible locations of these errors is given by $\binom{n}{j}$. Thus, the number of physical strings that can be obtained incurring in no more than d errors is:

$$\#strings = \sum_{j=0}^d \binom{n}{j}.$$

This number, multiplied by the number of logical strings (2^k), cannot exceed the number of physical strings (2^n):

$$\left[\sum_{j=0}^d \binom{n}{j} \right] 2^k \leq 2^n.$$

This relation is the classical Hamming bound. For example it states that, to protect one logical bit from one error ($d = 3, t = 1$), the number of physical bits can't be less than three.

The quantum Hamming bound puts an upper limit on the minimal distance of the code given the number of physical and logical qubits. In this case, the number of physical or logical strings is given by the possible results of qubits in the computational basis. If j errors are present in the physical qubits, the possible locations of these errors are $\binom{n}{j}$.

As shown in Section 2.3.5, for each location there are three possible errors: σ_x , σ_y and σ_z . The

number of physical states that can be obtained incurring in no more than d errors is given by:

$$\#states = \sum_{j=0}^d \binom{n}{j} 3^j.$$

This number multiplied by the number of logical strings 2^k cannot exceed the number of physical strings 2^n :

$$\left[\sum_{j=0}^d \binom{n}{j} 3^j \right] 2^k \leq 2^n.$$

For example, to protect one logical qubit from one error ($d = 3, t = 1$) the number of physical qubits can't be less than five.

Appendix F

Loss Functions

This appendix presents the most important loss functions used in machine learning for classification and regression problems. The target values (labels) are denoted by $\{y'_i\}$, while the algorithm's output is $\{y_i\}$; N is the number of elements in the training set.

- **Mean Squared Error (MSE):**

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - y'_i)^2 .$$

This loss function is commonly used in regression problems to fit data.

- **Mean Absolute Error (MAE):**

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - y'_i| .$$

MAE serves as an alternative to MSE. It is less sensitive to outliers but presents differentiability issues.

- **Binary Cross Entropy (BCE):**

$$BCE = \frac{1}{N} \sum_{i=1}^N [-y'_i \ln(y_i) - (1 - y'_i) \ln(1 - y_i)] .$$

This function is applied in neural networks for binary classification problems (class 1: $y'_i = 0$; class 2: $y'_i = 1$). The output y_i represents the probability that the element belongs to class 2, while $1 - y_i$ represents the probability of class 1. For BCE, it is recommended that the output be derived from a sigmoid activation function (Appendix G).

- **Categorical Cross Entropy (CCE):** a generalization of BCE for multiclass classification:

$$CCE = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^C -t_{ik} \ln(s_{ik}) ,$$

where C is the total number of classes, and $t_{ik} = 1$ if the i^{th} element belongs to the k^{th} class; otherwise, $t_{ik} = 0$ (one-hot encoding). Similarly, s_{ik} is the output probability that the i^{th}

element belongs to the k^{th} class ($\sum_{k \in C} s_{ik} = 1$).

For CCE, a softmax activation function (Appendix G) in the final layer of the neural network is recommended.

- **Independent Multiclass Cross Entropy (IMCE):** generalization of BCE for independent multiclass classification, where elements may belong to multiple classes, with no interclass correlations.

$$IMCE = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^C [-t_{ik} \ln(s_{ik}) - (1 - t_{ik}) \ln(1 - s_{ik})] ,$$

where $t_{ik} = 1$ if the i^{th} element belongs to the k^{th} class, and s_{ik} is the output probability for the i^{th} element belonging to the k^{th} class. This approach decomposes the problem into multiple binary classification tasks.

Appendix G

Activation Functions

This appendix presents the most commonly used activation functions in neural networks.

- **Sigmoid:**

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

This activation function is primarily used in the output layer for binary classification problems. If applied in hidden layers, it can cause saturation, leading to the vanishing gradient problem. Additionally, since this function is not centered at zero, it may slow down the training process.

- **Hyperbolic Tangent (tanh):**

$$\tanh(z) = 2\sigma(2z) - 1.$$

This function is a zero-centered variation of the sigmoid function. However, it is seldom used in hidden layers due to the vanishing gradient problem.

- **Softmax:**

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^N e^{z_k}}, \quad j = 1, 2, \dots, N.$$

This is a vector generalization of the sigmoid function, used for multiclass classification problems. Here, N represents the number of possible output classes.

- **Rectified Linear Unit (ReLU):**

$$\text{ReLU}(z) = \max\{0, z\}.$$

This is the most widely used activation function, as it does not saturate. However, it may pose challenges during training because the gradient is zero for negative values of z .

- **Leaky ReLU:**

$$\text{LReLU}(z) = \max\{\alpha z, z\},$$

where α is a hyperparameter, typically with a value around 0.1. This ReLU variant can help mitigate the zero-gradient issue associated with the standard ReLU function.

- **Exponential Linear Unit (ELU):**

$$\text{ELU}(z) = \begin{cases} z & \text{for } z > 0 \\ e^z - 1 & \text{for } z \leq 0 \end{cases}$$

This ReLU-like function has better differentiability properties than the standard ReLU, but it is more computationally intensive.

Appendix H

Optimization Methods

This appendix reports the most famous gradient-based optimization methods used in ML.

- **Stochastic Gradient Descent (SGD)**: in this simple variation of the standard gradient descent, the gradient is not computed using the entire dataset but rather using a subset of data. The training set is divided into subsets, and one of them is randomly picked at each training step. This is useful when the training set is large, as the weight update procedure may become computationally inefficient when using the entire dataset.

The use of a smaller dataset makes the algorithm more sensitive to statistical fluctuations. In exchange, the algorithm gains faster learning iterations. This accelerates the training process but the algorithm will only converge to a region near the minimum, rather than the exact minimum itself, due to these fluctuations. However, these fluctuations also prevent overfitting and make the descent less sensitive to local minima.

The size of the subsets used for training is an important hyperparameter. Typical dimensions are 32, 64, or 128 elements. It is also possible to use just one element for each training epoch, a technique called online learning, which is commonly used for recommender systems on websites.

