ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF PHYSICS AND ASTRONOMY ”A. RIGHI”

SECOND CYCLE DEGREE

PHYSICS

QUANTUM RESERVOIR COMPUTING ON PASQAL NEUTRAL ATOMS PLATFORM

Supervisor
Prof. Daniele Bonacorsi

Co-supervisor
Dr. Simone Gasperini

Defended by
Simone Angelozzi

Graduation Session September 2025

Academic Year 2024/2025

dedica

Abstract

In this thesis we investigate Quantum Reservoir Computing (QRC) as a non-variational paradigm for quantum machine learning on near-term neutral-atom devices. Motivated by the limitations of NISQ hardware, we implement a QRC pipeline in Pasqal's Pulser environment that encodes classical inputs into detuning waveforms and extracts embeddings from Pauli-Z and ZZ observables. We design two encoding strategies: a global detuning applied to all atoms and a local detuning. Their performance is benchmarked against a classical PCA baseline using logistic regression. We find that QRC embeddings consistently outperform the classical features, with peak test accuracies of 90.17% (global) and 89.61% (local) compared to $\simeq 85\%$ for PCA. The advantage is especially marked in low-data regimes, where quantum dynamics enrich the feature space accessible to linear classifiers. We also identify practical challenges, such as symmetry-induced redundancy in global encoding and the absence of per-qubit detuning in current hardware. This establishes QRC as a promising and experimentally grounded framework to exploit the dynamical richness of neutral-atom platforms for supervised learning in the NISQ era.

Contents

1 Introduction

The idea of using quantum computers to simulate and model natural phenomena has his roots in the famous article by Feynman [1]. In recent years significant progress has been achieved in the physical realization of quantum devices. Current machines operate in the so-called NISQ (Noisy Intermediate-Scale Quantum) regime, characterized by tens to hundreds of qubits, which makes classical simulation intractable while still suffering from noise and the absence of full error correction.
In parallel, the rapid advances in artificial intelligence have motivated increasing interest in the intersection between quantum computing and machine learning. These two fields are not independent: quantum devices have the potential to enhance several subroutines relevant for machine learning [2], while classical machine learning can be exploited to assist quantum computation [3]. This naturally motivates the exploration of hybrid classical–quantum approaches.
Among the different strategies, the most extensively studied paradigm is that of variational quantum algorithms (VQAs), which rely on parameterized quantum circuits optimized by a classical routine. Despite their success, these methods face important challenges such as barren plateaus in the optimization landscape, as well as limitations imposed by hardware noise.
More recently, an alternative approach called quantum reservoir computing (QRC) has been proposed. QRC is a non-variational model consisting of three layers: an input stage, where classical or quantum data are embedded into a quantum system; a quantum reservoir, which evolves according to a fixed unitary dynamics; and an output stage, where expectation values of observables are collected and processed by a simple linear model. The key advantage of QRC lies in the complex dynamics of the reservoir, which effectively maps the input data into a high-dimensional Hilbert space, enabling the extraction of rich nonlinear correlations. As a consequence, only a linear readout layer is required, and no training of the quantum evolution is necessary. This makes QRC inherently more robust to barren plateaus compared to variational approaches.

This thesis is organized as follows.

In the first chapter there is an introduction to the fundamental concepts of quantum computing, including qubits, unitary evolution, quantum operators, and the basic paradigms of quantum machine learning such as variational quantum algorithms and kernel methods.

The second chapter is devoted to the presentation of the principles of analog quantum computing with neutral atom systems, focusing on the Rydberg Hamiltonian and its experimental implementation. This chapter also describes the architecture of state-of-the-art neutral atom quantum computers, with particular attention to the platforms developed by QuEra and Pasqal.

The third chapter is focused on the study of quantum reservoir computing. After introducing the general idea of reservoir computing and its quantum extension, the chapter details the structure of the QRC algorithm, discussing encoding techniques, quantum evolution, data processing, and reporting results obtained on neutral atom hardware.

The final chapter presents the main contributions of this thesis, focusing on the implementation and evaluation of the Quantum Reservoir Computing (QRC) algorithm . It introduces the use of Pasqal's Pulser library to simulate and control the quantum device, describes the problem setup and design choices for the quantum reservoir, and reports the results obtained with different encoding strategies. The chapter highlights the performance improvements achieved with quantum-enhanced embeddings compared to classical linear methods, providing insights into the advantages, limitations, and potential applications of QRC.

2 Quantum Computing

2.1 Hilbert space and qubits

Any isolated quantum system is associated with a Hilbert space $\mathcal{H}$ known as the *state space* of the system. It is a complex vector space whose elements are called *state vectors*.
The state of the physical system with which we are dealing is described by a unit vector $|\psi\rangle$ belonging to the Hilbert space.
Being a vector space, the Hilbert space admits a basis of vectors such that any state in the space can be expressed as a linear combination of these basis elements. In the case of a finite-dimensional Hilbert space, a generic quantum state can thus be written as:

$$|\psi\rangle = \sum_{i=1}^{dim(\mathcal{H})} c_i|i\rangle \tag{2.1.1}$$

where the coefficients c_i are complex numbers and the states $|i\rangle$ are basis elements. The system is said to be in *superposition* whenever at least two of the coefficients c_i in the expansion are non-vanishing.
The simplest physical system is the *qubit*, whose state space is $\mathcal{H} \cong \mathbb{C}^2$. A qubit is a quantum system that, unlike a classical bit which can only be in one of two states (0 or 1), can be in a superposition of $|0\rangle$ and $|1\rangle$, allowing it to represent any linear combination of these states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{2.1.2}$$

The states $|0\rangle$ and $|1\rangle$ are known as computational basis states and form an orthonormal basis of $\mathcal{H}$. Therefore, the state $|\psi\rangle$ must be a unit vector, $\langle\psi|\psi\rangle = 1$, which implies that $\alpha, \beta \in \mathbb{C}$ must satisfy:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.1.3}$$

Because $|\alpha|^2 + |\beta|^2 = 1$, we may rewrite Eq.(1.2) as

$$|\psi\rangle = e^{i\gamma}\Big(cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle\Big), \tag{2.1.4}$$

where θ, φ, and γ are real numbers. By considering measurements and observables, as they will be introduced in the next sections, we can drop the factor $e^{i\gamma}$ in front, since it has no observable effects.
For this reason, we can rewrite the state of the qubit as:

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle. \tag{2.1.5}$$

This expression suggests an intuitive way to think about the qubit: a point on a three-dimensional sphere, known as the Bloch sphere, see Fig. 2.1.1.

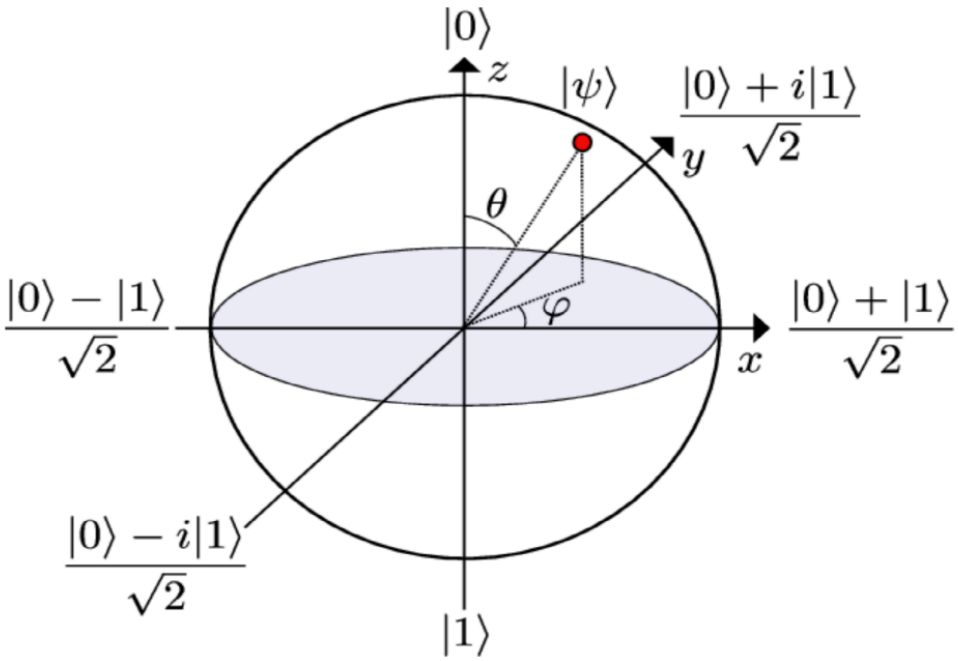

Figure 2.1.1: Bloch sphere representation of a qubit. The red dot indicates the current state $|\psi\rangle$, identified by the parameters θ and φ.

The angles θ and φ define a point on the unit-radius sphere, which represents the state of the qubit. The poles of the sphere correspond to the computational basis states. Fig. 2.1.1 also highlights the presence of other relevant states, which turn out to be the eigenvectors of the Pauli operators, on which we will focus in the next section.

In order to perform useful quantum computations, it is essential to consider systems composed of multiple qubits. The state of a multi-qubit system is described by the tensor product of the individual qubit states. If we denote the state of the i-th qubit as $|\psi_i\rangle$, then the overall state of an n-qubit system is given by

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots |\psi_n\rangle. \tag{2.1.6}$$

This formalism naturally extends the superposition principle to composite systems: if a two-qubit system can be in states $|\psi_1\rangle \otimes |\psi_2\rangle$ and $|\psi_1'\rangle \otimes |\psi_2'\rangle$, then any linear combination

$$a(|\psi_1\rangle \otimes |\psi_2\rangle) + b(|\psi_1'\rangle \otimes |\psi_2'\rangle) \tag{2.1.7}$$

is also a valid physical state. However, not all states of a multi-qubit system can be written as a simple tensor product of single-qubit states. For instance, the two-qubit state

$$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.1.8}$$

cannot be expressed as $|\psi\rangle \otimes |\phi\rangle$ for any single-qubit states $|\psi\rangle$ and $|\phi\rangle$.
Such states are called *entangled states*. Entanglement is a unique quantum feature and plays a

central role in quantum computing and quantum information. It enables correlations between qubits that cannot be explained classically, allowing quantum computers to perform operations and process information in ways that are fundamentally more powerful than classical systems.

2.2 Unitary evolution

So far, we have seen how to describe the state of a quantum system. However, in order to fix the mathematical framework of quantum mechanics we need to understand how the state, $|\psi\rangle$, change with time.
The evolution of a closed quantum system is described by a *unitary transformation*. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2 [4]

$$|\psi(t_2)\rangle = U(t_2, t_1)|\psi(t_1)\rangle. \tag{2.2.1}$$

In quantum computation there exists a set of particularly important unitary operators acting on a single qubit.
Among these, the three most fundamental ones are those whose eigenstates correspond to the extreme points along the axes of the Bloch sphere. Fig. 2.1.1:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.2.2}$$

The operator X, often referred to as the quantum analog of the classical NOT gate, flips the computational basis states: it maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. The operator Z leaves $|0\rangle$ invariant while introducing a phase flip to $|1\rangle$, mapping it to $-|1\rangle$. The operator Y combines a bit flip and a phase flip.
Geometrically, these operators are closely related to the orientation of the qubit on the Bloch sphere: X, Y, and Z correspond to rotations and measurements along the x, y, and z-axes, respectively. Their eigenstates define preferred measurement bases, which are crucial for interpreting the outcome of quantum measurements.

The mathematical form of the unitary U in Eq.2.2.5 can be recovered leveraging the equation of motion of quantum mechanical systems, that is, the *Schrödinger's equation*.

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H|\psi(t)\rangle \tag{2.2.3}$$

In this equation, $\hbar$ is the reduced Planck's constant, H is the *hamiltonian* of the system and, since it is an hermitian operator, has a spectral decomposition

$$H = \sum_{\epsilon} E|\epsilon\rangle\langle\epsilon|, \tag{2.2.4}$$

with eigenvalues E and corresponding orthonormal eigenvectors $|\epsilon\rangle$.
Solving the Schrödinger's equation one gets

$$U(t_1, t_2) = e^{-\frac{i}{\hbar}H(t_2 - t_1)} \tag{2.2.5}$$

with

$$|\psi(t_2)\rangle = e^{-\frac{i}{\hbar}H(t_2-t_1)}|\psi(t_1)\rangle \tag{2.2.6}$$

being the explicit expression of Eq.2.2.5.
A remarkable point to stress is that any unitary operator one may think of can be always written as an imaginary exponential of a given hamiltonian.

2.3 Quantum operators

Up to this point, we have described the dynamics of a quantum system assuming it is isolated. We introduced the state space of a quantum system, defined its state at a given time, and discussed its unitary evolution within a closed system. However, to extract information about the system, an interaction through a measurement is required, which inevitably opens the system and causes its evolution to no longer be purely unitary.

2.3.1 Measurement in Quantum Mechanics

In quantum mechanics measurements are described by a collection $\{M_m\}$ of *measurement operators*. These operators, acting on the state space of the system being measured, satisfy

$$\sum_m M_m^\dagger M_m = \mathbb{I}. \tag{2.3.1}$$

The index m refers to the measurement outcomes that may occur in the experiment.
When measuring a quantum system with state $|\psi\rangle$, the probability of getting m as outcome is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle, \tag{2.3.2}$$

and the state of the system after the measurement is

$$|\psi\rangle \to \frac{M_m|\psi\rangle}{\sqrt{p(m)}}. \tag{2.3.3}$$

Equation 2.3.1 implies that the probabilities add up to one, forming a valid probability distribution

$$\sum_m p(m) = \langle\psi|\sum_m M_m^\dagger M_m|\psi\rangle = \langle\psi|\psi\rangle = 1. \tag{2.3.4}$$

The simplest example of a measurement is the measurement of a qubit on the computational basis. The observable in this case would be the Pauli matrix $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$. The two measurement operators are then

$$P_{+1} = M_{+1}^\dagger M_{+1} = |0\rangle\langle 0|, \quad P_{-1} = M_{-1}^\dagger M_{-1} = |1\rangle\langle 1|. \tag{2.3.5}$$

If the general qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is measured, then the probability of having the outcome $+1$ is

$$p(+1) = \langle\psi|P_{+1}|\psi\rangle = \langle\psi|0\rangle\langle 0|\psi\rangle = |\alpha|^2, \tag{2.3.6}$$

and similarly the probability of getting the output -1 is $p(-1) = |b|^2$.
Therefore, the state after the measurement is

$$\frac{|0\rangle\langle 0||\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|}|0\rangle = |0\rangle, \tag{2.3.7}$$

$$\frac{|1\rangle\langle 1||\psi\rangle}{|\beta|} = \frac{\beta}{|\beta|}|1\rangle = |1\rangle, \tag{2.3.8}$$

where we have used that the global phase of a quantum state can be ignored [4].

In a more general framework we can consider measuring several qubits in order to gain information of correlation within the system. An important example is the two-qubit operator Z_iZ_j, defined as the tensor product of Pauli-Z operators acting on qubits i and j:

$$Z_iZ_j = \mathbb{I} \otimes \cdots \otimes Z \otimes \mathbb{I} \otimes \cdots \otimes Z \otimes \cdots \otimes \mathbb{I}, \tag{2.3.9}$$

where the Z operators are placed at the i-th and j-th positions.
Given this structure for the operator this means that will leave invariant the states at $k \neq i, j$ while measuring the Pauli Z matrix at the proper positions:

$$Z_iZ_j|\Psi\rangle = |\psi_1\rangle \otimes \cdots \otimes Z|\psi_i\rangle \otimes \cdots \otimes Z|\psi_j\rangle \otimes \cdots \otimes |\psi_n\rangle, \tag{2.3.10}$$

for a system of n qubits.
The operator Z_iZ_j measures the *correlation* between qubits i and j along the z-axis. If the two qubits are in the same state (both $|0\rangle$ or both $|1\rangle$), the measurement outcome is $+1$; if they are in opposite states , the outcome is -1.

2.3.2 Quantum gates and universality

Once defined the basic unit of quantum computation, i.e. the qubit, we point our attention to the backbone of a quantum computer: the quantum circuits.
Classical computers are based on bits, wires and logical gates that enable information manipulation. In analogy quantum computers are formed by quantum circuit based on qubits and quantum logic gates.

Single qubit gates

Single-qubit gates are the most basic quantum operations. Operations involving qubits must preserve the norm, and are therefore represented by 2×2 unitary matrices. The Pauli matrices, discussed in Section 1.2, correspond to rotations on the Bloch sphere by an angle π around the x, y, and z-axes, respectively. Another essential quantum gate is the Hadamard gate, which allows for the creation of a superposition between $|0\rangle$ and $|1\rangle$.

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.3.11}$$

$$H|0\rangle = |+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{2.3.12}$$

$$H|1\rangle = |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2.3.13}$$

Another set of fundamental gates is the *phase gates*. These are parametrized gates since one needs to specify the angle $\varphi \in \mathbb{R}$ to fully define the gate.

$$P = \begin{bmatrix} 1 & 0 \\ 1 & e^{i\varphi} \end{bmatrix} \tag{2.3.14}$$

Given this definition we notice that the Z gate is a phase gate with $\varphi = \pi$. Other relevant gates are S and T, which are obtained by fixing $\varphi = \pi/2$ and $\varphi = \pi/4$ respectively.

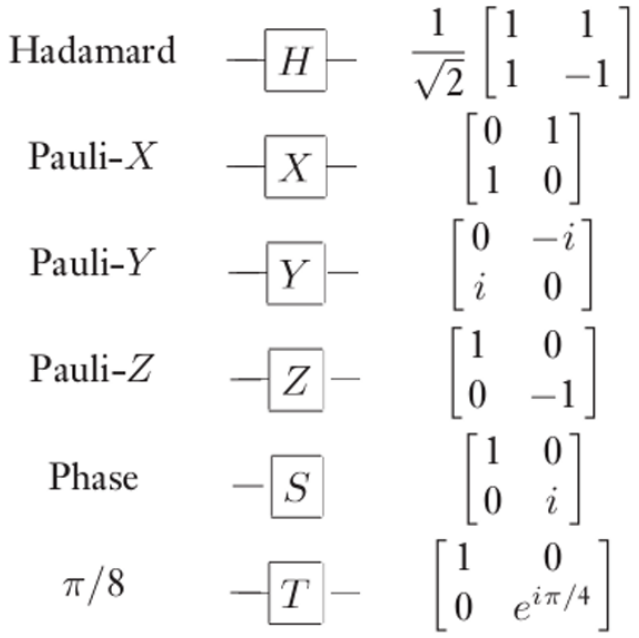

Figure 2.3.1: Symbols for the most common single-qubit gates.

Two qubits gates

When dealing with more than one qubits, more complex gates are needed. These gates are useful because they allow for the creation of entangled states, which are pivotal in quantum algorithms. Those gates are *controlled operations*, the most general form of a two qubits gates consist in a 4×4 matrix.
The most relevant is the CNOT which takes two input qubits, known as the *control* and *target* qubits. If the control qubit is $|1\rangle$, then the target qubit is flipped, leaving the control qubit intact.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.3.15}$$

Universality

In classical computation, Boolean functions are expressed using gates such as AND, OR, NAND, NOT, etc. The NAND gate is called universal because it can be combined to form NOT, OR, and AND gates, and from these, all other operations can be constructed. For this reason, the NAND gate is said to be universal.
A similar result can be achieved in quantum computation. If a set of quantum gates can approximate any unitary operation with arbitrary accuracy, we can say that they are universal for quantum computation.
A possible set of gates can be derived from the assumption that any unitary operator can be exactly expressed as the product of a CNOT gate and a single-qubit gate. The latter can be approximated by the Hadamard, phase, and $\pi/8$ gates, for proof see [Nielsen and Chuang]. Thus, we can conclude that any unitary operation can be approximated using the Hadamard, phase, $\pi/8$, and CNOT gates.

2.4 Quantum Machine Learning

The most general definition of Quantum Machine Learning is as a research area focused on exploring the relationship between classical Machine Learning and Quantum Computing, with the aim of advancing both fields through mutual insights and innovation.
In order to clarify the meaning of QML in this work we will focus on four approaches to combine quantum computing and machine learning (see Fig. 2.4.3).

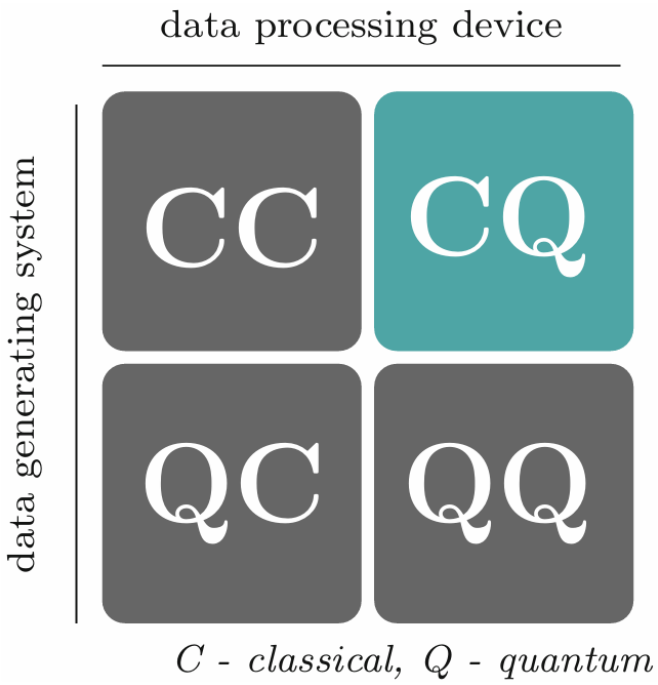

Figure 2.4.1: The first letter is associated with the system that generates the data, quantum (Q) or classical (C), the second letter stands for the information processing device, quantum (Q) or classical (C) [5].

1. **CC** refers to classical data being processed classically. This is the conventional approach to machine learning, but in this context it relates to machine learning based on methods borrowed from quantum information research. An example is the application of tensor networks, which have been developed for quantum many-body-systems

2. **QC** investigates how machine learning can help with quantum computing. For example, when we want to get a comprehensive description of the internal state of a quantum computer from as few measurements as possible we can use machine learning to analyze the measurement data

3. **CQ** uses quantum computing to process classical datasets. The datasets consist of observations from classical systems, such as text, images or time series, which are fed into a quantum computer for analysis.
   The central task of the CQ approach is to design quantum algorithms for data mining

4. **QQ** looks at 'quantum data' being processed by a quantum computer. The most powerful approach comes when a quantum computer is first used to simulate the dynamics of a quantum system and consequently takes the state of the quantum system as an input to a quantum machine learning algorithm [5].

In this work we will focus on **CQ** approach, applying a quantum algorithm to classical data.

A quantum machine learning model deals with data, and thus is necessary to find a way to encode such information onto the quantum computer. Suppose we are given a set of classical data $\{x\}_{i=1}^{n}$, the idea is to map this data to the Hilbert space of the quantum computer, via a procedure known as *feature embedding*.
This is done by using a parametrized unitary operation $U(\mathbf{x})$ so that the feature embedding consist of the map

$$\phi : \mathcal{X} \to \mathcal{H}, \quad \mathcal{X} \subset \mathbb{R}^d, \quad \mathcal{H} \cong \mathbb{C}^{2n} \tag{2.4.1}$$

$$\mathbf{x} \to |\phi(\mathbf{x})\rangle = U(\mathbf{x}) \, |0\rangle^{\otimes n} \tag{2.4.2}$$

The parameterized block $U(.)$ that actually maps the data to a quantum state is general, and various ansatze can be used to accomplish this task.

After the encoding phase, the chosen algorithm is ready to be implemented on the quantum system. A unitary matrix V is applied to the system, causing the initial quantum state to evolve in accordance with the dynamics defined by the algorithm. As we will explore in the subsequent sections, there is a wide variety of quantum algorithms, each tailored to perform distinct tasks.
The final step involves probing the quantum computer by measuring the system's state, thereby obtaining the output, which is crucial for extracting useful information and making inferences based on the quantum computation.

2.4.1 Variational Quantum Algorithm

Due to the current limitations in building fault-tolerant quantum computers, fully quantum algorithms remain out of reach in the near term. One of the most studied approaches are hybrid classical-quantum algorithms such as Variational Quantum Algorithms (VQAs), emerged as a

promising framework to exploit the capabilities of noisy intermediate-scale quantum (NISQ) devices.
The key idea behind these algorithms is to use a quantum and a classical device in cooperation to compute the value of an objective function $C(\theta)$, given a set of classical parameters θ. A classical optimization routine is then used to update the parameters by iteratively querying the quantum device [5].

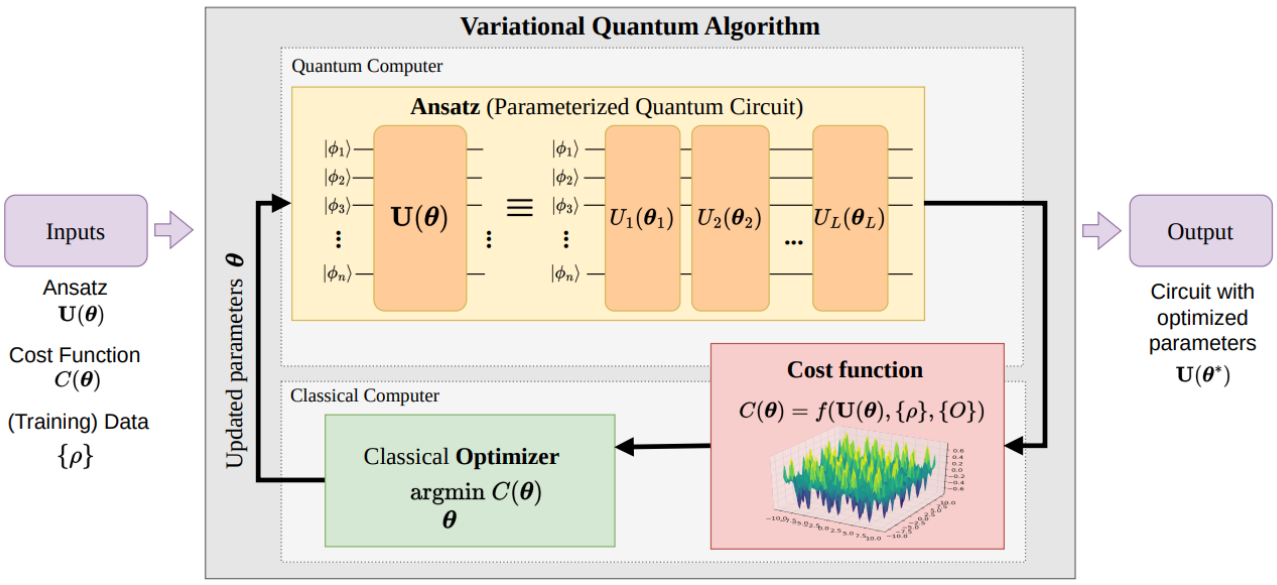

Figure 2.4.2: Components of a Variational Quantum Algorithm [6].

As shown in Fig. 2.4.2, the structure of the algorithm closely mimics that of a neural network. The training data $|\phi_i\rangle$ are encoded into the quantum state, and a parameterized quantum circuit (ansatz) $U(\theta)$, typically structured in layers, evolves the system. The quantum device produces an output that is processed by the classical optimizer, which updates the variational parameters. These updated parameters are fed back into the quantum circuit, and the process repeats until convergence is achieved.
This class of algorithms is promising for quantum machine learning applications, although they also face significant challenges, which will be discussed in the following sections.

2.4.2 Quantum Kernel Methods

Kernel methods compute distance between data point x, the training input, and x' the new input we aim to classify through a kernel $k(x, x')$. Such a kernel corresponds to an inner product of data points mapped to a higher dimensional feature space.
In the previous section we already faced some hints about kernels, that is input encoding.
Kernel methods compute a similarity measure between input data points x and x' through a kernel function $\kappa(x, x')$. For every positive semi-definite kernel, there exists an associated feature space $\mathcal{F}$, induced by a nonlinear feature map $\phi(x)$, such that the kernel corresponds to the inner product in that space:

$$\kappa(x, x') = \langle \phi(x), \phi(x') \rangle.$$

In this framework, one can understand the kernel function as defining a notion of distance in the input space. When used in classification tasks, the kernel allows for computing similarities between a new input and each training input, ultimately favoring the class of training data that is "closer" to the new point in the feature space. Let us consider an input domain $\mathcal{X}$, a positive semi-definite kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and a dataset $\mathcal{D} = \{(x_m, y_m)\} \subset \mathcal{X} \times \mathbb{R}$. We define a class of functions $f : \mathcal{X} \to \mathbb{R}$ which can be written in the form

$$f(x) = \sum_{l=1}^{\infty} \mu_l \kappa(x, x_l),$$

with $\mu_l \in \mathbb{R}$, $x_l \in \mathcal{X}$, and $\|f\| < \infty$. A cost function $C : \mathcal{D} \to \mathbb{R}$ evaluates the quality of a model by comparing predicted outputs $f(x_m)$ with targets y_m, and includes a regularization term $g(\|f\|)$ with $g : [0, \infty) \to \mathbb{R}$ strictly increasing. The solution to such a regularized optimization problem lies in the span of the kernel functions evaluated at the training data. That is, any minimizer f of the cost function admits a representation of the form:

$$f(x) = \sum_{m=1}^{M} \nu_m \kappa(x, x_m),$$

where M is the number of training sample [5].

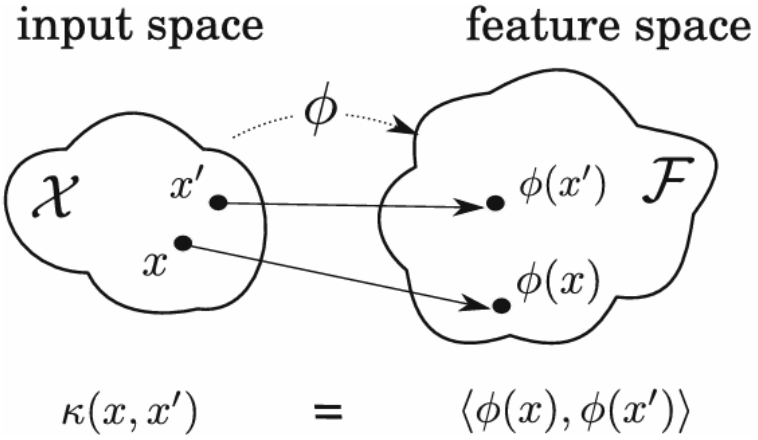

Figure 2.4.3: Relation between quantum feature map and quantum kernel [5].

This naturally extends to the quantum setting.
A quantum kernel is defined by interpreting the encoding of classical data $x \in \mathcal{X}$ into quantum states $|\phi(x)\rangle$ as a quantum feature map. The quantum kernel is then simply the inner product between these quantum feature states:

$$\kappa(x, x') = \langle \phi(x) | \phi(x') \rangle.$$

Any quantum computer capable of estimating such inner products can be used to compute a quantum kernel. If this kernel is classically intractable to simulate, then quantum advantage may be achieved. In this setting, the only quantum component is the state preparation and inner

product estimation, while the rest of the learning algorithm remains entirely classical.
This hybrid approach opens a promising direction for quantum machine learning, provided that one can design useful feature maps $\phi(x)$ that exploit the high-dimensional structure of quantum Hilbert spaces while being hard to replicate classically.

2.5 Quantum Computing Paradigm

Since the famous article by Feynman, significant progress has been made, leading to the development of Noisy Intermediate-Scale Quantum (NISQ) devices, capable of controlling up to a few hundred qubits [7]. However, these devices are inherently noisy and lack error correction, which limits the depth of quantum circuits that can be reliably executed. In this context, two main paradigms of quantum computing have emerged: the digital and the analog.

Digital quantum computing

follows a circuit-based model where quantum information is processed through discrete unitary gates, much like in classical digital logic. Fault tolerance in this paradigm is achieved via reliable quantum error correction codes, such as the surface code, which in principle enable arbitrarily long computations, but require a large overhead in terms of physical qubits. As of today, fault-tolerant quantum computing remains a long-term goal, due to the current limitations in qubit quality and control.

Analog quantum computing

Another approach to simulating quantum systems by quantum mechanical means is analog quantum computing, in which one quantum system mimics another.
The Hamiltonian of the system to be simulated, H_{sys}, is directly mapped onto the Hamiltonian of the simulator, H_{sim}, which can be controlled:

$$H_{\text{sys}} \rightarrow H_{\text{sim}} \tag{2.5.1}$$

An important advantage of analog quantum computing is that it could be useful even in presence of errors, up to a certain tolerance level [8]. Analog quantum computation have demonstrated useful applications in simulating quantum many-body physics and optimization problems in the NISQ regime.
The choice between digital and analog paradigms is often determined by the nature of the task and the available hardware.
In 2000, DiVincenzo [9] proposed a set of foundational criteria to evaluate the physical feasibility of platforms for quantum information processing. Later, in 2012, Cirac and Zoller [10] introduced an analogous framework tailored specifically to quantum simulators, outlining the essential requirements such systems should meet (see Table 2.5 for a summary).
In this work, we will focus on analog quantum computation using a system based on Rydberg-interacting atoms which fulfill both sets of criteria, making them a very attractive platform for quantum computation.

Criteria	**Quantum computers**	**Quantum simulators**
Quantum system	A scalable physical system with well characterized qubits	A system of quantum particles (bosons, fermions, pseudo-spins) confined in space and collectively possessing a large number of degrees of freedom
Initialization	The ability to initialize the state of the qubits to a simple fiducial state, such as $\|000\cdots\rangle$	The ability to prepare (approximately) a known quantum state (typically a pure state)
Coherence	Long relevant decoherence times, much longer than the gate operation time	
Interactions	A "universal" set of quantum gates	An adjustable set of interactions used to engineer Hamiltonians/quantum master equations including some that cannot be efficiently simulated classically
Measurement	A qubit-specific measurement capability	The ability to perform measurements on the system; either individual particles or collective properties
Verification		A way to verify the results of the simulation are correct

Table 2.1: Comparison between the key criteria that define a universal quantum computer and a quantum simulator. While both platforms exploit quantum systems to process information, quantum computers are designed to be scalable, programmable and capable of performing arbitrary algorithms, whereas quantum simulators are tailored to reproduce specific models of interest with controllable interactions.