- **Learning Rate Decay**: to speed up the convergence of the optimization algorithm, the learning rate can be decreased at the beginning of each training epoch. This ensures that the first steps of gradient descent are long enough to reach a region near the minimum. This helps the algorithm become less sensitive to local minima. Once the algorithm gets closer to a good region, a low learning rate can help find the exact minimum.

Given t as the current epoch, T as the total number of epochs, and α_0 as the initial learning rate, typical functions used to decrease the learning rate over epochs (α_t) are:

$$\text{Cosine: } \alpha_t = \frac{1}{2}\alpha_0 \left(1 + \cos \left(\pi \frac{t}{T} \right) \right),$$

$$\text{Linear: } \alpha_t = \alpha_0 \left(1 - \frac{t}{T} \right),$$

$$\text{Inverse Square Root: } \alpha_t = \alpha_0 \frac{1}{\sqrt{t}}.$$

- **Adaptive Learning Rate:** with this technique, the learning rate is scaled for each parameter independently. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. For example, in ADAGrad, the learning rate associated with each parameter is individually scaled inversely proportional to the root of the historical sum of the squares of the gradients for that parameter. However, ADAGrad reduces the learning rate too quickly, which is why RMSProp is more commonly used today. In RMSProp, the learning rate is scaled inversely proportional to a weighted average of previous gradient values, with the weights scaling exponentially. Given a weight W , a loss function L , and an initial learning rate α ; at the training epoch t , the weight is updated as follows:

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{E[g^2]_t}} \frac{\partial L}{\partial W},$$

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) \left(\frac{\partial L}{\partial W} \right)^2,$$

where β is a hyperparameter called the forgetting factor.

- **Momentum:** this technique is a popular method to speed up the convergence of gradient descent techniques, especially when facing large curvature loss functions, saddle points, and plateau regions. In this method, a fraction of the previous update's direction is added to the current update. Given a weight W , a loss function L , and an initial learning rate α ; at the training epoch t , the weight is updated as follows:

$$w_t = w_{t-1} - \alpha v_t,$$

$$v_t = \beta v_{t-1} + (1 - \beta) \frac{\partial L}{\partial W}.$$

Where β is a hyperparameter that, in analogy to physics, acts similarly to inertia during gradient descent. Typical values for β range from 0.9 to 0.99.

- **ADAM:** this optimizer combines both momentum and adaptive learning rate (RMSProp), and is the most widely used nowadays. Given a weight W , a loss function L , and an initial learning rate α ; at the training epoch t , the weight is updated as follows:

$$w_t = w_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon},$$

$$m_t = \frac{E[g]_t}{1 - \beta_1^t} \quad E[g]_t = \beta_1 E[g]_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial W},$$

$$v_t = \frac{E[g^2]_t}{1 - \beta_2^t} \quad E[g^2]_t = \beta_2 E[g^2]_{t-1} + (1 - \beta_2) \left(\frac{\partial L}{\partial W} \right)^2.$$

Typical values for the hyperparameters are: $\alpha = 0.001$, $\epsilon = 10^{-8}$, $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

Appendix I

Regularization Methods

This appendix presents some key regularization methods used to mitigate overfitting in neural networks. We will use w to indicate the vector containing all the parameters of the model: $w = (w_1, w_2 \dots w_N)$.

- **Dropout:** this regularization technique simulates training an ensemble of neural networks. During training, a random subset of neurons, along with their connections, are removed with a fixed probability [453] as reported in Figure I.1. This forces the model to rely less on particular features, encouraging it to generalize better.

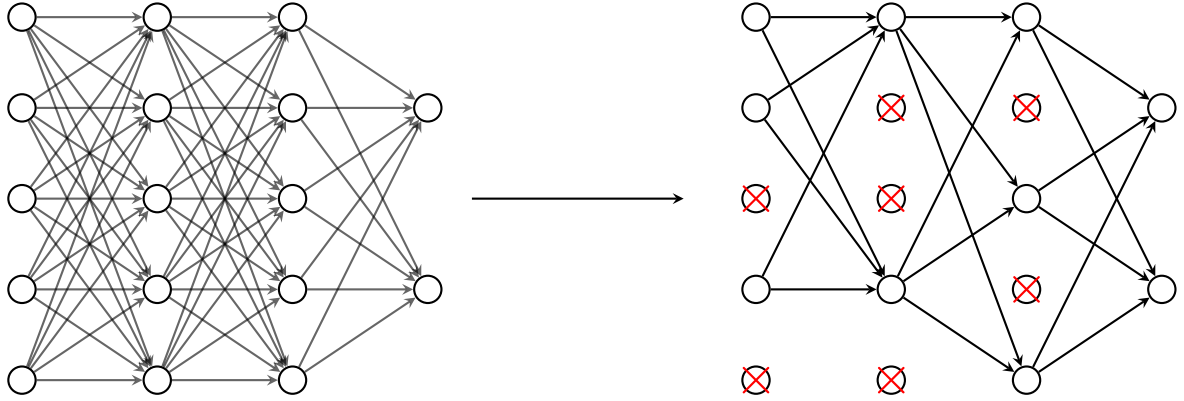


Figure I.1: Graphic representation of the dropout technique in a FFNN.

- **L1 regularization:** this method introduces an extra term ($\omega_1(w)$) in the loss function to penalize large weight values, helping prevent overfitting.

$$\omega_1(w) = \alpha \|w\|_1 = \alpha \sum_k |w_k|,$$

where α is a hyperparameter that controls the extent of regularization; larger α values drive more weights toward zero.

- **L2 regularization:** this technique adds a term to the loss function ($\omega_2(w)$) defined as:

$$\omega_2(w) = \alpha \|w\|_2^2 = \alpha \sum_k w_k^2,$$

where again, a higher α value further penalizes large weights.

- **L1 + L2 regularization:** A combination of L1 and L2 regularization can also be applied adding an extra term ($\omega_3(w)$) to the loss function:

$$\omega_3(w) = \lambda [(1 - \alpha)\|w\|_1 + \alpha\|w\|_2^2] ,$$

where α and λ are hyperparameters.