3 Analog Quantum Computing

In the previous chapter, we introduced the two main paradigms of quantum computing: digital and analog.
We now turn our attention to analog quantum computing, which constitutes the core computational model used in this work.
In what follows, we provide a more in-depth discussion of its principles, mathematical formulation, and physical realizations with a special focus on neutral atom platforms, such as those developed by QuEra and Pasqal.

3.1 Principles of Analog Quantum Computing

Analog quantum computing is based on the idea of directly implementing the Hamiltonian evolution of a quantum system on a controllable experimental platform [11]. In practice, continuous physical parameters of the simulated system (such as interaction strengths) are exposed as programmable variables in the analog quantum device. As a result, an analog quantum program consists of continuously modulating the system's Hamiltonian over time, letting the initial quantum state evolve according to the Schrödinger equation. This approach stands in contrast to the digital paradigm, where the problem is compiled into a discrete sequence of quantum logic gates. In other words, analog computation operates directly on the system's Hamiltonian rather than through the composition of discrete gate operations.

From a formal perspective, analog quantum simulation requires that the Hamiltonian of the simulator, H_{sim}, has to be related in a known way to the Hamiltonian of the target system, H_{sys}. This is often referred to as a direct mapping of the problem Hamiltonian onto the simulator hardware. In such cases, the analog device mimics the target model by implementing a local, controllable Hamiltonian that reproduces the behavior of the physical system under investigation [12].
For instance, if U is a suitable unitary transformation, one may require:

$$H_{\text{sim}} = U H_{\text{sys}} U^{-1} \tag{3.1.1}$$

so that the evolution of a state $|\phi(0)\rangle$ of the system corresponds to the evolution of $|\psi(0)\rangle = U|\phi(0)\rangle$ on the simulator. After time evolution ruled by H_{sim}, the final state of the simulator $|\psi(t)\rangle$ yields the system's final state via $|\phi(t)\rangle = U^{-1}|\psi(t)\rangle$. This allows one to prepare physically relevant quantum states directly on the device and to measure local observables without decomposing the evolution into gate sequences [8].

Operationally, the controlled dynamics in an analog quantum computer are achieved by tuning the system's Hamiltonian parameters—such as voltages, magnetic fields, or laser pulses—rather than

by applying discrete logic gates. In certain platforms, like Rydberg atom quantum computers, this corresponds to applying global pulses that simultaneously modulate the interactions among all qubits [11]. The system evolves according to the time-dependent Schrödinger equation:

$$i\hbar\partial_t |\psi(t)\rangle = H(t) |\psi(t\rangle \tag{3.1.2}$$

where the Hamiltonian $H(t)$ is continuously programmable.
This analog quantum paradigm offers significant advantages in the NISQ era.
First, analog devices are typically tailored to narrower, physically-motivated problem—such as local condensed matter models—thus requiring less demanding hardware than universal digital quantum computers. In particular, they avoid the computational overhead of gate decomposition: no Trotterization is needed, which drastically reduces the number of quantum operations and associated errors. Moreover, the observables of interest (e.g., local observables or short-range correlations) are typically intensive and local, and therefore more robust to noise than the full quantum state. This means that even in the presence of significant decoherence or experimental noise, analog simulators can still produce reliable measurement outcomes [12].

In summary, the analog quantum computing paradigm can be formally defined as the use of a controllable quantum system with continuous Hamiltonian dynamics to simulate another quantum system. Unlike digital quantum computing, this model exploits the native time evolution of the system's Hamiltonian. These devices are not necessarily universal—they do not implement arbitrary algorithms—but they aim to accurately replicate specific physical models. In today's NISQ context, analog quantum computing offers a promising complementary approach: lower overhead and higher noise resilience for specific physical tasks, at the expense of general-purpose flexibility. Formal definitions and validation protocols in literature [13] provide a rigorous conceptual framework for the ongoing development of this class of quantum devices.

3.2 Neutral Atom-Based Models

In the last two decade incredible progress has been made in studying quantum computation, characterizing different and scalable platforms. One of the most promising platform for quantum computation is the neutral atom-based model.
Beginning with the pioneering work in [14] proposing neutral atoms as a quantum computing platform several proof-of-concept experiments with Rydberg atoms as qubit has been performed. Key milestones include works that achieve deterministic loading of large neutral-atom qubit arrays using optical tweezers [15], and the observation of non-equilibrium dynamics in a one-dimensional chain of 51 atoms [16], showcasing the platform's potential as an analog quantum computer.

In the following sections, we will describe the working principles of a quantum computer based on neutral atoms, starting from its Hamiltonian and providing a detailed explanation of each of its constituent terms.

3.2.1 Rydberg Hamiltonian

At the core of analog quantum computation there is the dynamics induced by the Hamiltonian and how a state evolve through out time.

The Hamiltonian of a typical neutral atom quantum computer is

$$H(t) = \frac{\Omega(t)}{2} \sum_i e^{i\phi(t)} |g_i\rangle \langle r_i| + e^{-i\phi(t)} |r_i\rangle \langle g_i| - \Delta(t) \sum_i n_i + \sum_{i<j} \frac{C_6}{|\mathbf{x}_i - \mathbf{x}_j|^6} n_i n_j \tag{3.2.1}$$

In this work we use $|g_i\rangle \equiv |0\rangle$ and $|r_i\rangle \equiv |1\rangle$ as representations of the ground and excited state of the i-th qubit. $n_i = |r_i\rangle \langle r_i|$ counting Rydberg excitations.
Furthermore, there are four parameters to control the quantum evolution:

- $\Omega(t)$: Rabi drive amplitude. Defines the frequency at which an individual qubit transitions between its ground and excited state.
- $\phi(t)$: Rabi drive phase. Determines the axis of rotation on the Bloch sphere around which the qubit undergoes evolution.
- $\Delta(t)$: detuning parameter. Quantifies the mismatch between the drive frequency and the qubit's natural transition frequency.
- $\mathbf{x}_i$: atom position in the array. Influences the interaction energy between qubits.

3.2.2 Qubits and register

A neutral atom quantum processor are based on configurable arrays of single neutral atoms. The array can be seen as a register where each atom is precisely positioned in space with separations of a few micrometers, each playing the role of a qubit.
A commonly used option is rubidium atoms, a widely studied species in atomic physics that benefits from mature technological solutions, especially in laser systems.
Specifically, two electronic energy levels of the rubidium atom are selected to serve as the qubit states, denoted as $|0\rangle$ and $|1\rangle$.
The most used way to arrange atoms in this programmable arrays is by laser cooling the atoms and then trapping them in optical micro-trap generated by a spatial light modulator or digital micro-mirror device, see Fig. 3.2.1.
This setup for constructing a neutral atom QPU not only allows each trap site to be filled with exactly one atom—thanks to light-assisted collisions—but, remarkably, it also provides the flexibility to reconfigure the spatial arrangement of atoms in the register after each processing cycle, unlike superconducting devices where the qubit topology is fixed. These techniques have been used to realize deterministically highly complex quantum register structure, assembled in different geometry including 3D arrays, see Fig. 3.2.2.
Once defined the structure of the atomic register, the next step is to initialize the system in a desired state. Due to the fact that in a given array the same atomic species, and thus, the energy levels are all identical the state of the system can be initialized in a well defined state (e.g., $|0\rangle^{\otimes N}$).
Once prepared, these states exhibit significant stability. the qubit energy splittings—on the order of several hundred megahertz—are much larger than both the typical strengths of external field couplings and the thermal energy scale $k_B T/h \lesssim 1\text{MHz}$.
Rydberg states generally have lifetimes around 100 microseconds, which is substantially longer than both the scales of Rydberg-Rydberg interactions and typical gate durations $t_g = (0.05-6)\mu\text{s}$.
In contrast, ground-state atoms confined in a trap can remain stable for several seconds.

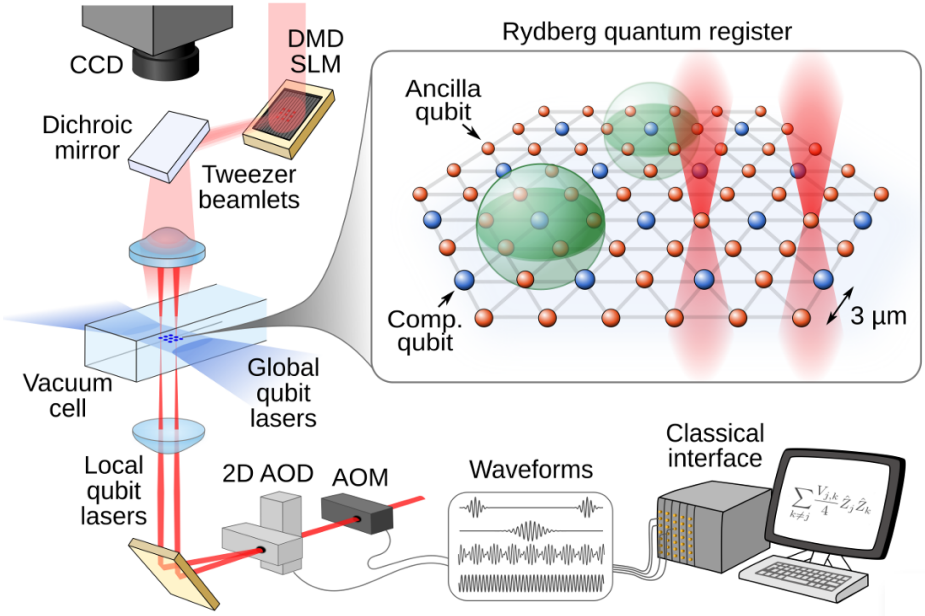

Figure 3.2.1: Sketch of a typical setup consisting of ultra-cold atoms trapped in an array of optical tweezers produced by a digital micro-mirror device (DMD) or spatial light modulator (SLM). Qubits can be manipulated by optical fields controlled by acousto-optical modulators (AOMs) and two-dimensional acousto-optical deflectors (AODs). The quantum register shows two species of atomic qubits (blue and orange spheres). Semi-transparent green spheres depict the Rydberg-Rydberg interactions (blockade spheres). Red shaded areas depict the addressing lasers for implementing single and multi-qubit operations [17].

3.2.3 Single qubit manipulation

Transitions between different electronic states are implemented by absorption and emission of photons, thus, the best way to manipulate Rydberg qubits is using laser fields or combined laser and microwave fields.
An atom interacting with light whose energy matches a specific atomic transition can absorb or emit a photon, allowing it to move between ground and excited states through quantum processes. For this interaction to remain coherent, the electromagnetic field of the photons must be extremely stable in frequency—within about 10 kHz, corresponding to a precision better than one part in 10^{11}. This level of stability is achieved using ultra-stable lasers that are frequency-locked to an optical cavity. One of the main challenges in neutral-atom quantum computing lies here: achieving high-fidelity operations requires lasers that are not only exceptionally stable, but also capable of delivering high power at the same time.
In what follows we briefly describe the underlying physical model of interactions among Rydberg qubits and a monochromatic field, in order to explain many of the terms present in Eq.3.2.1.

Atomic Hamiltonian in presence of a laser

Let us examine an atom characterized by a Hamiltonian involving two electronic energy levels, denoted by $|\alpha\rangle$ and $|\beta\rangle$. The Hamiltonian includes two main contributions: the first arises from the interaction between the atomic dipole moment d and a time-dependent electric field $E(t) =$

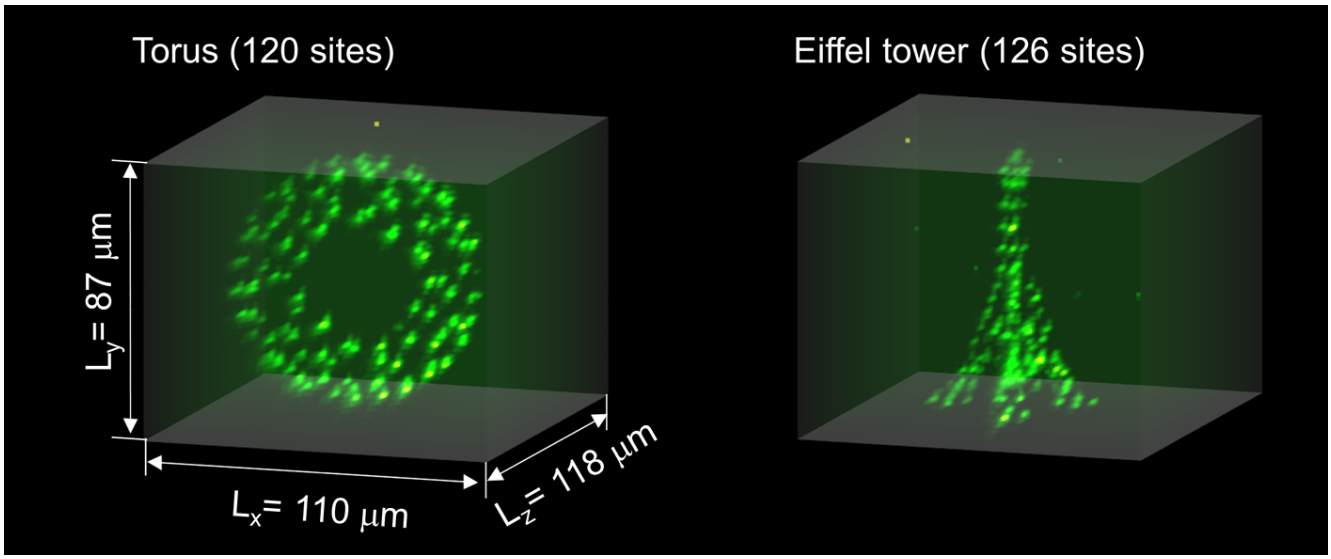

Figure 3.2.2: Examples of neutral atom arrays in 3D geometries [18].

$E_0\cos(\omega_0 t + \phi)$, with angular frequency $\omega_0 = 2\pi f = 2\pi c/\lambda$. The interaction term represented by the product of the electric field and the dipole moment, is commonly written as $-E_0 d = \Omega$, the Rabi frequency. The second term corresponds to the energy difference between the two states ω_α and ω_β.

$$H(t) = \Omega(|\alpha\rangle\langle\beta| + |\beta\rangle\langle\alpha|)\cos(\omega_0 t + \phi) + (\omega_\beta - \omega_\alpha)|\beta\rangle\langle\beta| \tag{3.2.2}$$

This Hamiltonian can be simplified assuming $|\omega_\beta - \omega_\alpha| = \omega_0 > \Omega$ and moving in the rotating frame of reference $|\beta\rangle \to e^{i\omega_0\beta t}|\beta\rangle$, which adds an internal diagonal term to the Hamiltonian

$$H(t) = \Omega(e^{-i\omega_0 t}|\alpha\rangle\langle\beta| + e^{i\omega_0 t}|\beta\rangle\langle\alpha|)\cos(\omega_0 t + \phi) + (\omega_\beta - \omega_\alpha - \omega_0)|\beta\rangle\langle\beta| \tag{3.2.3}$$

Next, we use the identity $2\cos(\omega_0 t + \phi) = e^{i\omega_0 t + i\phi} + \text{h.c.}$ to expand

$$H(t) = \frac{\Omega}{2}\left(e^{i\phi} + e^{-i2\omega_0 t - i\phi}\right)|\alpha\rangle\langle\beta| + \text{h.c.} + (\omega_\beta - \omega_\alpha - \omega_0)|\beta\rangle\langle\beta| \tag{3.2.4}$$

Noticing that the exponential functions are sums or differences of different frequencies, one can choose $\omega_\alpha - \omega_\beta + \omega_0 = 0$ to set one at zero frequency, in other terms the second exponential is $e^{-2i\omega_o t}$. Using the rotating wave approximation, this extremely high frequency exponential can be neglected. Obtaining as final Hamiltonians:

$$H(t) = \frac{\Omega}{2}e^{i\phi}|\alpha\rangle\langle\beta| + \frac{\Omega}{2}e^{-i\phi}|\beta\rangle\langle\alpha| + (\omega_\beta - \omega_\alpha - \omega_0)|\beta\rangle\langle\beta|\,. \tag{3.2.5}$$

In this final equation we have all of the necessary terms to describe atom-light interactions that enable transitions among ground and excited states of Rydberg atoms.
The Rabi frequency Ω is the characteristic frequency at which the atom is driven between states $|\alpha\rangle$ and $|\beta\rangle$. The value $\omega_\beta - \omega_\alpha - \omega_0 = -\Delta$, represents how off-resonant the laser is from the atomic energy transition, called detuning. the value ϕ, also called phase, is defined by the offset time of the laser drive and can always be set to zero, according to the $U(1)$ symmetry of the system [19].

3.2.4 Multi-qubit manipulation

A key enabling mechanism for multi-qubit manipulation in neutral-atom quantum processors is the use of highly excited Rydberg states, whose exaggerated atomic properties offer strong, tunable interactions between individual qubits. When a neutral atom is excited to a Rydberg state, its valence electron occupies a large spatial volume, giving rise to an enormous electric dipole moment. This leads to interatomic interactions governed by dipole-dipole couplings, which are fundamentally distance-dependent. At first order in perturbation theory, two Rydberg atoms in states such as $|nS, n'P\rangle$ and $|n'P, nS\rangle$ can exchange energy via resonant dipolar (flip-flop) interactions, scaling as $1/R^3$, where R is the interatomic separation. However, in most quantum computing architectures, it is the second-order van der Waals interaction that dominates. When both atoms are excited to the same Rydberg state, such as $|nS, nS\rangle$, they are coupled off-resonantly to intermediate states like $|n'P, n''P\rangle$ with opposite parity. The resulting second-order energy shift, which scales as $\propto |\langle nS, nS| V_{dd} |n'P, n''P\rangle|^2 / \Delta(n', n'')$, yields an effective van der Waals potential of the form $U = C_6(n)/R^6$, where the coefficient C_6 increases rapidly with the principal quantum number n.

This van der Waals interaction introduces an energy penalty when two qubits are simultaneously excited to the Rydberg state, forming the basis for entangling operations. Crucially, if only one or zero atoms are in the Rydberg state, no such energy shift occurs.

This lead to the Rydberg blockade effect (see Fig. 3.2.3): within a characteristic blockade radius, the energy of the doubly excited state becomes so large relative to the Rabi frequency Ω and detuning Δ that it can be effectively removed from the dynamics via adiabatic elimination. The result is a constrained Hilbert space in which doubly excited states are energetically suppressed, enabling robust, high-fidelity two-qubit gates. Importantly, this mechanism is insensitive to the precise value of the interaction strength—what matters is the structure of the interaction, not its fine-tuned value. As such, Rydberg-mediated interactions offer a powerful and scalable pathway to entanglement generation in neutral-atom QPUs.

3.2.5 Measurement process

Currently the main method to read out Rydberg qubits is via single-atom sensitive fluorescence imaging from the ground states. Rydberg excited atoms can be detected either by first transferring them to a suitable ground state, or by removing them from the trap prior to imaging in which case they show up as the absence of a signal. Rydberg state detection efficiencies > 0.95 are of the entire array, with the best results reported so far of > 0996. While this type of detection is usually destructive, high-fidelity lossless readout schemes for ground state qubits have also been demonstrated using state-selective fluorescence in free space, using cavity enhancement, or using state-dependent potentials. This would enable repeated measurements on qubits and to act on measurement outcomes, e.g., for quantum feedback and quantum error correction protocols [17].

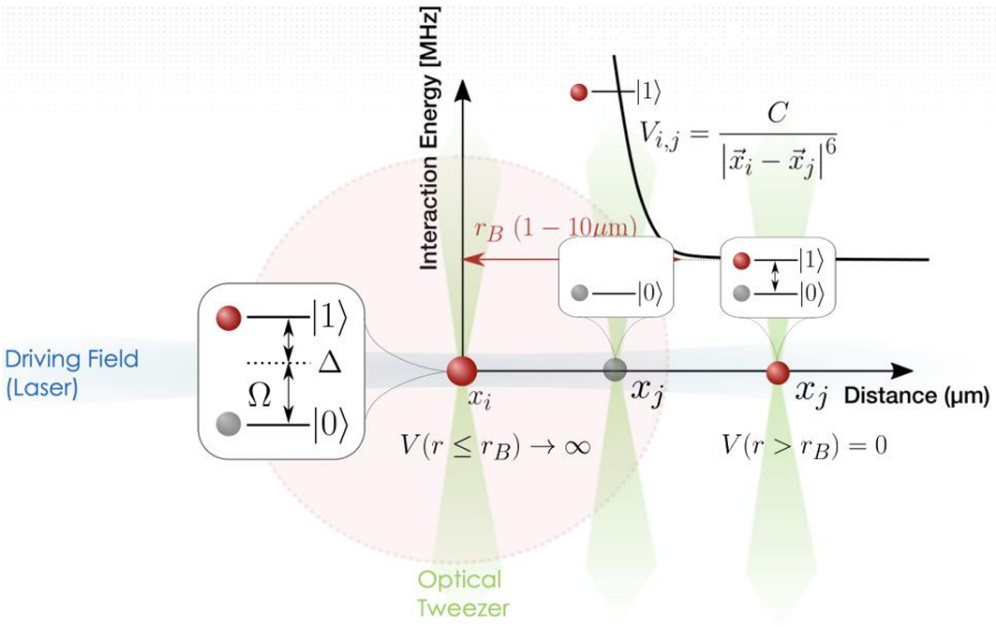

Figure 3.2.3: The Rydberg blockade mechanism. Two atoms are at some distance away from each other, where atom j is in the Rydberg state. Outside of the blockade radius (red), atom i can freely be driven to the Rydberg state. Inside the blockade radius, the Rydberg state is significantly detuned from the driving laser due to the strong interactions between nearby Rydberg-state atoms, preventing the atom j from going into the excited state. This behavior is independent of the specific position of the atoms, and so entanglement can be generated robustly not just through the specific values of the interactions, but in the structure of the Hilbert space [19].

3.3 QuEra Quantum Computer

QuEra Computing is a leading company in the development of quantum computers based on neutral atoms and Rydberg interactions. Building on advances in atomic physics and quantum optics, QuEra has developed Aquila, a cutting-edge quantum computing platform that harnesses individual atoms trapped in optical tweezers to perform coherent quantum operations. The system enables the realization of highly tunable quantum many-body dynamics by arranging up to 256 neutral-atom qubits in programmable geometries and controlling their interactions via laser-induced Rydberg excitations. Aquila operates as an analog Hamiltonian simulator, offering a powerful and flexible platform for investigating complex quantum phenomena and performing quantum simulations with direct relevance to condensed matter physics, optimization problems, and quantum information science.

3.3.1 Aquila's set-up

Aquila is a neutral-atom quantum computing platform that operates at room temperature, utilizing rubidium-87 (Rb-87) atoms cooled to the micro-Kelvin regime by means of laser cooling techniques inside a high-vacuum glass cell. Quantum information is encoded in the internal electronic states of individual atoms, which are manipulated through precisely timed and shaped laser pulses. State readout is achieved via state-dependent fluorescence, allowing for projective mea-

surements on the qubit states.
At the heart of the system is a compact 2-centimeter-scale vacuum cell, observed through a high-numerical-aperture microscope objective coupled to a low-noise camera. Around this core setup, a combination of standard optical components delivers and shapes the laser beams needed for atom cooling, trapping, and control, see Fig. 3.3.1. All subsystems are integrated into a data-center-style infrastructure, including racks of electronics responsible for laser stabilization, timing control, and data acquisition.
The atomic qubits are arranged in programmable two-dimensional geometries within a region smaller than 200 micrometers—comparable to the width of a few human hairs—where quantum operations are performed. To realize quantum computation, Aquila relies on four essential elements: (1) individual Rb-87 atoms as qubit carriers, (2) dynamically reconfigurable optical tweezer arrays for creating custom qubit layouts, (3) ultra-stable laser systems to coherently manipulate atomic states, and (4) Rydberg excitations, which mediate controllable interactions between qubits through strong, long-range dipole–dipole coupling.

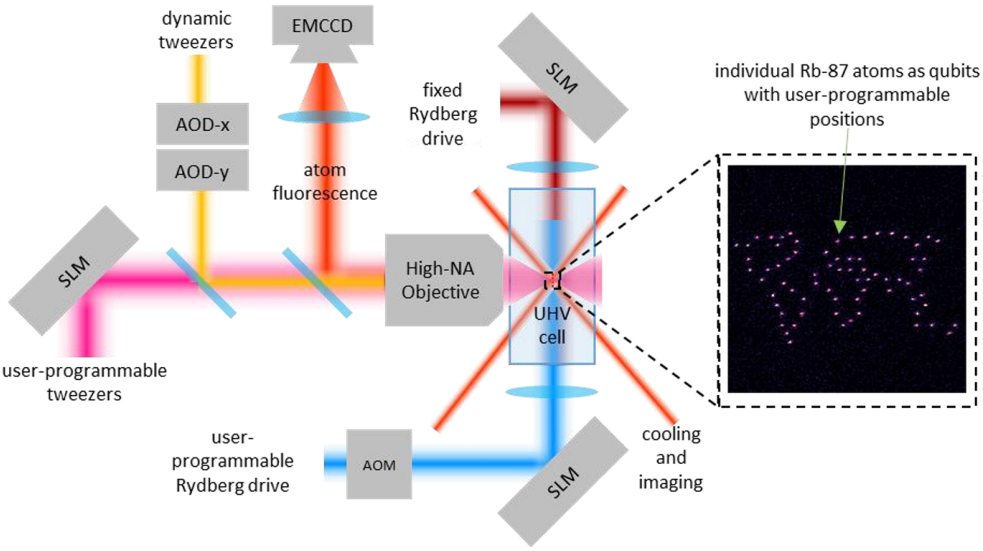

Figure 3.3.1: Functional block diagram of Aquila. There are six wavelengths of laser light that focus on the computational area in the vacuum cell. One laser (pink) is controlled by a spatial light modulator (SLM) to arbitrarily position up to 256 atom traps. Another laser (yellow) uses a crossed set of acousto-optic deflectors (AOD) to dynamically move atoms in traps and deterministically sort the array. A set of lasers (red) is used to cool the atoms to μK temperatures, and two counter-propagating lasers (deep red and blue) implement a two-photon drive between ground and Rydberg state. A final laser and camera (orange) is used to image the position of atoms in each trap using fluorescence. An example image of individual atoms leveraging arbitrary positioning is shown to the right [19].

3.3.2 Qubit Encoding and Readout in the Aquila Platform

In Aquila, each qubit is physically realized by a single rubidium-87 (Rb-87) atom. These atoms must undergo a precise sequence of operations: isolation, laser cooling to micro-Kelvin temperatures, spatial rearrangement into the desired geometry, qubit state initialization, coherent

manipulation, and finally measurement.
The qubit states are encoded in the electronic orbitals of the atom's valence electron. Specifically, quantum information is stored in two distinct energy levels: the logical state $|0\rangle$ corresponds to the electronic ground state $|g\rangle = |5S_{1/2}\rangle$ while the excited state $|1\rangle$ is mapped to a high principal quantum number Rydberg state, typically $|r\rangle = |70S_{1/2}\rangle$.
Intermediate electronic levels are employed during state preparation and manipulation processes. The relevant level structure is illustrated schematically in Fig. 3.3.2.

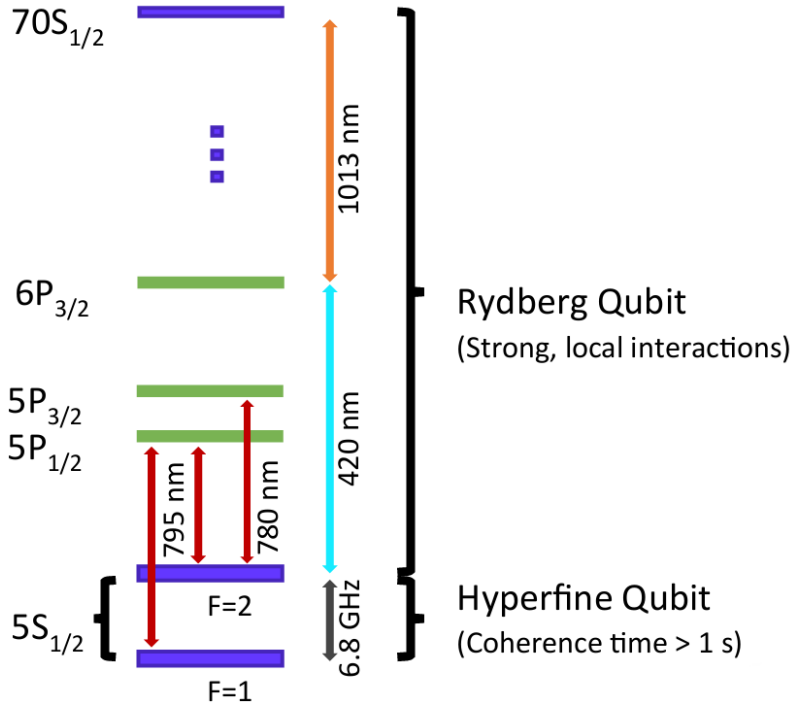

Figure 3.3.2: Rb-87 valence electron states utilized in Aquila to manipulate the atom as a qubit and as a physical host to the qubit. Arrows are the various optical fields used to drive transitions. Purple lines are the states that represent the qubit, while green lines represent other states used in the various manipulations [19].

The Rydberg state plays a key role in enabling qubit–qubit interactions, as it possesses exaggerated atomic properties—such as strong dipole moments—that are essential for generating entanglement. During the quantum evolution phase, optical trapping potentials are temporarily turned off to avoid perturbing the coherent dynamics. Once the evolution is complete, the traps are reactivated to perform a projective measurement.

This measurement process exploits the fact that the ground state is tightly trapped by the optical tweezers, whereas the Rydberg state experiences an anti-trapping potential that rapidly expels the atom from the trap region. As a result, the presence or absence of an atom in a given site—detected via fluorescence imaging—can be used to infer the qubit state: a detected atom indicates the qubit was in $|0\rangle$, while its absence corresponds to $|1\rangle$.
Measurements in this system are inherently destructive. Atoms found in the Rydberg state are lost during readout, and even ground-state atoms may occasionally escape or be misidentified due to imperfections in the trapping and detection process.

These errors can influence the measured fidelity of quantum operations, particularly in experiments probing many-body dynamics. Because the atom array must be rebuilt after each experimental shot, the repetition rate is limited (typically below 10 Hz). Nevertheless, this reconstruction process offers a unique advantage: the system's geometry can be reprogrammed between shots, effectively allowing the quantum processor to be reshaped dynamically depending on the needs of the computation or simulation being performed.

3.3.3 Optical Trapping and Atom Array Initialization

Aquila utilizes optical trapping to isolate and manipulate individual neutral atoms, enabling a flexible and reconfigurable architecture for quantum information processing. This technique relies on focused laser beams—optical tweezers—which use the optical dipole force to confine atoms in space. Specifically, a laser near-resonant with an intermediate atomic transition (in this case, a 780 nm laser tuned to the $6P_{3/2}$ level of Rubidium-87) induces a dipole moment in the atoms. The resulting radiation pressure pulls the atoms toward the high-intensity region of the beam, where they become trapped. Additional laser cooling then reduces the kinetic energy of the atoms, and optical pumping prepares them in their electronic ground state.
Aquila employs two main trapping modes: static and dynamic.

In the static trapping mode, a spatial light modulator (SLM) generates a two-dimensional array of tightly focused traps. The SLM, similar in concept to the technology used in digital projectors, modulates the phase of the incoming laser wavefront using a matrix of liquid crystals. Through holographic techniques, this phase modulation produces an interference pattern that forms hundreds of discrete focal spots—each serving as a trap for a single atom. Although the SLM's refresh rate is too slow for real-time quantum operations, it can be reprogrammed between runs to define arbitrary atomic configurations. Some examples of such flexible positioning are illustrated in Fig. 3.3.3, where fluorescence images reveal various trap geometries.

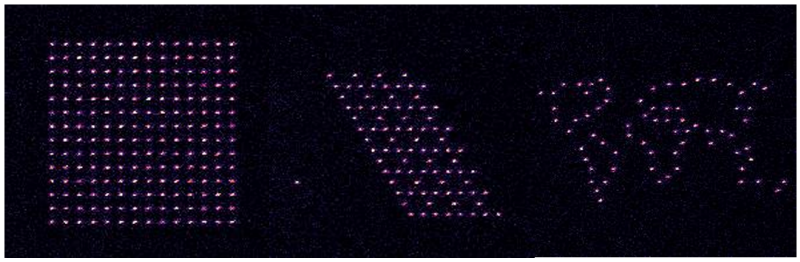

Figure 3.3.3: Examples of arbitrarily positioned atom arrangements enabled by reconfigurable tweezers. Left: regular array of qubits as a quantum register in a gate-based architecture, Middle: qubits arranged in a Kagome lattice to encode a quantum simulation problem, Right: qubits arranged in the shape of the world coastlines to encode a geographical optimization problem [19].

Despite the flexibility of this approach, physical limitations apply. Due to the resolution of the optics, no two trapping sites can be closer than 4 μm, and the total trapping area is confined to a 75 $\mu m \times 76$ μm region. These constraints must be taken into account when designing qubit arrays, although they can be partially relaxed under a "premium access" configuration.
The dynamic trapping mode makes use of acousto-optic deflectors (AODs) to actively move atoms

within the array. AODs function by launching acoustic waves through a crystal, creating a tunable diffraction grating that deflects laser beams according to the wave frequency. By combining two crossed AODs, the system can precisely control multiple tweezers simultaneously in two dimensions and move atoms on microsecond timescales.
The preparation sequence is illustrated in Fig. 3.3.4.