Appendix J

Backpropagation

The backpropagation algorithm is used to adjust the weights of neurons in a neural network based on its output values. To train a NN, it is necessary to define a loss function. In supervised learning, the loss function (L) depends on the target, which is the correct output of the training set (y'), and the output vectors of the NN (y). Since the output vectors of a NN depend on the input vector (x) and all neuron weights (w), the loss function also depends on these parameters:

$$L = L(y', y(x, w)).$$

It is useful to assume that the loss function can be expressed as an average over individual loss functions for each training sample (L^i):

$$L(y', y(x, w)) = \frac{1}{n} \sum_{i=1}^n L^i(y'^i, y^i(x^i, w)),$$

where n is the total number of samples in the training set. With this assumption, we can compute weight adjustments based on the individual loss functions L^i and later average the adjustments over the entire training set. For simplicity, we will omit the superscript denoting individual loss functions, targets, and outputs from now on.

The backpropagation algorithm has two phases. In the forward phase, the output for the input vectors is computed with fixed weights, propagating information from one layer to the next. In the backward phase, the loss is calculated by comparing the NN output with the target. The correction is propagated backward (from one layer to the previous layer) by recursively applying the chain rule. The weights are updated using standard gradient descent:

$$w_{j,i}^k \rightarrow w_{j,i}^k - \alpha \frac{\partial L(w)}{\partial w_{j,i}^k},$$

where α is the learning rate. To compute the partial derivatives of the loss function with respect to weights and biases, $\frac{\partial L(w)}{\partial w_{i,j}^k}$, it is useful to introduce the neuron error δ_j^k :

$$\delta_j^k := \frac{\partial L(w)}{\partial z_j^k}.$$

This quantity represents the partial derivative of the loss function with respect to the input of the

activation function (z_j^k) for the j^{th} neuron in the k^{th} layer. Using the chain rule, we can express the partial derivatives of the loss function with respect to weights in terms of the neuron error:

$$\frac{\partial L(w)}{\partial w_{j,i}^k} = \frac{\partial L(w)}{\partial z_j^k} \frac{\partial z_j^k}{\partial w_{j,i}^k} = \delta_j^k a_i^{k-1},$$

where we have used the relation $z_j^k = \sum_i w_{j,i}^k a_i^{k-1}$.

In the output layer, the neuron error depends on the form of the loss function and the activation function ϕ . For example, using the MSE activation function:

$$\delta_j^{\text{out}} = \frac{\partial (y'_j - \phi(z_j^{\text{out}}))^2}{\partial z_j^{\text{out}}} = -2(y'_j - \phi(z_j^{\text{out}}))\phi'(z_j^{\text{out}}).$$

For hidden layers, a complication arises because the correct output values of the neurons are unknown. The chain rule can be applied to express the neuron error in terms of the neuron errors in subsequent layers:

$$\delta_j^k = \frac{\partial L(w)}{\partial z_j^k} = \sum_l \frac{\partial L(w)}{\partial z_l^{k+1}} \frac{\partial z_l^{k+1}}{\partial z_j^k} = \sum_l \delta_l^{k+1} \frac{\partial z_l^{k+1}}{\partial z_j^k}. \quad (\text{J.1})$$

This expression can be written more explicitly by noting that:

$$z_l^{k+1} = \sum_{m=0}^M w_{l,m}^{k+1} a_m^k = \sum_{m=1}^M w_{l,m}^{k+1} \phi(z_m^k) + w_{l,0}^{k+1},$$

where ϕ is the activation function, and M is the number of neurons in the k^{th} layer. Differentiating this equation, we obtain:

$$\frac{\partial z_l^{k+1}}{\partial z_j^k} = w_{l,j}^{k+1} \phi'(z_j^k).$$

Substituting into Equation J.1 gives:

$$\delta_j^k = \sum_l \delta_l^{k+1} w_{l,j}^{k+1} \phi'(z_j^k).$$

This is a recursive equation that can be used to compute neuron errors, starting from the output layer and moving backward.

Appendix K

Parameter Shift Rule

The parameter shift rule allows the computation of the gradients of a loss function with respect to the parameters of the gates in a quantum circuit. In this appendix, we will demonstrate the parameter shift rule in the case of gates that implement a transformation of the type:

$$\hat{U}(\theta) = e^{-i\alpha\theta\hat{G}}, \quad (\text{K.1})$$

where α is a positive real number and the generator $\hat{G}$ is a Hermitian operator with eigenvalues in the set $\{+1, -1\}$, thus $\hat{G}^2 = I$. A more general version of the parameter shift rule can be found in [355]. For the demonstration, consider a quantum circuit implementing the transformation $\hat{U}(\theta)$, composed of n gates $\{\hat{U}_i(\theta_i)\}$:

$$\hat{U}(\theta) = \hat{U}_n(\theta_n)\hat{U}_{n-1}(\theta_{n-1})\dots\hat{U}_i(\theta_i)\dots\hat{U}_1(\theta_1).$$

We can define the loss function $\mathcal{L}$ as the expectation value of an observable $\hat{L}$:

$$\mathcal{L}(x, \theta) = \langle 0 | \hat{U}_x^\dagger \hat{U}(\theta)^\dagger \hat{L} \hat{U}(\theta) \hat{U}_x | 0 \rangle,$$

where $\hat{U}_x$ is a circuit that prepares the input state x . Let us explicitly show the dependence of the loss function on the parameter θ_i , from now on we will consider all the other parameters as fixed,

$$\mathcal{L}(x, \theta_i) = \langle \psi_{i-1} | \hat{U}_i^\dagger(\theta_i) \hat{L}_{i+1} \hat{U}_i(\theta_i) | \psi_{i-1} \rangle,$$

where we have absorbed the initial part of the circuit in the state $|\psi_{i-1}\rangle$:

$$|\psi_{i-1}\rangle = \hat{U}_{i-1}(\theta_{i-1})\dots\hat{U}_1(\theta_1)\hat{U}_x | 0 \rangle,$$

and we have included the final part of the circuit in the observable $\hat{L}_{i+1}$:

$$\hat{L}_{i+1} = \hat{U}_{i+1}(\theta_{i+1})^\dagger \dots \hat{U}_n(\theta_n)^\dagger \hat{L} \hat{U}_n(\theta_n) \dots \hat{U}_{i+1}(\theta_{i+1}).$$