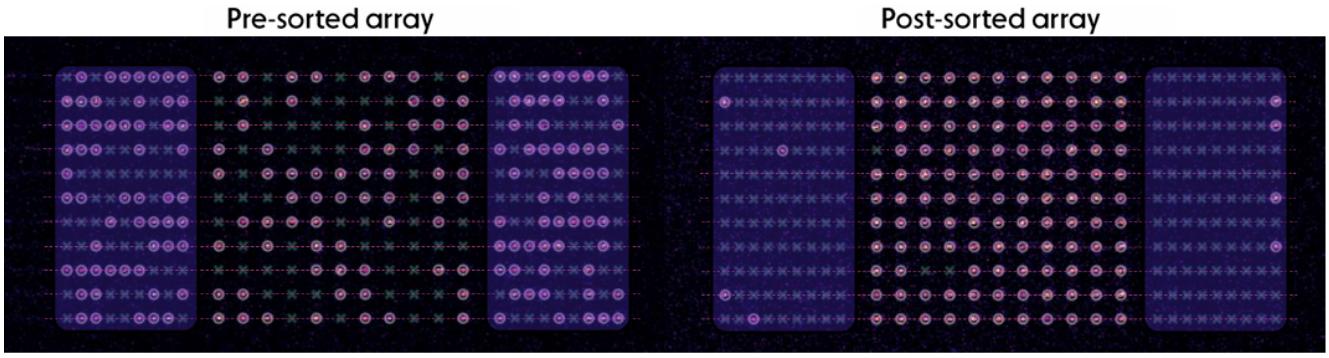

Figure 3.3.4: The deterministic loading process of an 11×11 square lattice. On the left is an image of the presorted, stochastically loaded array; empty sites are indicated with $\times$, and filled sites with $\circ$. Using a laser tweezer, atoms are moved from the reservoir regions (purple) on each side to the user region in the center to create a deterministically loaded array on the right image [19].

Initially, atoms are loaded probabilistically from a vapor into the SLM-generated traps, with each site being filled with 60% probability. To reach the maximum configuration of 256 atoms in a square array, about 600 traps are created, including auxiliary reservoir traps placed at the periphery. After fluorescence imaging identifies which traps are filled, the AODs orchestrate a real-time sequence of atom transfers, moving atoms from randomly occupied sites into the desired target configuration with a success probability exceeding 99%.While these constraints can be limiting for completely arbitrary qubit arrangements, they are typically compatible with structured configurations like square or Kagome lattices. Moreover, these constraints are architectural design choices rather than fundamental limitations and can potentially be lifted in enhanced-access modes.
Finally, it is important to emphasize that due to probabilistic loading and occasional measurement errors, experiments typically require post-selection on successfully filled arrays to ensure meaningful results.

3.3.4 Measurement cycle

All the individual stages of atom trapping, initialization, manipulation, and measurement in Aquila are organized into a single, coherent experimental sequence, executed via a series of carefully timed laser pulses. This process is illustrated in Fig. 3.3.5, which outlines the full measurement cycle.

A key characteristic of Aquila's measurement protocol is that it is destructive: atoms are detected by removing them from their optical traps, typically through fluorescence imaging or state-dependent loss. As a result, after each measurement shot, the entire atomic array must be reloaded and reinitialized from scratch.
While the quantum evolution itself typically lasts only on the order of $10\mu s$, the reload-and-prepare phase is significantly slower. Consequently, Aquila's current shot rate is limited to fewer

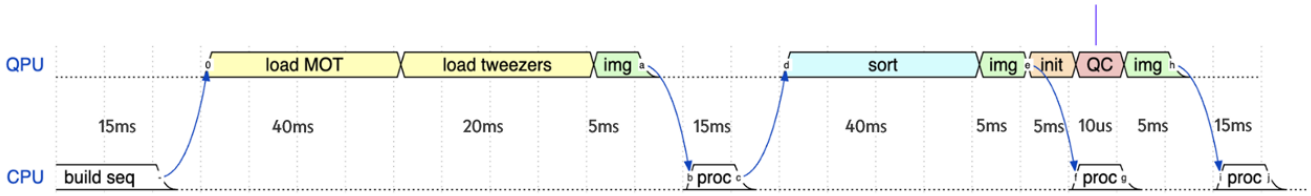

Figure 3.3.5: A full cycle of the Aquila processor. First, the magneto-optical trap (MOT) is loaded and then the static traps are loaded from the atoms in the MOT. Next, the occupancy of every randomly filled trap is imaged (img) and processed (proc), and the dynamic laser tweezers sort the array into the user-specified configuration. Another image is taken to determine the success of the sorting and is returned as the pre sequence data key. Then, the quantum computation (QC) is executed on a fast μs time scale. Finally, the traps are turned back on, pushing away the Rydberg state and trapping the ground state to perform a measurement. The atom occupancy is imaged and returned as the post sequence data key, which is interpreted as the bitstring measurement in the Z basis. Finally, the atoms are released back into the vacuum chamber and the cycle repeats, up to 10 times per second [19].

than 10 repetitions per second. This limitation affects data acquisition rates and can pose a bottleneck for applications requiring large statistical sampling or extensive circuit repetition.
However, ongoing development aims to address this constraint. Next-generation systems plan to incorporate non-destructive measurement techniques and improved atom retention mechanisms, which would allow the array to be reused across multiple measurement cycles. Such improvements could increase the shot rate by several orders of magnitude, greatly enhancing the system's efficiency and making it more suitable for near-term quantum applications.

3.4 Pasqal Quantum Computer

Pasqal is a leading company in the development of quantum computers based on neutral atoms and Rydberg interactions. Its platform offers a powerful architecture for quantum information processing by trapping individual atoms in configurable arrays of optical tweezers and exploiting their strong, controllable interactions via laser-induced Rydberg excitations.
In what follows, we present the main stages of computation on Pasqal's hardware, outlining how quantum registers are initialized, manipulated, and measured. Specifically, we discuss the process of register loading, where atoms are cooled and trapped in optical tweezers to form the physical qubits of the quantum processor. We then describe the quantum processing stage, during which quantum gates or analog Hamiltonian dynamics are applied by precisely tuning laser pulses and interatomic distances. Finally, we focus on the register read-out phase, where measurements are performed to extract classical information from the quantum system. This modular structure reflects the general workflow of a quantum computation on Pasqal's hardware, see Fig. 3.4.1.

3.4.1 Register loading

Pasqal's quantum processors are based on neutral rubidium atoms, a species widely used in atomic physics due to the availability of mature and reliable laser technologies. In this platform, two internal energy levels of each atom are designated as the computational basis states $|0\rangle$ and $|1\rangle$,

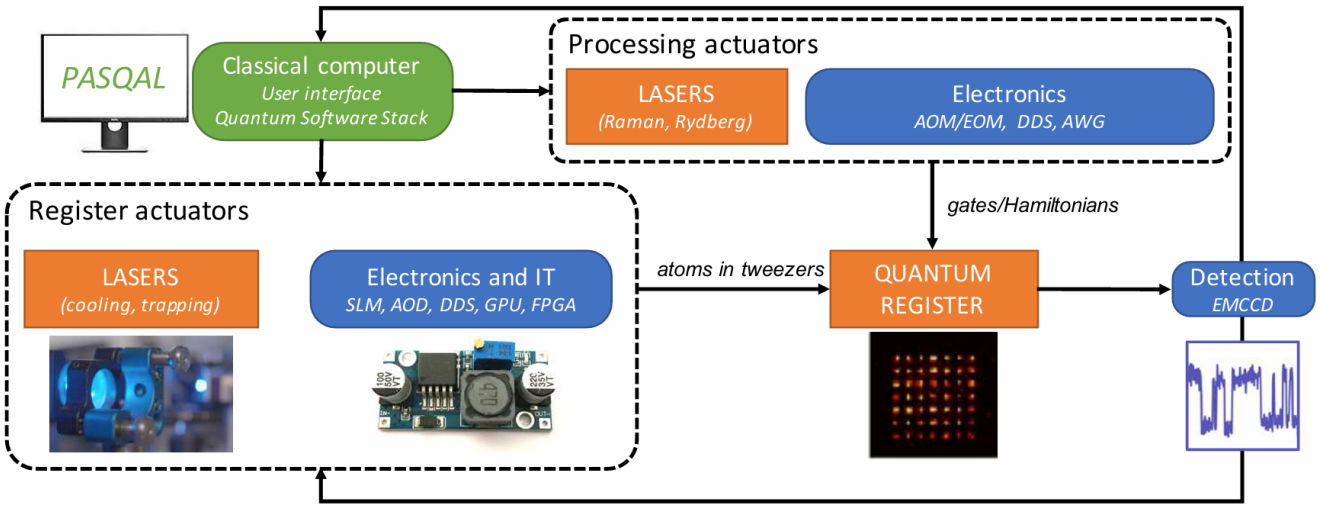

Figure 3.4.1: Schematic of the hardware components of a neutral atom quantum device. The user sends, through the quantum software stack, instructions to the register actuators, which initialize the quantum register, and to the processing actuators, which perform the computation. Information in the quantum register is extracted through detection of an image. It serves as an input for real-time rearranging of the register and as an output of the computation [20].

forming the qubit. The preparation of the quantum register is made possible thanks to the hardware components shown in Fig. 3.4.2(a).

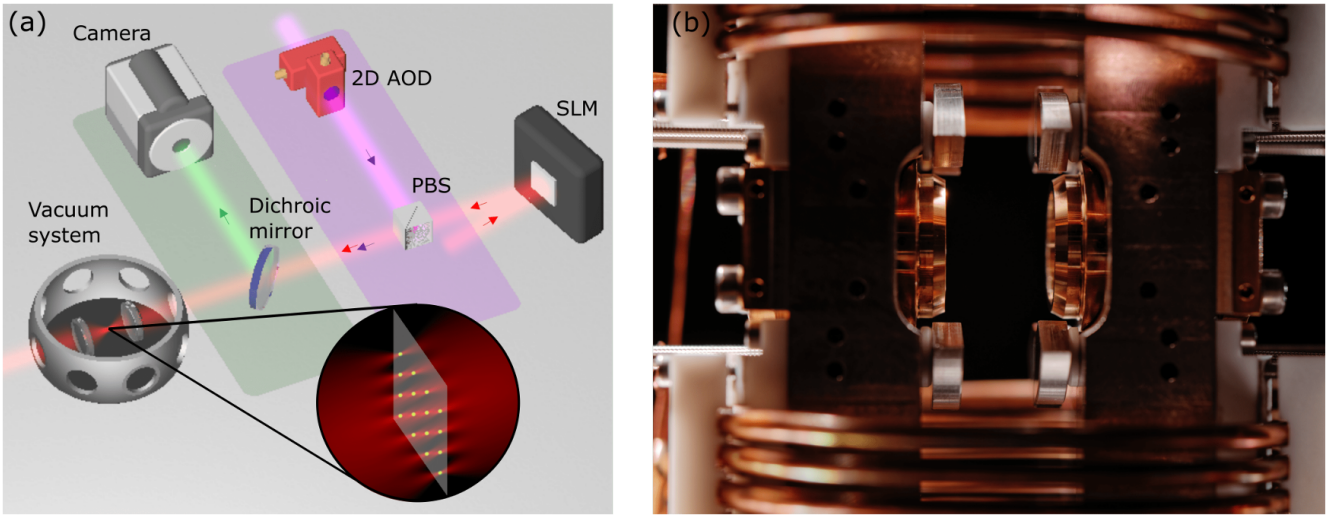

Figure 3.4.2: (a) Overview of the main hardware components constituting a quantum processor. The trap ping laser light (in red) is shaped by the spatial light modulator (SLM) to produce multiple micro-traps at the focal plane of the lens (see inset). The moving tweezers (in purple), dedicated to rearranging the atoms in the register, are controlled by a 2D acousto-optic laser beam deflector (AOD) and super imposed on the main trapping beam with a polarizing beam-splitter (PBS). The fluorescence light (in green) emitted by the atoms is split from the trapping laser light by a dichroic mirror and collected onto a camera. (b) Photography of the heart of a neutral-atom quantum co-processor. The register is prepared at the center of this setup [20].

As a starting point, a dilute atomic vapor is formed inside an ultra-high vacuum system operated at room temperature. With a first laser system (not shown), a cold ensemble of about 10^6 atoms and 1 mm^3 volume is prepared inside a 3D magneto-optical trap (3D MOT), leveraging numerous laser cooling and trapping techniques. Then, a second trapping laser system isolates individual atoms within this ensemble. Using high numerical aperture lenses, the trapping beam gets strongly focused down to multiple spots of about 1 μm diameter: the so-called optical tweezers. Since the spots are only 10 mm away from the lenses, the latter are placed inside the vacuum chamber (see Fig. 3.4.2(b)). Within a trapping volume of a few μm^3, each tweezer contains at most one single atom at a time. The configuration of the tweezers is fully programmable through a SLM, which shapes the phase profile of the trapping laser before it enters the optical system. This modulation produces a corresponding intensity pattern in the focal plane of the lens, enabling arbitrary tweezer arrangements in one, two and even three dimensions (see Fig. 3.2.2).

Since atom loading in tweezers is probabilistic, each trap being filled with roughly 50% success, an active rearrangement procedure is used to build a defect-free register. After the initial image is processed, a control algorithm computes in real time the necessary moves to assemble a sub-register with unit filling. These moves are executed via dynamically steerable tweezers, controlled by acousto-optic deflectors (AODs), with the overall process orchestrated by an FPGA-GPU pipeline that ensures low-latency feedback (see Fig. 3.4.3). Once rearrangement is complete, a second image is taken to confirm the successful preparation of the atomic register.

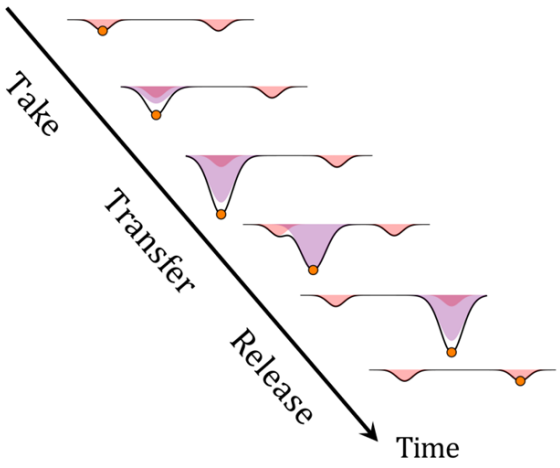

Figure 3.4.3: Moving a single atom from one site to another in the register. The moving optical tweezer first takes the atom, then transfers it and finally releases it into the other site. This operation takes less than 1 ms [20].

3.4.2 Quantum processing

Once the atomic register has been successfully assembled, the system is ready for quantum processing. The execution of quantum operations occurs on very short timescales, typically under 100

μs, whereas the full experimental sequence, including register loading and measurement, takes on the order of 200 ms.PASQAL's platform supports both digital and analog approaches to quantum computation.

In the analog regime, quantum evolution is driven by engineered Hamiltonians implemented through tailored laser configurations. The system evolves coherently under the Schrödinger equation, with interactions and dynamics dictated by the applied optical fields (see Fig. 3.4.4).

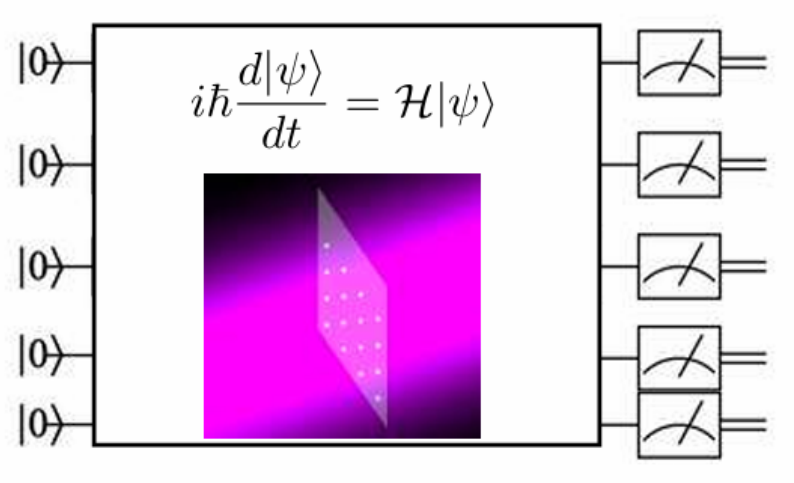

Figure 3.4.4: Qubits evolve under a tailored Hamiltonian H, for instance by illuminating the whole register with a laser beam. The wavefunction $|\psi\rangle$ of the system follows the Schrödinger equation [20].

At the end of the evolution, the quantum state of each atom is measured to extract the computational result. This versatility allows the hardware to simulate complex quantum dynamics and perform computational tasks by exploiting the native interactions between neutral atoms.
Rydberg atoms act as large electric dipoles, giving rise to dipole-dipole interactions that can be described using spin Hamiltonians.
Each qubit in the register effectively behaves like a spin, with the states $|g\rangle = |0\rangle$ and $|r\rangle = |1\rangle$ corresponding to the ground and Rydberg states of each atoms.
The Hamiltonian guiding the evolution in Pasqal's set-up is:

$$H(t) = \frac{\hbar}{2}\Omega(t)\sum_j \sigma_j^x - \hbar\delta(t)\sum_j n_j + \sum_{i\neq j}\frac{C_6}{r_{ij}^6}n_i n_j \tag{3.4.1}$$

with $n_j = (1+\sigma_j^z/2)$ the rydberg state occupancy. The first terms are induced by the laser that couples the qubit states and relate to an effective magnetic field, with transverse and longitudinal components $B_\perp \propto \Omega(t)$ and $B_\parallel \propto -\delta(t)$. Thus, one can modify them changing the intensity and frequency of the laser field. The third term in Eq. 3.4.1 relates the interactions between individual atoms, as previously discussed in Sec.3.2.4.
At this stage quantum processing can begin. The actual quantum operations are extremely fast, typically taking less than 100 μs, while the full experimental sequence, including atom loading and state readout, lasts approximately 200 ms.

3.4.3 Register read-out

The standard method for reading out Rydberg qubits relies on single-atom-resolved fluorescence imaging from the ground state. After processing, a final fluorescence image is acquired, in which atoms in the $|0\rangle$ state appear bright, while those in the $|1\rangle$ state remain dark.
This state-dependent fluorescence is typically captured using an electron-multiplying charge-coupled-device (EMCCD) camera, which converts the emitted photons into an amplified electronic signal with very high sensitivity. Detection efficiencies exceeding 98.6% have been demonstrated using this technique, which is on par with readout fidelities achieved in superconducting qubit platforms.

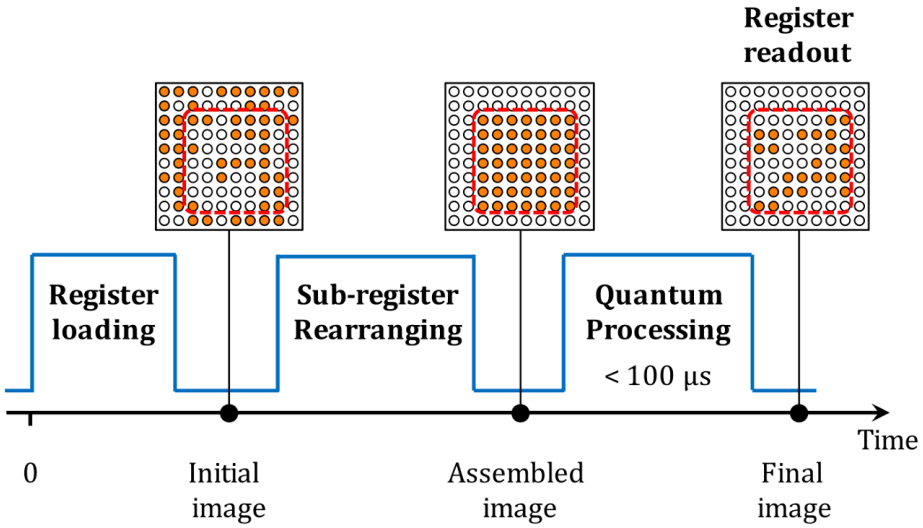

Figure 3.4.5: Temporal sequence of one computation cycle. The loading of the register being random, atoms are first rearranged to realize a defect-free sub-register, on which the quantum processing is performed [20].

4 Quantum Reservoir Computing

This chapter is focused on the study of Quantum Reservoir Computing (QRC), a hybrid quantum-classical machine learning model designed to process data by leveraging the complex dynamics of a quantum system driven by classical inputs.

We begin by briefly reviewing the essential ideas behind classical Reservoir Computing, focusing on the key intuition that underpins both classical and quantum versions.

We then turn to the quantum extension of this framework, examining how the intrinsic properties of quantum systems can be harnessed to enhance the expressive power and computational efficiency of reservoir models. In particular, we highlight the advantages of QRC over other paradigms, discuss its practical implementation, and outline the classes of tasks that can be effectively addressed using this approach.

Finally, we present in detail the algorithmic structure of QRC, focusing on the encoding of classical data, the evolution of the quantum reservoir, the extraction of observables, and the classical post-processing steps.

4.1 Reservoir Computing

Reservoir Computing (RC) is a machine learning paradigm that leverages an input-driven dynamical system. The core idea of RC is that a significant portion of the computational task is not performed by a trained network, but by a high-dimensional system called the reservoir, which is treated as a black box. The outputs of the reservoir are fed into a single readout layer, which is the only component of the network that is trained, as shown in Fig. 4.1.1.

The foundational papers on which reservoir computing relies were independently introduced in the early 2000s, in [22] and [23]. In these seminal works, although the term reservoir computing was not explicitly used, the authors introduced the fundamental concept of exploiting complex dynamical systems to process and enrich input features.

In the Echo State Network (ESN) approach, the reservoir consists of a recurrent neural network (RNN) with randomly assigned weights that satisfy certain algebraic conditions (the echo state property). The untrained part of the RNN acts as a reservoir, with its states representing a nonlinear transformation of the recent input history. The trained component is a linear output layer [24]. On the other hand, Liquid State Machines (LSMs) were inspired by neuroscience studies, with the term liquid referring to how input spikes induce ripples in neural activity, capturing

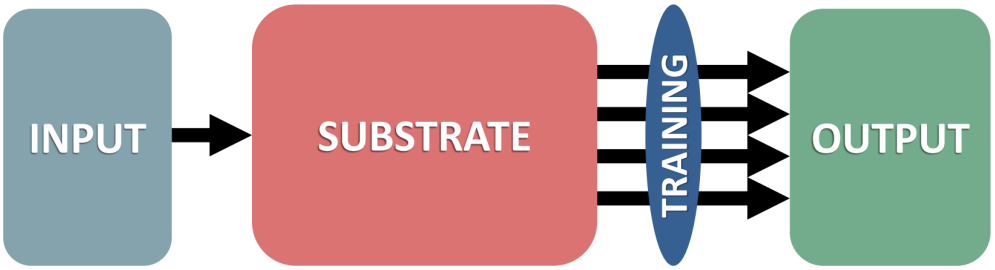

Figure 4.1.1: Schematic representation of the basic components of RC. The information from the input is fed into the substrate, which acts as a hidden layer or reservoir. The response of the substrate, through a selection of observables, is then used to produce the desired output after optimization of the output connections by training [21].

temporal information in a distributed way. In this case, the reservoir is derived from a biological context, but the underlying principles are the same.
Later on, these two approaches were unified under the common term reservoir computing, coined in [25].
In order to implement this technique, a training input signal $\mathbf{u}_k \in \mathbb{R}^{N_u}$ is required, which is supposed to lead to the desired N_y-dimensional output $\mathbf{y}_k$. The most general form of this mapping is:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{u}_k, \mathbf{x}_{k-1}), \tag{4.1.1}$$

where $\mathbf{f}$ is a fixed function determined by the system and the encoding technique, and $\mathbf{x}_k$ is the state of the reservoir at the k^{th} step. The final step is to extract information from the reservoir using a readout function $\mathbf{F}$:

$$\mathbf{F}(\mathbf{x}_k) = \tilde{\mathbf{y}}_k. \tag{4.1.2}$$

The readout function contains free parameters that are optimized during the training process, and is the only part of the algorithm affected by training. It is designed to minimize a loss function that typically depends on the difference between the actual output $\tilde{\mathbf{y}}_k$ and the desired output $\mathbf{y}_k$. In most cases, $\mathbf{F}$ is obtained via linear regression. Once $\mathbf{F}$ is known, the system can be applied to new inputs.
This strategy offers several advantages over conventional deep neural networks, which are usually expensive to train:

- Training usually consists of a simple linear regression, which is computationally inexpensive and easy to implement.
- Only the readout function $\mathbf{F}$ needs to be trained, not the entire system. This is especially advantageous when dealing with a physical system where the precise interactions between components are hard to modify or even fully understand.
- The same reservoir can be reused for different computing tasks by training different readout functions $\mathbf{F}$.

4.1.1 Reservoir dynamics

Beyond serving as a computational black box, the reservoir plays a critical role in shaping the performance of Reservoir Computing by embedding the input into a dynamically rich state space. Its internal structure, typically composed of recurrent and nonlinear interactions, induces a form of temporal processing, where the state of the system reflects not only the present input but also a fading trace of its past history.
This implicit memory is crucial for time-dependent tasks: the reservoir state $\mathbf{x}_k$ at a given step does not simply mirror $\mathbf{u}_k$, but emerges from the combined influence of the current input and the dynamical evolution from prior inputs. However, unlike in standard recurrent neural networks, this temporal memory is not explicitly engineered via training, but rather emerges from the inherent dynamics of the fixed system.
An essential requirement for a functional reservoir is the so-called echo state property: over time, the effect of initial conditions must vanish, and the system should asymptotically reflect only the influence of the input stream. If this condition is not met e.g., if internal dynamics dominate too strongly or chaos ensues, the reservoir may become insensitive to external inputs, thus undermining its ability to perform useful computations.

Another key feature of the reservoir is its ability to perform nonlinear expansion of the input signal. Through the system's intrinsic complexity, the input trajectory is projected into a high-dimensional space where subtle differences or patterns become more distinguishable. This is particularly beneficial in tasks where the original input features are not linearly separable. The readout layer, though linear, can then recover complex input-output relationships thanks to this enriched representation.
The interplay between memory and nonlinearity is what enables a fixed reservoir to generalize across a wide range of computational tasks. This combination of fading memory, high-dimensional representation, and dynamic sensitivity is what makes Reservoir Computing a powerful framework for real-time processing, particularly in settings where the underlying substrate, be it digital or physical, cannot be modified or trained internally.

4.2 Quantum Extension: Motivation and Advantages

Current NISQ devices faces significant limitations due to noise and decoherence. The predominant class of QML algorithms are VQA leveraging on a classical computer for the optimization of parameters required in the quantum process. As anticipated in Sec. 2.4.1 these approaches are fundamentally constrained from barren plateaus and complex training landscape, preventing trainability of these models.
One of the main obstacles to trainability in VQA arise from the Barren Plateau (BP) phenomenon, occurring when the loss function, or its gradients, become exponentially concentrated around their mean as the number of qubits increases. This means that the optimization landscape is mostly flat and featureless. Therefore, slightly changing the model's parameters θ results in only an exponentially small change in $\mathcal{L}_\theta$, the loss function.
Considering that information can only be extracted from a quantum computer through a finite number of measurements, the presence of Barren Plateaus in the optimization landscape means that, for most choices of parameters, an exponentially large number of measurement shots is required to determine the direction that minimizes the loss. Ultimately, the exponentially large

Hilbert space, initially expected to give variational quantum computing an advantage over classical methods, leads instead to fundamental challenges in the practical application of variational algorithms. [26]. These challenges make VQAs hard to be implemented in current NISQ devices, motivating the exploration of alternative frameworks that avoid costly internal optimization.

The RC framework, as discussed in Sec. 4.1, provides an alternative machine learning paradigm that bypasses the need for costly gradient-based optimization. By mitigating several of the limitations currently affecting quantum machine learning methods, QRC has recently emerged as a promising strategy for near-term quantum devices.

The structure of reservoir computing, see Fig. 4.1.1, allows huge flexibility in modifying each part of the algorithm and thus, enabling different tasks and physical realization of the algorithm. Specifically, as first pointed out by [21], with the quantum reservoir computing algorithm several possibilities are opened: classical or quantum character of the input, the substrate used and the task to assess.

	Classical Substrate $\mathbf{x_k}$	**Quantum Substrate $\mathbf{H}(t)$**
Classical Input s_k	**CCC** $\{s_k\}, x_k^{out} \to T_C$	**CQC** $\{s_k\}, \{O_i^{out}\} \to T_C$
	CCQ $\{s_k\}, x_k^{out} \to T_Q$	**CQQ** $\{s_k\}, \{O_i^{out}\} \to T_Q$
Quantum Input $\|\psi_k\rangle$	**QCC** $\|\psi_k\rangle, x_k^{out} \to T_C$	**QQC** $\|\psi_k\rangle, \{O_i^{out}\} \to T_C$
	QCQ $\|\psi_k\rangle, x_k^{out} \to T_Q$	**QQQ** $\|\psi_k\rangle, \{O_i^{out}\} \to T_Q$

Table 4.1: All possible combinations of input, substrate and task being classical (C) or quantum (Q) [21].

In table 4.1, all the opportunities in QRC are summarized. The sub-index k labels each time step or instance. The sub-index i is associated to the internal degrees of freedom of the substrate. For the classical input, $\{s_k\}$ is a data sequence, e.g. a string of real numbers. In the quantum case, $|\psi_k\rangle$ represents an input state. The state of the substrate at a given instant is defined by x_k in the classical regime and by the evolved state $|\psi_k(t)\rangle = e^{-i/\hbar H(t)} |\psi_k\rangle$ in the quantum one. For the training process with a classical substrate, a selection of the substrate variables are used, x_k^{out}. With a quantum substrate, the readout for the training is obtained after a set of measurements, $\{O_i^{\text{out}}\}$. We distinguish between classical tasks, T_C, and quantum tasks, T_Q.
The configurations that are currently feasible are CCC, CQC, and, to some extent, QQC. CCC represents the classical baseline, while CQC—the most investigated case in QRC, it combines classical inputs with quantum evolution and a classical task, and can already be implemented on NISQ devices. QQC is possible only at small scale, being strongly limited by qubit number and measurement overhead.
All the other configurations remain mostly conceptual, as they require fully quantum inputs, tasks, or readout stages that are either not accessible with today's hardware or would scale exponentially

in terms of qubits and measurements.
In this work, we focus on the CQC scheme. This configuration is the most natural extension of the classical RC paradigm and can be summarized by the following sequence:

$$\{s_k\} \to \{O_i^{\text{out}}\} \to T_C \tag{4.2.1}$$

Classical inputs, $\{s_k\}$ are encoded into the quantum reservoir. After the quantum evolution, a collection of observables $\{O_i^{\text{out}}\}$ is measured and used as features for a simple classical readout model (e.g., linear regression or a support vector machine) to solve a classical learning task T_C.
In this set-up, a complex quantum system replaces the classical reservoir, exploiting the dynamics of its exponentially large Hilbert space to transform input data and generate rich, classically intractable correlations. The key motivation for using quantum substrates lies in their vast number of degrees of freedom: a reservoir of N qubits evolves in a 2^N-dimensional space, enabling an exponential nonlinear embedding of the input data. Thanks to this enriched data representation, even a simple final layer such as a linear support vector machine is sufficient to achieve optimal performance.
This new set-up, introduced for solving both classical and quantum tasks, makes real advantage of the exponentially large Hilbert space, avoiding by construction of the algorithm the BP phenomenon.

4.2.1 Hardware implementations

Recently, several proposal for QRC hardware implementation have been successfully tested, showing the viability of the approach on current devices. In Fig. 4.2.1, an overview of the most successful hardware implementations reported to date.

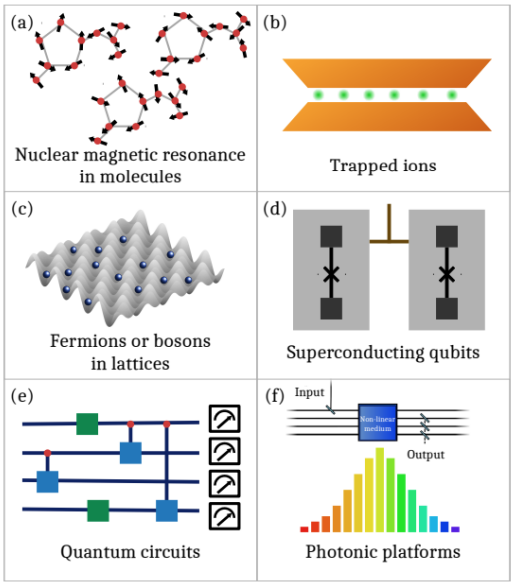

Figure 4.2.1: Various examples of platforms to be used as quantum substrates for information processing. Two suitable candidates to implement spin-network models are (a) nuclear magnetic resonance in molecules, and (b) trapped ions. (c) State-of-the art ultra-cold atomic setups in optical lattices with bosonic and fermionic species are well-suited for Bose-Hubbard and Fermi-Hubbard discrete models, respectively. Additionally, other physical implementations are possible, e.g. in arrays of quantum dots in semiconductor devices, and in (d) coupled superconducting qubits. (e) Quantum circuits have been used as a substrate in the IBM platform. (f) Continuous-variable models could be engineered in photonic experimental setups as well, i.e. by coupling different frequency modes in non-linear media [21].

4.2.2 Use cases

The hardware implementations presented The hardware implementations presented above demonstrate that quantum reservoir computing is not merely a theoretical construct, but can already be implemented on several platforms in the NISQ era, each with its own advantages and challenges. To further highlight the reliability of the QRC approach, in the following, many examples are shown where QRC is applicable today. These tasks include quantum state tomography, chemistry and high-energy experiments. In these tasks QRC is applicable bypassing the critical bottleneck of most VQA models.
Conventional quantum tomography schemes rely on the full state reconstruction through exponentially many projective measurements, in Ref. [27] the authors are able to reconstruct an arbitrary quantum state by measuring only the average occupation numbers in a single physical setup. QRC has also been applied to quantum chemistry tasks.