The property in Equation K.1 allows us to write the gradient of the transformation with respect to the parameter θ_i

$$\frac{\partial \hat{U}_i(\theta_i)}{\partial \theta_i} = -i\alpha \hat{G}_i \hat{U}_i(\theta_i).$$

Thus, the gradient of the loss function with respect to the parameter θ_i can be computed as:

$$\begin{aligned} \frac{\partial \mathcal{L}(x, \theta_i)}{\partial \theta_i} &= i\alpha \langle \psi_{i-1} | \hat{U}_i^\dagger(\theta_i) \hat{G}_i \hat{L}_{i+1} \hat{U}_i(\theta_i) | \psi_{i-1} \rangle - i\alpha \langle \psi_{i-1} | \hat{U}_i^\dagger(\theta_i) \hat{L}_{i+1} \hat{G}_i \hat{U}_i(\theta_i) | \psi_{i-1} \rangle = \\ &= i\alpha \langle \psi_{i-1} | \hat{U}_i^\dagger(\theta_i) \left[\hat{G}_i, \hat{L}_{i+1} \right] \hat{U}_i(\theta_i) | \psi_{i-1} \rangle. \quad (\text{K.2}) \end{aligned}$$

To rewrite the commutator $[\hat{G}_i, \hat{L}_{i+1}]$, it is useful to use the Euler decomposition on the operator $\hat{U}_i$:

$$e^{-i\alpha\theta\hat{G}} = I \cos(\alpha\theta) - i\hat{G} \sin(\alpha\theta).$$

With an explicit calculation it is possible to obtain the following identity:

$$i [\hat{G}_i, \hat{L}_{i+1}] = \hat{U}_i^\dagger \left(\frac{\pi}{4\alpha} \right) \hat{L}_{i+1} \hat{U}_i \left(\frac{\pi}{4\alpha} \right) - \hat{U}_i^\dagger \left(-\frac{\pi}{4\alpha} \right) \hat{L}_{i+1} \hat{U}_i \left(-\frac{\pi}{4\alpha} \right).$$

By substituting this identity in Equation K.2, we obtain the parameter shift rule:

$$\begin{aligned} \frac{\partial \mathcal{L}(x, \theta_i)}{\partial \theta_i} &= \alpha \langle \psi_{i-1} | \hat{U}_i^\dagger \left(\theta_i + \frac{\pi}{4\alpha} \right) \hat{L}_{i+1} \hat{U}_i \left(\theta_i + \frac{\pi}{4\alpha} \right) | \psi_{i-1} \rangle - \\ &- \alpha \langle \psi_{i-1} | \hat{U}_i^\dagger \left(\theta_i - \frac{\pi}{4\alpha} \right) \hat{L}_{i+1} \hat{U}_i \left(\theta_i - \frac{\pi}{4\alpha} \right) | \psi_{i-1} \rangle = \alpha \left(\mathcal{L}(x, \theta_i + \frac{\pi}{4\alpha}) - \mathcal{L}(x, \theta_i - \frac{\pi}{4\alpha}) \right). \end{aligned}$$

The standard convention for quantum circuits is to use parametrized gates in the form of $U(\theta) = e^{-i\frac{\theta}{2}\sigma}$, where σ is a Pauli matrix. In this case, the value of α is $\frac{1}{2}$, we obtain the standard form of the parameter shift rule reported in Section 5.1.2.

List of Figures

2.1	Representation of a qubit state as a point on the Bloch sphere.	7
2.2	Schematic representation of the EPR paradox.	9
2.3	Circuit model representation of a quantum circuit composed of two qubits for the EPR paradox (Section 2.1.2). This circuit allows to generate entangled states using single and multi-qubit gates. At the end of the circuit measurement gates are used to indicate a measurement apparatus.	11
2.4	Gate decomposition into C-NOT and single-qubit gates of the SWAP gate (left) and the CZ gate (right).	13
2.5	Circuit symbol for the Toffoli gate (left) and its decomposition into single-qubit gates and C-NOT gates. The operator T is defined as $T := R_z(\pi/4)$	14
2.6	Inclusion relations between computational complexity classes (P=Polynomial, NP=Non-Polynomial, NPC=Non-Polynomial Complete, BPP=Bounded Probabilistic Polynomial, BQP=Bounded Quantum Polynomial) based on the current most accredited conjectures. The set of algorithms where it may be possible to obtain quantum advantage has been highlighted, the factorization algorithm is included in this set.	15
2.7	Schematic representation of parametric downconversion: a pump photon with angular frequency ω_p enters the nonlinear crystal (NLC) and generates two lower-energy photons, the signal photon (with angular frequency ω_s) and the idler photon (with angular frequency ω_i), where $\omega_p = \omega_s + \omega_i$. By measuring the idler photon (M), the generation of the signal photon is ensured.	20
2.8	Schematic representation of a linear Paul trap. Four parallel rods are arranged at the vertices of a square, the potential applied to the rods oscillates in the radio frequency range. The equilibrium region created by the pseudopotential is represented by the dashed line.	22
2.9	Representation of commonly used orbital energy levels in different kinds of trapped ions qubits. On the left, structure of an ion with no nuclear spin angular momentum ($I = 0$) and an external magnetic field (Zeeman splitting). In this kind of ions it is possible to implement optical, fine structure and Zeeman qubits. On the right, structure of an ion with nuclear spin angular momentum ($I = 1/2$) and external magnetic field. This kind of ions is used to implement hyperfine qubits.	23
2.10	Schematic representation of state-dependent fluorescence for the measurement of trapped ion qubits. A laser induces a transition only if the ion is in a specific state (ground state in this example), fluorescence light is collected to determine the qubit state.	25
2.11	On the left: an LC superconducting circuit with its classical and quantized Hamiltonians. On the right: the harmonic potential and corresponding energy levels are depicted.	27

2.12	On the left: an anharmonic circuit with a Josephson junction, showing the classical and quantized Hamiltonians (to first-order perturbation). On the right: the anharmonic potential and energy levels of the system, with the harmonic potential and energy levels overlaid using dashed lines. The gap between the ground state and the first excited energy level is ω_q , while the gap between the first and second excited states is $\omega_q - \alpha$, where typical values of α are around 10% of ω_q	28
2.13	Circuit model of a Transmon qubit coupled with a drive line ($V_d(t)$) through a capacitor (C_d).	30
2.14	Circuit model of two Transmon qubits directly coupled through a capacitor (C_c). The qubit on the left is flux tunable, the dc-SQUID element is highlighted.	31
2.15	Circuit model a Transmon qubit coupled with a resonator (LC circuit) through a capacitor (C_c). A readout line capacitive coupled with the resonator (C_{rl}) allows to perform a measurement of this system.	33
2.16	Resonator response as a function of the probe signal frequency. Since the resonator is coupled to the qubit, its resonant frequency shifts depending on the qubit's state. This results in two distinct response curves: one for each qubit state. By observing these shifts in the resonator's response, the state of the qubit can be discriminated.	34
2.17	Schematic representation of the interaction between a quantum system and the environment.	39
2.18	Representation of the effect of the dephasing channel on the Bloch sphere.	41
2.19	Representation of the effect of the depolarizing channel on the Bloch sphere.	42
2.20	Representation of the effect of the amplitude damping channel on the Bloch sphere.	43
2.21	Nine-qubit Shor's error correction code circuit. The circuit has been divided into the encoding and decoding circuits. In the decoding circuit a red dashed line separates the parts of the circuit in charge of finding and correcting X and Z errors.	45
2.22	Benchmark of the performance of the built-in Qibolab transpilers (routing pass) evaluated considering the CNOT overhead. Circuits of different types have been considered in order to test the transpilers on both structured and unstructured circuits. In particular we have considered a five qubits QFT circuit and random circuits with 5, 20 and 100 gates. The results for random circuits have been averaged over 50 circuits.	51
3.1	Schematic representation of the main difference between traditional programming (left) and machine learning (right).	54
3.2	Schematic summary of the main learning paradigms, tasks and applications of machine learning.	58
3.3	Example of a dataset for with a linear correlation between the input data x and the label y	59
3.4	Consequences of the learning rate choice on the behavior of the gradient descent algorithm when optimizing the loss function $J(\theta)$. With a low learning rate (left), the algorithm requires many updates to reach the minimum and is more sensitive to local minima. An optimal learning rate (center) allows the algorithm to rapidly converge to the minimum. A too high learning rate (right) results in drastic parameter updates, which may lead to divergent behavior.	60
3.5	Dataset without a linear correlation. A linear function (orange) is too simple to fit the model (underfitting). An high order polynomial (green) perfectly fits the data but is not a good predictor (overfitting). A quadratic function (red) is a good trade-off between complexity of the model and generalization properties	61
3.6	Common behavior of the loss function computed on the training and test sets during the training process. The optimal training epoch (early stopping) is at the minimum of the loss function evaluated on the test set.	62

3.7	Schematic representation of an artificial neuron. The input signals $\{x_1, x_2, \dots, x_n\}$ are combined with weights $\{w_1, w_2, \dots, w_n\}$, and a bias input x_0 (shown with dashed lines). The weighted sum, denoted by z , serves as the input to the activation function $f(z)$, which then computes the output y . A perceptron model is obtained by using a step function as the activation function.	64
3.8	Graph representation of a dense FFNN with three hidden layers. The i^{th} neuron in the k^{th} layer is denoted as a_i^k	66
3.9	Notation to indicate the weights and activation functions in a FFNN.	67
3.10	A deep neural network may use intermediate features to accomplish different tasks.	69
3.11	Typical behavior of the average update of the parameters in different layers of a FFNN during the training process.	70
3.12	Toy model of a FFNN with n hidden layers with just one neuron (Left). Sigmoid function and its derivative (Right).	71
3.13	Example of a residual connection in a neural network. In this example the initial information skips two layers and is summed element-wise with the processed information.	71
3.14	Input neurons in the local receptive field are connected to a neuron of the first hidden layer. In this example the local receptive field have dimension 3×3	73
3.15	An example of convolution operation with a kernel of dimension 3×3	74
3.16	An example of an architecture of a CNN for image classification.	74
3.17	Schematic representation of an autoencoder architecture. The input data is compressed by the encoder, to a smaller number of features in the latent space. The decoder tries to reconstruct the original data from the compressed one.	75
3.18	Typical behavior of the loss function distributions for standard and anomalous data in an anomaly detection problem. Anomalous data are usually expected to have a higher reconstruction loss for the autoencoder.	75
3.19	In a RL algorithm an agent is trained to perform optimal actions in an environment to maximize the reward.	79
3.20	Schematic representation of the main reinforcement learning paradigms and some important algorithms.	80
4.1	Schematic representation of a surface code. Data qubits are represented as circles while X and Z measurement qubits are represented as squares. Each qubit is connected with its nearest neighbours.	85
4.2	Examples of logical operators in a $d = 5$ surface code. Z and X logical operators are composed of chains of single-qubit Z or X operators connecting the Z or X sides respectively.	86
4.3	Signal error produced by single data qubit errors. The highlighted measurement qubits change their measurement value.	86
4.4	Comparison between accuracy obtained with HLD based on FFNN and CNN and the accuracy obtained with MWPM for different code distances (d) and single-qubit error probabilities (Physical Qubit error).	90
4.5	Performance of MWPM for the depolarising plus measurement errors model using a different number or imperfect measurement cycles (3 and 9). The depolarising error probability (p) has been set equal to the measurement error probability (q). For this noise model a different number of measurement cycles does not modify significantly the decoding problem.	91
4.6	Comparison between accuracy obtained with HLD based on CNN and the accuracy obtained with the classical decoder MWPM for the depolarising plus measurement errors model, the depolarising error probability (p) has been set equal to the measurement error probability (q).	91