In Ref. [28] the authors showed a theoretical scheme for predicting excited states of a molecule by giving only their ground state wavefunction as input to the reservoir, significantly reducing the computational cost. This algorithm was subsequently implemented in Ref. [29] [Domingo2023] showing an improvement of performance in certain noisy conditions. Furthermore, even drug discovery tasks can be enhanced by hybrid architectures including a QRC step. A notable application of QRC in life science is reported in Ref. [30] where a hybrid quantum-classical neural network is proposed for protein-ligand binding affinity prediction. In this architecture, classical convolutional networks are fused with a quantum reservoir acting as a non-trainable feature transformer. The quantum reservoir is implemented using a quantum circuit composed of two blocks: a data encoding layer, maps molecular descriptors to quantum states, and a reservoir evolution layer, where random unitaries generate non-linear transformation of the input layer. This hybrid model achieved a 6% accuracy increase, up to 40% shorter training times and 20% reduction in parameters. This work validates the applicability of QRC to high-dimensional learning tasks, but also demonstrates possible integration schemes in modern AI pipelines for real-world use cases.
In the digital approach to QRC, after the data encoding, random unitary operators are sampled from carefully selected families in order to explore as much as possible the Hilbert space, important results [31] identified families of random unitaries that optimize the QRC performances. Remarkably has been shown that QRC can benefit from noise under specific conditions. In particular, in the paper [29] they showed that amplitude damping noise can improve QRC performance on a quantum chemistry task, outperforming the noiseless case for sufficiently shallow circuits. Due to the quantum nature of the data being analyzed, together with the high dimensionality and temporal structure of collider experiments such as those at the LHC, quantum machine learning techniques are naturally suited to address tasks such as rare signal detection, event classification, and model-independent anomaly detection. In Ref. [32], the authors identify QML techniques as promising tools to address these challenges, particularly for low-latency, online data processing within the constraints of NISQ hardware. These requirements are naturally aligned with the QRC paradigm, which combines training-free quantum evolution with the ability to extract complex temporal correlations via shallow circuits or analog dynamics. QRC thus emerges as a well-matched framework for real-time inference in next-generation high-energy physics experiments.
The advantages over the most popular choice for QML, hardware implementations and use cases discussed in this section showed quantum reservoir computing as a solid alternative suited for NISQ devices. Its training-free architecture, combined with rich nonlinear dynamics and robustness to realistic noise, enables it to tackle a wide variety of tasks, from quantum state tomography and molecular simulation to real-world data analysis in high-energy physics and drug discovery. This broad versatility and concrete realizability provide strong motivation to further explore QRC models, both in algorithmic design and in their deployment on near-term quantum platforms.

4.3 Quantum Reservoir Computing Algorithm

Having discussed the principles of classical reservoir computing and the motivations behind its quantum extension, we now turn to the specific quantum algorithm that forms the foundation of this thesis work.
The approach is based on the recent proposal in Ref. [33], which presents a scalable and training-free QRC scheme implemented on neutral-atom analog quantum hardware. This algorithm is designed to operate without variational parameter optimization, leveraging the natural dynamics of a highly controllable quantum system to perform nonlinear transformations on classical input data. It stands out for being gradient-free, scalable, and well-suited to current NISQ devices.
The QRC framework consists of three main stages: classical pre-processing of the input data, quantum evolution under a time-dependent Hamiltonian, and classical postprocessing for supervised learning tasks, see Fig. 4.3.1

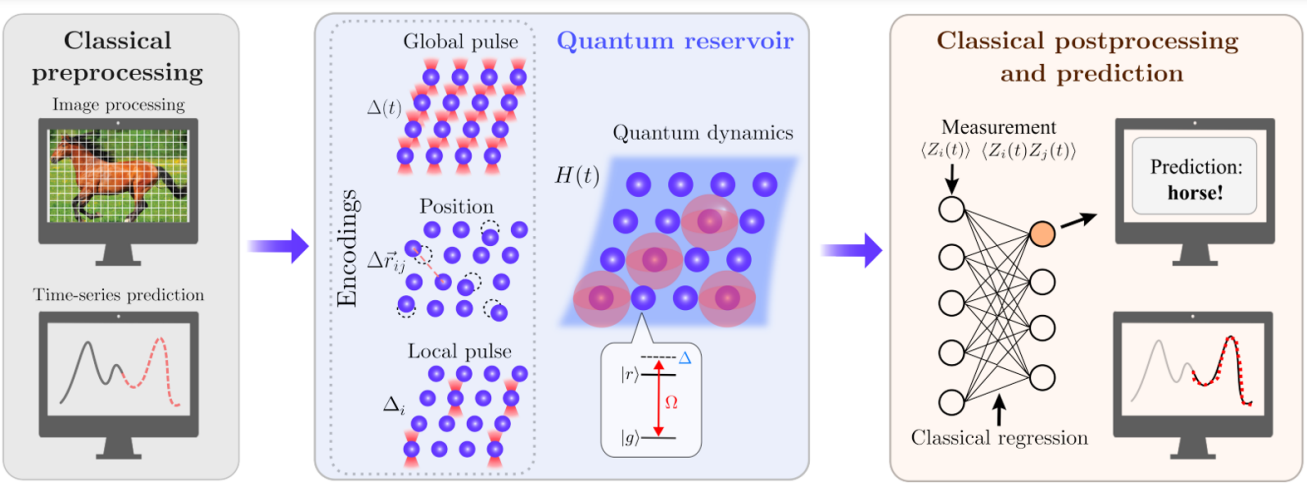

Figure 4.3.1: Overview of the quantum reservoir computing algorithm with neutral atoms [33].

In the pre-processing step, the training and testing datasets are organized as pairs $\{(x_i[n], y_i[n])\}$, where $x_i[n]$ denotes the input feature vector and $y_i[n]$ the corresponding labels. Here n is denoting the data samples, while i and j refer, respectively, to the components of the feature and label vectors. Depending on the nature of the input data, this step may involve optional dimensionality reduction (e.g., for high-dimensional data such as images) or feature engineering techniques.

Once appropriately preprocessed, the data is mapped onto the quantum reservoir through one of three encoding strategies: encoding into the time profile of the global detuning pulse, modulation of interaction strengths via atom position arrangement, or application of site-dependent local detuning pulses. Once the data feature has been encoded in the reservoir, the quantum system evolves under the specified Hamiltonian dynamics, which enrich the input data through non-linear transformations in the Hilbert space.
After a sufficient evolution time, the system is probed through projective measurements, from which expectation values of local observables are collected. These embeddings are then passed to a fast classical training step, typically using linear support vector machines or regression. Once trained, the model performs inference by applying the QRC pipeline to unseen inputs and pro-

ducing predictions based on the resulting embeddings.

In the next sections we will deep dive in the methods employed in the article, focusing on the different encoding strategies, the physical implementation of the quantum reservoir, the extraction of features via projective measurements and finally the results obtained with the analog quantum computer, Aquila.

4.3.1 Encoding techniques

The first step in the QRC pipeline is to find a proper encoding technique for the classical data under consideration. Choosing the appropriate encoding strategy is a pivotal step in order to obtain reliable results on a quantum computer. As shown in Fig. 4.3.1 in the QRC algorithm three different encoding techniques are possible, each suited for different tasks. The availability of these encoding strategies arises from the structure of the neutral-atom QRC, specifically we aim to encode features into the parameters of the Rydberg Hamiltonian, the more flexible this Hamiltonian is, the more encoding strategies become available.
The Rydberg Hamiltonian is:

$$H(t) = \frac{\Omega(t)}{2} \sum_j (|g_j\rangle \langle r_j| + |r_j\rangle \langle g_j|) - \sum_j [\Delta_g(t) + \alpha_j \Delta_l(t)] n_j + \sum_{j<k} \frac{C_6}{|\mathbf{x}_j - \mathbf{x}_k|^6} n_j n_k \qquad (4.3.1)$$

where $\Omega(t)$ is the global Rabi drive amplitude between a ground $|g_j\rangle$ and a highly-excited Rydberg state of an atom $|r_j\rangle$. The detuning term is split into a global $\Delta_g(t)$, and a site-dependent one $\Delta_l(t)$, with site modulation $\alpha_j \in [0, 1]$. Therefore, these three tunable parameters allow the exploration of three encoding scheme:

1. **Global pulse encoding**: implemented by mapping data features in the time varying profile of the global detuning pulse, $\Delta_g = \Delta_g^{max} x_i$. Here the features are encoded as pulse parameters at different times. This is particularly convenient when one want to encode some time-dependent feature, enabling a natural way to encode those data in the time varying shape of the detuning pulse. Further the encoding capacity does not depend on the system size (N_q), depends only on the pulse control.

2. **Position encoding**: this encoding strategy is well suited when we want to preserve the spatial information in the input data. The encoding is implemented by modifying the interaction strength between neighboring atoms such that $V_{ij} = V^{(0)}(1 + \lambda x_i)$, where $V^{(0)}$ is the unperturbed interaction value, λ is a scaling parameter controlling the encoding strength, and j identifies a nearest neighbor of the i^{th} site. In a one-dimensional system composed of N_q qubits, this scheme allows the encoding of up to $N_q - 1$ distinct features through the tunable nearest-neighbor couplings.

3. **Local pulse encoding**: this encoding strategy enable to address single qubits via site dependent local pulse. In this local encoding strategy the detuning is a local quantity and is modified for each atoms in a different way according to $\alpha_i \Delta_l(t) = \Delta_l^{amx} x_i$, therefore, enable to encode N_q features in a reservoir composed by N_q atoms.

These three encoding strategies differ not only in how the classical data is embedded into the quantum system, but also in their practical effectiveness. The comparative performance of the

different encoding strategies is discussed here in detail, as it provides critical insights into the design of the QRC pipeline. While this section focuses specifically on the role of encoding, a broader discussion of experimental results across all learning tasks is presented in Section 4.3.4.

Based on the specific control parameters involved, the encoding capacity, and whether the strategy acts globally or locally, one can already anticipate that certain encoding methods may be better suited for specific tasks than others. Moreover, guided by both physical intuition and numerical simulations, it is possible to establish a hierarchy among the encoding strategies in terms of their overall effectiveness.
According to physical arguments, we can already anticipate that global encoding techniques, despite being conceptually well-suited for time series tasks, tend to perform less effectively than other strategies. This can be understood by considering that quantum dynamics generally leads to thermalization of local observables. As a result, if measurements are taken too late in the evolution, there is a significant risk that the system has already reached a thermalized state, and the local observables no longer retain meaningful information about the original input data. Furthermore, due to the inherently noisy nature of current quantum devices, decoherence further degrades the quality of the late-time measurements, enhancing the loss of information in global pulse encoding schemes.
The limitation of global pulse encoding can, in principle, be mitigated by adopting protocols that continuously extract information from the system during its evolution or to concentrate the dynamics in a narrower evolution window. One possible approach is the use of mid-circuit measurement techniques, which would allow the reservoir to emit meaningful embeddings at various stages of the dynamics, thereby reducing the impact of thermalization and decoherence on the encoded information. This considerations are well supported from experimental results performed in the Aquila neutral atom quantum computer. They apply the QRC algorithm to process the Santa Fe timeseries task (further detail in Sec. 4.3.4) using the three different encoding techniques on this task they compare the normalized mean-square error (NMSE) in a finite-sampled simulation and experiment.

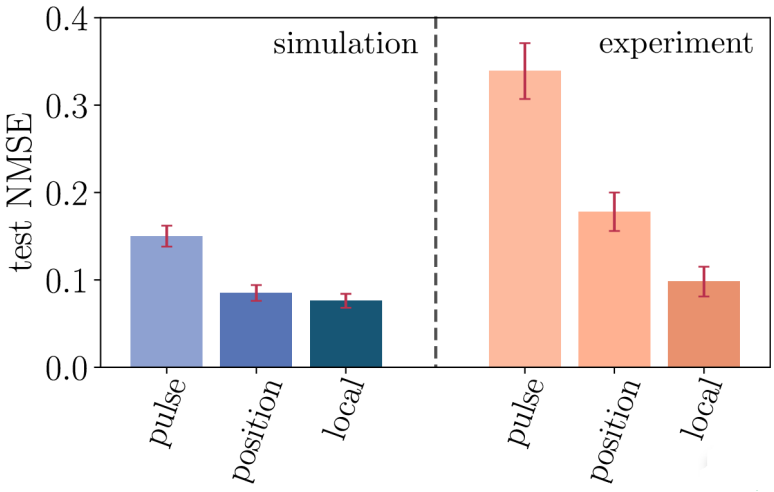

Figure 4.3.2: Normalized mean-square error for different encodings [33].

As shown in Fig. 4.3.2, the experimental results support the previously discussed limitations of

the global pulse encoding. When applied to the Santa Fe laser timeseries task, the normalized mean-square error (NMSE) clearly indicates that global pulse encoding performs worse than both position and local pulse encodings. This behavior is consistent across both finite-sampled simulations and experimental runs on the Aquila quantum processor.
In contrast, local pulse and position encodings show approximately equivalent performance in simulations, suggesting that both strategies effectively preserve the input information throughout the evolution. This equivalence can be physically understood by noting that the position encoding induces an effective site-dependent detuning. This becomes clear when expressing the Rydberg interaction term in the Hamiltonian in Eq. 4.3.1 through Z operators, using $Z_i = 2n_i - I_i$ as follows:

$$\sum_{i<j} V_{ij} n_i n_j + \sum_i \Delta_i n_i \to \frac{1}{4} \sum_{i<j} V_{ij} Z_i Z_j - \frac{1}{2} \sum_i \left(\Delta_i + \frac{1}{2} \sum_{i \neq j} V_{ij} \right) + \text{const.} \tag{4.3.2}$$

Thus, the position encoding modulation effectively induces local detuning modulation, with the local detuning modulation on one site being dominated by the features encoded at its nearest neighbors. Since both methods apply the data-dependent modulation at the start of the evolution, their embeddings are less vulnerable to information degradation from thermalization and decoherence, as discussed earlier. Overall, the results in Fig. 4.3.2 not only confirm the hierarchy among the encoding strategies, but also highlight the practical trade-offs introduced by hardware imperfections. While position and local pulse encodings offer similar robustness and accuracy, their sensitivity to different types of experimental noise must be considered when selecting an encoding scheme for specific tasks and hardware configurations.

4.3.2 Evolution

Once the data features have been encoded into the quantum reservoir, the next step is to let the system evolve unitarily under the action of the Hamiltonian, following the transformation $e^{-\frac{i}{\hbar}Ht} |0\rangle$. This is the most crucial step of the algorithm, as it is expected to induce a non-linear and non-classically replicable transformation of the input data. Such transformation arises from the quantum nature of the evolution, where superposition and entanglement play a central role in enriching the structure of the quantum state. Furthermore, one of the main advantages of this algorithm lies in the absence of variational parameters to optimize. Therefore, it is essential to demonstrate that the dynamics governed by the Rydberg Hamiltonian alone is sufficient to provide the necessary expressivity for the learning task.
As discussed in the previous section, data can be encoded into the quantum reservoir by appropriately setting the parameters of the Hamiltonian. Therefore, in order to achieve a rich and expressive quantum dynamics, it is crucial to properly configure the remaining Hamiltonian parameters that are not determined by the input data. To do so, one need to understand the relevant energy scale of the system:

- mixing scale, determined by the effective maximum Rabi amplitude, $\bar{\Omega}$;
- entanglement scale, determined by the average Rydberg interaction strength in the array, $\bar{V}$;
- encoding scale, which, depending on the encoding method, can include average local detuning (local pulse encoding), fluctuating part of the interaction strength (position encoding), or average global detuning encoding carrying amplitude (pulse encoding), $\bar{\lambda}$;

- probing frequency scale, given by the inverse of the quantum dynamics probing time, $(\bar{\Delta t})^{-1}$
- encoding frequency scale, specific to global pulse encoding, defined as the inverse of the data-point encoding interval, $(\bar{\Delta\tau})^{-1}$.

Based on physical reasoning, one can identify a region in the relevant QRC parameter space where the system exhibits optimal performance. Once this regime is determined, there is no need to perform variational optimization over these parameters, as the dynamics is already sufficiently expressive. This region is referred to as the *universal parameter regime*, and can be discovered through physically motivated design principles and verified via numerical simulations.
The universal parameter regime can be described by the rough equivalence of all the relevant energy scales:

$$\bar{\Omega} \sim \bar{V} \sim \bar{\lambda} \sim (\bar{\Delta t})^{-1} \sim (\bar{\Delta\tau})^{-1} \tag{4.3.3}$$

This condition ensures that no single process (encoding, interaction, or measurement) dominates the quantum evolution, allowing the system to exploit its full dynamical complexity. Physically, this regime emerges from the need to balance key energy and time scales. If, for instance, the scale responsible for mixing (i.e., Rabi oscillations) is much smaller than the interaction strength, the resulting dynamics becomes slow and fails to generate sufficient entanglement. Conversely, if the interactions are too weak compared to the encoding-driven terms, the system cannot correlate features effectively, leading to poor transformation capacity.
A similar argument applies to the encoding scale: if it is too small, it cannot imprint the data meaningfully into the quantum state; if too large, the system approaches a quasi-classical limit where the quantum reservoir ceases to induce non-trivial transformations. Likewise, if the probing and encoding frequencies are much higher than the characteristic energy scales of the Hamiltonian, the system may yield highly redundant embeddings, which increase dimensionality without improving performance and can reduce effective trainability. On the other hand, probing too slowly may result in accessing observables that have already thermalized or decohered, thus carrying little information about the input. Altogether, the universal regime represents a balanced configuration where all relevant processes are simultaneously active and mutually reinforcing, enabling effective and robust QRC performance.
Impressively, this balanced configuration of scales not only removes the need for any optimization or fine-tuning procedure, but also demonstrates a remarkable level of generality: the underlying physical reasoning that leads to the identification of the universal parameter regime applies consistently across different encoding strategies. This suggests that the QRC framework possesses a form of structural robustness, where optimal performance can be achieved through principled parameter design rather than task-specific training. The transferability of these design principles across encodings is a strong indication of the universality of the QRC approach itself, highlighting its scalability and adaptability to a broad class of learning problems without the need for re-optimization. Evidence of the existence and consistency of the universal parameter regime comes from numerical simulations performed across different tasks and encoding strategies.