4.7	Comparison of the accuracy, on different code distances (d), obtained with a CNN with no dilation factor (N) and the same model with a dilated convolution with a dilation rate equal to two on all the convolutional layers except the first one (D). For the depolarising error model (Top) the neural networks have been trained on a dataset with $p = 0.1$. While for the depolarising plus measurement errors model (Bottom) $p = q = 0.01$ (three measurement cycles). The graphs report the accuracy of the CNN decoder obtained using the described neural network architectures for different physical qubit error rates.	93
4.8	Accuracy of the HLD trained and tested on different single-qubit error probabilities for different code distances d . The training set single-qubit error probability (p) is reported in the legend.	95
4.9	Accuracy of the HLD trained and tested on different depolarising and measurement error probabilities for different code distances d . The training set single-qubit error probability (p) and measurement error probability (q) is reported in the legend. . . .	95
4.10	Saliency map of a manually generated error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubits with X errors are highlighted in green, while the measurement qubits that report the error signal are highlighted in blue. Two single X errors, E_1 and E_2 , and two errors chains composed of three X errors, E_3 and E_4 have been inserted in the surface code. The errors closer to the centre between the X sides (E_1 and E_3) are the ones that contribute more for the classification.	97
4.11	Saliency map of a manually generated error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubit with depolarising errors are highlighted in green, while the measurement qubit that report the error signal are highlighted in blue. Two errors chains composed respectively of five X (horizontal) and Z (vertical) errors have been inserted in the middle of the surface code. The HLD is not able to identify the creation of an Y logical error after the correction of the simple decoder.	98
4.12	Saliency map of an error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubit with X errors are highlighted in green, while the measurement qubit that report the error signal are highlighted in blue. In this case the HLD is able to correct the error syndrome while the MWPM fails. This example has been generated using a depolarising error model (perfect measurements) with single-qubit error probability $p = 0.15$, the error chain responsible for the failure of MWPM has been extracted and analysed.	99
4.13	Performance comparison between HLD trained on a standard train set with single-qubit error probability $p = 0.1$ and the model trained on the enhanced train set that includes samples of error chains of length five.	100
4.14	Saliency map of the same error syndrome reported in Figure 4.11, decoded with the HLD trained on the enhanced train set. The dark red regions are the ones that contribute more to the output, they are placed near the error signals and the centre of the surface code. This HLD is able to correctly identify the presence of error chains from a sparse error syndrome.	101
4.15	Example of a two-qubit quantum circuit (top) and its vector representation (bottom). A two-qubit circuit of depth three is represented as a tensor of size $(2, 3, 8)$, where the first entry identifies the qubit, the second denotes the circuit moment, and the third specifies the type of gate or noise channel.	105

4.16	Schematization of the policy neural network for the DM algorithm. A common feature extractor (FE), composed of a CNN, maps the QCR to a high-dimensional latent feature space. The resulting feature vector serves as input to both the actor policy ($A\pi$) and the critic policy ($C\pi$). The actor policy is responsible for selecting the best action for the agent, while the critic policy estimates the future reward for the action chosen by the actor.	106
4.17	Training process of the RL algorithm. At each step the RL agent receives an observation of the environment, the quantum circuit representation of a non-noisy circuit. The agent policy performs an action that consists in putting any number of noise channels at a given circuit moment. At the end of the circuit the reward is computed to minimize the trace distance of the density matrix of the original noisy circuit and the reconstructed one.	107
4.18	Average density matrix fidelity throughout the training process of the RL agent on single-qubit circuits (left) and three-qubit circuits (right). The metrics were evaluated on a dataset of 100 circuits for the single-qubit case and 800 circuits for the three-qubit case, using 80% for the training set and 20% for the test set. Shaded regions represent the standard deviation. In the single-qubit case, convergence of the training process was reached after approximately 4×10^5 episodes, achieving an average fidelity of about 0.99 on the test set. For the three-qubit case, convergence was reached after approximately 1.5×10^6 episodes, achieving an average fidelity of about 0.98 on the test set. No significant overfitting was observed in either case.	110
4.19	Performance evaluation of different noise models on Clifford circuits with depths varying from 3 to 30 for the single-qubit case (left) and the three-qubit case (right). Performance is measured using average density matrix fidelity; shaded regions represent the standard deviation. The RL agent was benchmarked against the RB method, the noiseless scenario, and the maximally mixed state (MMS). The absolute advantage of using the RL model is evident when compared to the case with no noise channel insertion. Moreover, in both cases, the RL model outperforms the RB model. Fidelity between the noisy circuit and the MMS is provided as a reference for the overall noise level—as the depth increases, the circuit’s behavior can approximate that of a completely depolarizing channel, converging toward the MMS.	110
4.20	Average density matrix fidelity (left) and trace distance (right) throughout the training process of the RL agent on single-qubit circuits executed on a superconductive quantum chip. The metrics have been evaluated on a dataset of 60 circuits using 80% for the train set and 20% for the test set; shaded regions represent the standard deviation. Convergence was achieved after approximately 2×10^5 episodes, reaching an average fidelity of 0.99 on the test set. No evident signs of overfitting were observed.	112
4.21	Performance evaluation of different noise models on single-qubit Clifford circuits with depths ranging from 5 to 50. The performance is assessed using average DM fidelity (left) and trace distance (right), with respect to the real noisy DMs obtained from execution on quantum hardware. The RL agent is benchmarked against the RB method, the noiseless scenario, and the maximally mixed state (MMS). In the trace distance plot, a logarithmic scale is used to enhance the distinguishability between RB and RL. The lower level of noise, compared to the simulation, can be inferred from the fidelity with the MMS, which does not reach the same values as in the simulation. A comparison between the RL and RB models shows that the former outperforms the latter on average; however, due to large errors, their performances are comparable within one standard deviation.	113
4.22	QFT circuit for three qubits. In this circuit, $R_k = R_z(2\pi/2^k)$ and H denotes the Hadamard gate. The last SWAP gate can be omitted as this reordering can be handled classically. This circuit implementation has to be transpiled to native gates before being passed to the RL agent.	114

4.23	Grover's search algorithm circuit, the target state is $ 11\rangle$ and an ancillary qubit is required. In this configuration it is sufficient to interrogate the oracle only one time to obtain the correct result. The circuit has been transpiled to native gates before being used as input for the RL agent.	115
4.24	Computational basis states probabilities for the QFT circuit with a high error noise model (top), Grover's algorithm circuit with a low error noise model (mid) and a random non-Clifford circuit with a high error noise model (bottom). The histograms show a comparison between probabilities obtained with the ground truth noise model, the RL agent noise model and the RB noise model. It is possible to observe that while the RB model tends to flatten the distribution, the RL agent is able to reconstruct the peaks with more fidelity.	117
4.25	Heatmap of the absolute error between the ground truth noisy DM and the DM obtained with the RL agent (left) and RB model (right) for the three-qubit QFT circuit. It is possible to observe that the RL agent is able to better reconstruct the noisy DM not only in the diagonal part. This behavior has been observed in all the circuits tested in this section.	118
5.1	Hardware-efficient ansatz for a system of six qubits arranged in a ring topology with CNOT native gates. The circuit is composed of two layers of single-qubit rotations (R_x , R_y , R_z) and CNOT gates.	120
5.2	Schematic representation of a quantum autoencoder with three qubits. The encoder acts as a unitary transformation U that tries to disentangle two qubits and force them to the $ 0\rangle$ state. In this way, the initial quantum information is compressed into a latent quantum space. The decoder implements the inverse transformation $U^\dagger$ to reconstruct the original data. The loss function is taken as the measurement expectation value of the qubits that are eliminated after the compression.	127
5.3	Example of images for the "zero" and "one" digits for the MNIST handwritten digits dataset. Images are compressed to 8×8 pixels.	128
5.4	Circuit representation of the quantum encoder used for anomaly detection of handwritten digits. The encoder circuit is composed of six qubits and six layers. At the end of the quantum circuit the first three qubits are measured in order to compress information on the three remaining qubits. The loss function is computed as the sum of the probabilities of having any of the measured qubits in the ground state.	128
5.5	Quantum autoencoder loss function values distribution. The graph has been made using 2000 normal data images (zeros) and 2000 anomalous data images (ones) from the MNIST dataset.	129
5.6	A normal (left) and an anomalous (right) event used for quantum anomaly detection applied to long-lived particles detection. Normal and anomalous data are respectively prompt and displaced decays in multi-muons (from two to ten muons). Hits produced by the muons are reconstructed in a toy simulation of the ATLAS MDT chambers, that includes random hit background noise mimicking expected ATLAS phase-2 noise. Data are represented in the form of 100×20 pixels images.	130
5.7	Loss function values distribution for the quantum anomaly detection algorithm (left) and the classic counterpart (right). The graphs have been made using 2000 normal data images (decays that happen just as the particle enters the detector) and 2000 anomalous data images (decays that happen after the particle has travelled inside the detector).	132
5.8	ROC curve and AUC for the quantum anomaly detection algorithm (blue) and the classical counterpart (orange).	132

5.9	Architecture of IBM_hanoi quantum computer. The C-NOT connectivity is reported. The colours of single qubits and their connections represent respectively the single qubit readout assignment error and C-NOT error probabilities. Darker colours represent a lower error probability, in a range between 5.9×10^{-3} and 9.8×10^{-2} for readout error and 3.3×10^{-3} and 1 for C-NOT gates. Data from calibration on 19/10/2022.	133
5.10	Parametrized quantum circuit used for approximated amplitude encoding. The circuit is composed of four layers made of rotation gates and five C-NOT gates plus a final layer composed only of rotation gates.	134
5.11	Counts distribution of 2048 shots for a simulated circuit with no noise (top) a simulated circuit with realistic noise (center) and a noisy quantum circuit (bottom). . . .	135
5.12	Quantum autoencoder loss function values distribution. Simulated circuits with no noise (top left), simulated circuits with noise (top right) and a noisy quantum circuits (bottom). The histogram has been realized using 200 normal data and 200 anomalous data, with 2048 shots each circuit. The loss function is one minus the probability of the $ 111\rangle$ state.	137
5.13	ROC curves and AUC for anomaly detection, simulated circuits with no noise (orange), simulated circuits with noise (green) and noisy quantum circuits (blue). . . .	137
5.14	Representation of the bottleneck (left) and reverse-bottleneck (right) architectures with the measurement of one qubit. Each unitary transformation block in the image uses the layered structure reported in Figure 5.1.	142
5.15	Full quantum model samples.	145
5.16	Detail on the denoising process of a “zero” and a “one” digit for the full quantum model.	145
5.17	Unconditioned latent model (MNIST).	147
5.18	Unconditioned latent model (F.MNIST).	147
5.19	Conditioned latent model (MNIST).	147
5.20	Conditioned latent model (F.MNIST).	147
5.21	2D representation (t-SNE) of the original distribution (left) and generated distribution (right) in the latent space (8D). Each digit is generated separately by conditioning the model. The same t-SNE has been simultaneously performed on both the original and generated data points.	147
5.22	Architecture of IBM_hanoi quantum computer. The C-NOT connectivity is reported. The colors of single qubits and their connections represent respectively the $T1$ and C-NOT error probabilities. Darker colours represent a lower $T1$, in the range $[47, 265]\mu s$. For the C-NOT errors, darker colors represent a lower error rate, in the range $[3.2 \cdot 10^{-3}, 1]$. Calibration data are referred to 19/10/2023.	148
5.23	Quantum hardware adapted PQC. The central part of the circuit is repeated three times in total. State preparation is necessary only to encode the initial noise. Similarly, the state qubits are measured only at the end.	149
5.24	2D representation (PCA) of the latent vectors of the original distribution (KDE plots) and generated distribution obtained in simulation (circles). The original distribution is composed by all the “zeros” and “ones” handwritten digits of the MNIST dataset. The generated distribution contains $5 \cdot 10^3$ samples for each class.	150
5.25	State amplitudes of a generated latent vector obtained with: state vector simulation (a), shots simulation (b), realistic noise simulation (c) and hardware execution (d). . . .	151
5.26	Generated images of a “zero” and a “one” handwritten digits. The latent vector has been generated with both a noiseless state vector simulation (left) and with real quantum hardware (right) in order to study the effects of noise.	151

5.27	2D representation obtained with PCA of the latent vectors distribution obtained from the original MNIST dataset (KDE plots), data points generated with quantum hardware (stars) and with noiseless shots simulation (circles). The generated distributions contain 64 samples generated with $3.2 \cdot 10^5$ shots and starting from the same initial noise.	152
B.1	Schematic representation of a Josephson junction. Two superconductors, labeled S_A and S_B , are separated by a thin insulating barrier. The wavefunctions $\psi_A = \sqrt{N_A}e^{i\phi_A(t)}$ and $\psi_B = \sqrt{N_B}e^{i\phi_B(t)}$ describe the state of the Cooper pairs within each superconductor, where N_A and N_B are the numbers of electron Cooper pairs in each superconductive region. An applied voltage difference across the junction, indicated by $+V$ and $-V$, influences the phase difference between the superconductors.	195
C.1	Example of randomized benchmarking experiment. The survival probability P decays with the depth of the circuit d in the form: $P(d) = Ap^d + B$	199
I.1	Graphic representation of the dropout technique in a FFNN.	211

List of Tables

2.1	Feasibility of DiVincenzo criteria (five fundamental criteria plus two optional) across different quantum computing hardware. Criteria: (1) Qubit scalability, (2) Initialization, (3) Coherence time, (4) Universal gates, (5) Qubit measurement, (6) Inter-qubit communication, (7) Error correction. Each criterion is marked as high (H), medium (M), or low (L) based on the feasibility for each hardware type, these labels are indicative and intended to provide a general sense of the strengths and weaknesses of each hardware platform.	18
2.2	Summary of calibration procedures for superconductive qubits.	36
3.1	Truth table of a NAND gate obtained using a neuron with a sigmoid activation function and weights: $w_0 = 3$, $w_1 = w_2 = -2$. For inputs: <i>True</i> = 1, <i>False</i> = 0. For outputs: <i>True</i> if $\sigma(z) > 1/2$, <i>False</i> if $\sigma(z) < 1/2$	65
4.1	This table reports the optimal number of trainable weights (P) of the feed forward neural network obtained for different code distances (d) for a depolarising noise. . . .	89
4.2	This table reports best hyperparameters of the CNN for different code distances correcting a depolarising error model. d is the code distance, CL is the number of convolutional layers, DL is the number of dense layers, N is the number of neurons in the dense layers and P is the total number of trainable weights.	89
4.3	Best hyperparameters of the CNN for different code distances for a depolarising plus measurement errors model. d is the code distance, CL is the number of convolutional layers, DL is the number of dense layers, N is the number of neurons in the dense layers and P is the total number of trainable weights.	92
4.4	Noise models used in simulation to train and evaluate the RL algorithm in different conditions. The columns show the noise channels parameters: λ for the depolarizing channel, γ for the amplitude damping channel and θ_x and θ_z for the coherent error R_x and R_z respectively. The rows indicate the gates subject to the noise channel, a zero noise parameter means that the noise channel is not applied to that gate. . . .	109
4.5	Summary of the RL model performance across different training, test, and evaluation sets for single- and three-qubit circuits. Performance is reported as the average fidelity between the reconstructed and original density matrices, with standard deviation as the error. The training and test sets consist of Clifford circuits for the single-qubit case, and both Clifford and non-Clifford circuits for the three-qubit case. The evaluation set includes non-Clifford circuits of depth 15 in both cases. Fidelity on Clifford circuits of depth 30 is also reported as an alternative evaluation set.	111
4.6	Number of gates and circuit depth of the transpiled circuits for QFT and Grover's algorithm. A random circuit from the training set is also included as a benchmark to highlight the differences including depth and a higher fraction of two-qubit gates. . . .	115

4.7	Fidelity between the density matrix reconstructed with a noise model (RL agent, RB, the limit case where no noise channels are added and the MMS) and the ground truth noisy one. The result is reported for QFT and Grover's algorithm circuits for both a high and low noise models. The result obtained with a non-Clifford random circuit is also reported for comparison. The RL agent outperforms, or is comparable, with respect to the RB model in all the studied cases.	116
5.1	PQC characteristics of different models.	144
5.2	Evaluation metrics for QDMs on different datasets.	145
5.3	Evaluation metrics for classical diffusion models.	145
5.4	ROC-AUC values for each digit obtained with the latent conditioned model. Lowest and highest values are highlighted.	146

List of Acronyms

AE AutoEncoder.

AUC Area Under (ROC) Curve.

CNN Convolutional Neural Network.

DM Density Matrix.

DNN Deep Neural Network.

FFNN Feed Forward Neural Network.

GAN Generative Adversarial Network.

GPU Graphic Processing Unit.

HEP High Energy Physics.

HLD High Level Decoder.

KLD Kullback–Leibler Divergence.

LR Learning Rate.

MDP Markov Decision Process.

MDT Muon Drift Tube.

ML Machine Learning.

MSE Mean Squared Error.

MWPM Minimum Weight Perfect Matching.

NISQ Noisy Intermediate Scale Quantum.

NN Neural Network.

PCA Principal Component Analysis.

PPO Proximal Policy Optimization.

PQC Parametrized Quantum Circuit.

QCR Quantum Circuit Representation.

- QDM** Quantum Diffusion Model.
- QEC** Quantum Error Correction.
- QFT** Quantum Fourier Transform.
- QML** Quantum Machine Learning.
- RB** Randomized Benchmarking.
- RL** Reinforcement Learning.
- RMS** Root Mean Square.
- ROC** Receiver Operating Characteristic.
- SGD** Stochastic Gradient Descent.
- TD** Trace Distance.
- TII** Technology Innovation Institute.
- VAE** Variational AutoEncoder.