In Fig. 4.3.3, by scanning key ratios between the characteristic energy scales (encoding-to-mixing and interaction-to-mixing ratios) one observes that optimal performance tends to concentrate around configurations where these quantities are of comparable magnitude. This behavior emerges as a broad region of enhanced accuracy, stable across moderate parameter variations, and has been consistently identified in benchmarks involving both local pulse and position encodings. Interestingly, the location of the optimal regime can often be predicted from simple estimates based on

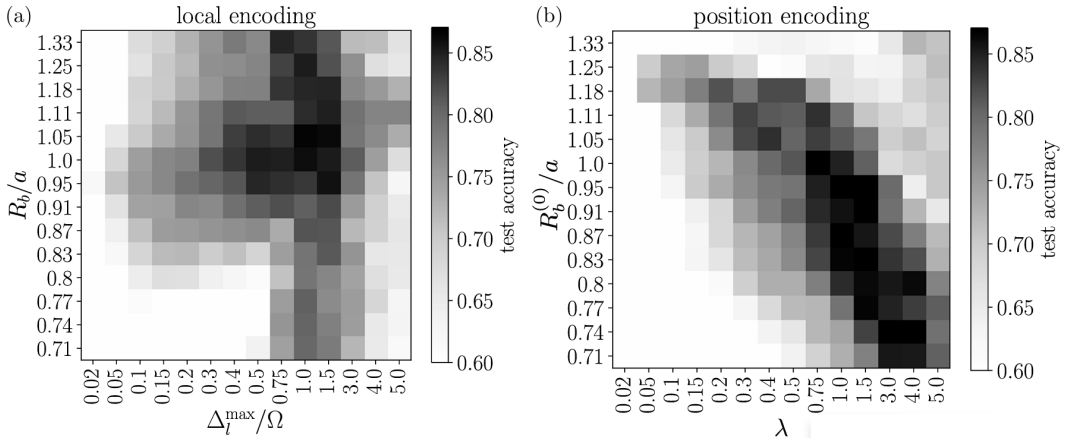

Figure 4.3.3: An example of universal parameter regime in numerical simulations of QRC.(a) Test classification accuracy of local pulse encoded QRC on an 8-PCA 10-class MNIST task as a function of local pulse encoding scale (Δ_l^{max}/Ω), and effective blockade radius (R_b/a).(b) Test accuracy of the position-encoded QRC on the same task as a function of position encoding scale (λ) and effective initial blockade radius ($R_b^{(0)}/a$) [33].

the typical scale of the input features, further reinforcing the physical intuition behind the design of this parameter regime and reducing the need for exhaustive parameter sweeps.
Beyond its dynamical role, the evolution of the quantum reservoir also has a direct interpretation in terms of quantum kernel methods. Specifically, the reservoir embeddings obtained after evolution can be viewed as mapping the classical inputs into a high-dimensional feature space, where the inner products between embedding vectors define a kernel function.
That is, given two inputs $x[n]$ and $x[m]$, the kernel is defined as:

$$K(\mathbf{x}[n], \mathbf{x}[m]) = \langle \mathbf{u}[n], \mathbf{u}[m] \rangle \tag{4.3.4}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in the embedding space.
This kernel matrix redefines the notion of similarity between inputs, effectively altering the geometry of the data in a task-dependent way. Such a perspective reinforces the idea that the QRC evolution implicitly performs a powerful, non-linear feature transformation analogous to kernel methods in classical machine learning.

4.3.3 Data processing

The last step in the QRC pipeline is to extract relevant information from the reservoir and train a simple model to evaluate the performance of the approach.
To obtain measurement results from the evolved dynamics, the system must be probed at several successive time steps, with each instance repeated using N_s measurement shots. From these measurement results, one can compute expectation values of local observables, e.g., $\langle Z_i \rangle$ and $\langle Z_i Z_j \rangle$, which represent the enriched input features encoded in the quantum reservoir.
After data extraction, the flattened vector of local observable expectation values forms the QRC embeddings, namely $\mathbf{u}_i$. The final step is classical post-processing, where a supervised machine learning model is trained. In practice, since the dynamics induced by the reservoir is highly nonlinear, a simple linear support vector machine (SVM) is typically sufficient to achieve good performance on the test set.

Once the data has been processed and trained a model with different algorithms depending on the task assessed (classification or regression) a key step for understanding the QRC performances is to compare this new method with standard classical machine learning models.
A natural baseline for comparison is to assess whether the quantum reservoir dynamics actually implement a meaningful nonlinear transformation of the input data. To this end, the authors consider a version of the QRC pipeline in which the quantum evolution step is omitted. In this setting, the classical SVM is trained directly on the raw input features (those encoded in the Hamiltonian), using the same training protocol and hyperparameter settings. This setup effectively represents a linear model acting on unprocessed data.
In addition to this linear baseline, nonlinear classical models were also explored. One example is a fully connected feedforward neural network with four layers, trained on the same input features. Furthermore, due to its conceptual similarity with kernel-based quantum methods, a Gaussian kernel (RBF) SVM was tested, providing a nonlinear mapping of the data through kernel geometry. Finally, an important comparison was made with the Classical Reservoir Computing (CRC) counterpart, obtained by taking the classical limit $\mathbf{S} \to \infty$ of the Rydberg Hamiltonian. The embeddings were constructed from the classical spin dynamics, and fed into the same linear SVM models as in the QRC pipeline. These comparisons serve to evaluate the expressive power of quantum correlations and the advantage brought by the quantum evolution itself.
A more detailed discussion of the performance differences is presented in the next section, where benchmark results are analyzed.

4.3.4 Results on Aquila hardware

Having described the key components involved in designing an effective QRC pipeline, this section presents the results of the extensive experimental studies conducted on the Aquila neutral-atom quantum processor.

Timeseries prediction

One of the most successful applications of classical reservoir computing has been in timeseries prediction, largely thanks to the *echo state property* first referred in [22] that ensures stable and memory-rich dynamics. Motivated by these results, and by the particularly natural way in which temporal data can be encoded in QRC, through global pulse encoding, the first task explored is a timeseries forecasting benchmark. More specifically the QRC process have been applied to process the Santa Fe timeseries task, which represents the intensity profile of a laser in a chaotic regime. All the three different encodings has been applied for this tasks, enabling the establishment of a hierarchy of encodings, as clear from Fig. 4.3.2.
In all encoding schemes, the feature extraction procedure follows the same principle: features are obtained from a time window of d steps, and the task is to predict future values of the sequence. The Santa Fe laser timeseries task served as a benchmark to evaluate the QRC algorithm across the three encoding strategies—global pulse, position, and local detuning—all implemented on 12-qubit chains with consistent probing and evolution protocols.
Global pulse encoding provides a particularly natural framework for temporal data, as time-window features are directly mapped onto segments of a piecewise linear global detuning pulse. This results in intuitive and expressive embeddings, where local observables reflect different aspects of the input. However, its performance was more susceptible to experimental noise, resulting in a noisy background but still enabling feature interpretability (see Fig. 4.3.4).

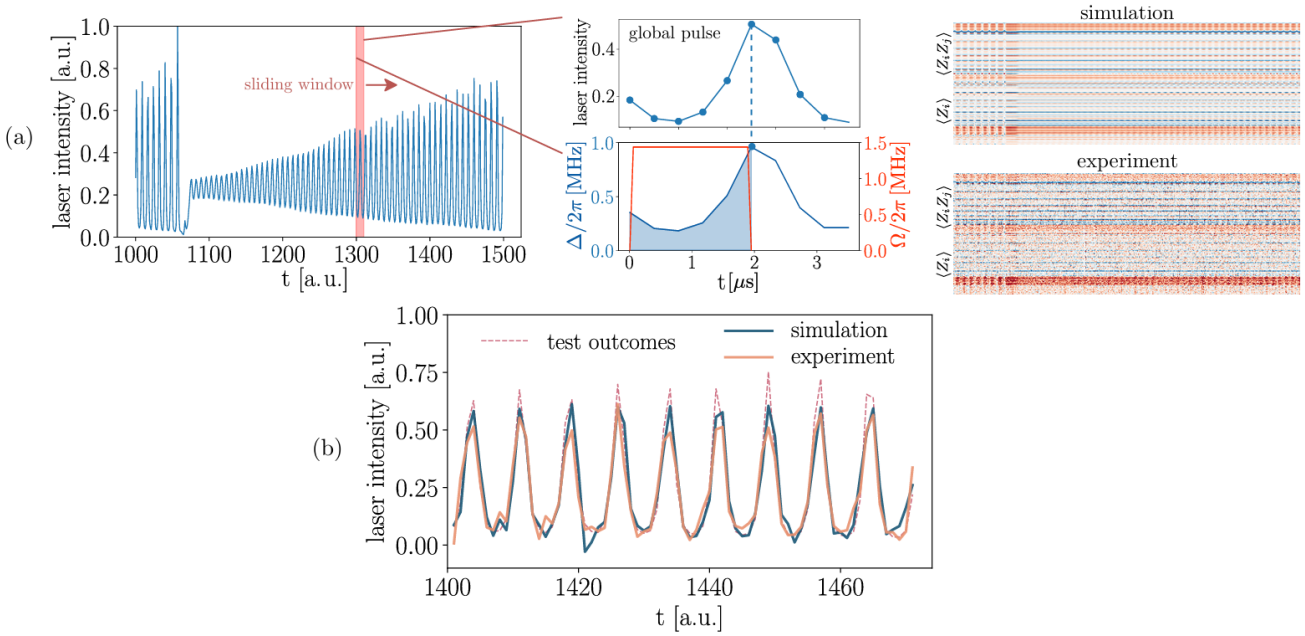

Figure 4.3.4: Timeseries prediction with QRC. (a) The profile of the window feature is encoded into the piecewise linear global detuning pulse. The selected local observables obtained by probing the quantum evolution are shown for exact simulation and experiment, with the vertical axis corresponding to different embedding components and the horizontal to the timeseries steps. (b) An example of test outcomes predicted by local pulse encoded QRC with 12 qubits, compared to the equivalent finite-sampled simulation (110 shots per data point) and the true outcomes [33].

On the other hand, position and local pulse encodings showed greater robustness. In particular, local pulse encoding produced predictions that closely matched simulated results and test outcomes, except for minor deviations at the edges of the dynamic range. Importantly, the relative performance ranking observed in simulation, local and position encodings outperforming the global pulse, was preserved experimentally, confirming the reliability of the encoding hierarchy even in the presence of hardware noise.

MNIST classification

The QRC algorithm is also suited for image classification tasks. In the paper they applied the pipeline to the MNIST dataset, a dataset containing gray-scaled handwritten digits (from 0 to 9) , each image is a 2D matrix composed of 28×28 pixel. As a starting point the binary classification of digits 3/8 has been studied, in order to check the performance of the QRC algorithm and evaluating the comparison with other classical and non-linear approaches.
In order to fit the Aquila's hardware constraint, images are down-sampled into an 8-dimensional vector using principal component analysis (PCA). The extracted features are then positionally-encoded in a 9-atom qubit chain. In order to facilitate and optimize hardware runtime quantum dynamics has been performed in parallel across six decoupled chains, collecting N_s measurements per embedding.

The first classification task explored has been the 3/8 binary classification, using 1000 train and

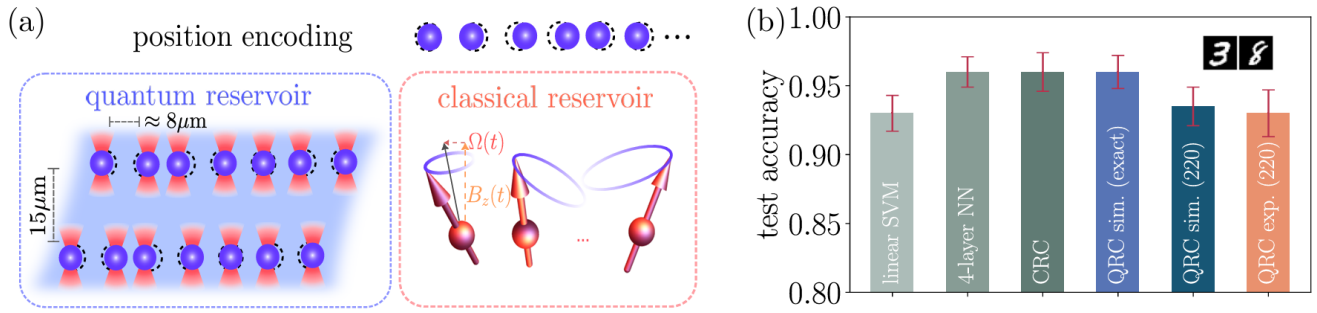

Figure 4.3.5: Binary classification with QRC. (a) The MNIST images of handwritten digits are down sampled to feature vectors that are encoded into the modulation of the nearest neighbor Rydberg interaction strengths via the position encoding. The quantum reservoir consists of parallel, well-separated neutral-atom chains evolving under the Rydberg Hamiltonian. The equivalent classical spin reservoir(CRC), where the vector spins precess in the external and neighbor magnetic field, is simulated for comparison. (b) The test classification accuracy of several classical machine learning methods and QRC on the 3/8-MNIST binary classification tasks [33].

200 test data. In order to evaluate the performance of the method both quantum and classical reservoir has been tested, Fig. 4.3.5 (a). In the Classical Reservoir Computing (CRC) framework, the quantum spins (qubits, treated as spin-1/2 particles) are replaced by classical unit vectors. This transforms the quantum system into a classical one, where each spin evolves according to a classical precession equation:

$$\frac{d\hat{S}_i}{dt} = \frac{\partial H[\hat{S}]}{\partial \hat{S}_i} \times \hat{S}. \tag{4.3.5}$$

Here, the term $\partial H[\hat{S}]/\partial \hat{S}_i$ acts as an effective magnetic field experienced by spin $\hat{S}_i$, given by:

$$\frac{\partial H[\hat{S}]}{\partial \hat{S}_i} = \frac{\Omega(t)}{2}\hat{x} + \left[-\frac{\Delta_i(t)}{2} + \frac{1}{4}\sum_{i\neq j} V_{ij}\left(1 + \hat{S}_j^{(z)}\right)\right] \tag{4.3.6}$$

This formulation enables the simulation of the system by numerically integrating a set of 3 N_q classical differential equations, one for each spatial component of each spin. The CRC procedure mirrors the QRC pipeline, using the same time-dependent control parameters and Hamiltonian profiles. The features used for learning tasks are extracted from the z-components of the classical spins at readout time, along with their pairwise products. Depending on the application, the same linear models employed in QRC are used for training and prediction.
Importantly, since each classical spin configuration corresponds uniquely to a quantum product state, the CRC can be interpreted as the non-entangled classical limit of the QRC.
The test accuracy comparison is shown in Fig. 4.3.5(b), it is clear how linear methods do not suffice in capturing the data expressivity, non linear methods are needed. QRC, CRC and 4 layer-neural network reaches same performance, when QRC is exactly simulated. performances in quantum hardware are constrained by the finite number of shots $N_s = 220$, nevertheless QRC in real hardware and with constrained shot number is still capable of reaching comparable performance with other CML techniques, reaching 0.935 test accuracy.
To further investigate how the shot number influence the QRC's performances, the 10-class MNIST task has been explored. In this case 1500 train and 500 test data has been used. The 10-class

MNIST classification task is remarkably more complex on current quantum hardware but serves as a more stringent test for experimental noise.

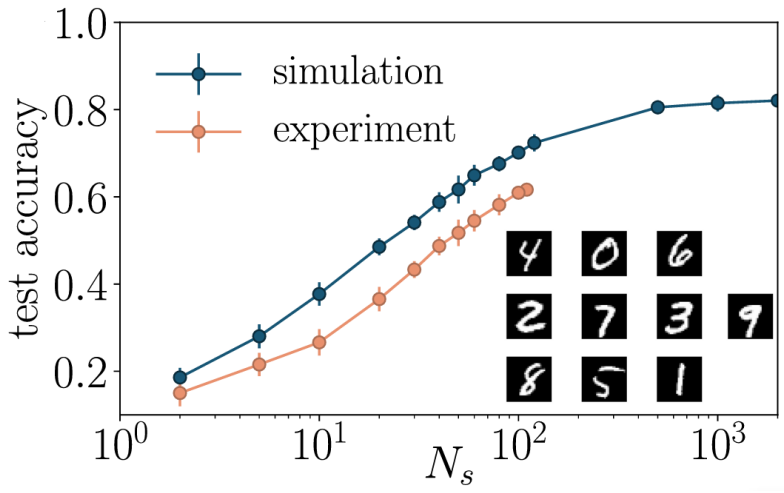

Figure 4.3.6: Test accuracy of QRC simulation and experiment as a function of the number of shots drawn per data point, N_s for the 10-class MNIST classification task [33].

From the previous plot QRC performances tends to plateau approximately $N_s \sim 1000$. Therefore, in order to fill the performance gap, among simulation and experimental results on QRC, in Fig. 4.3.5(b) increasing the number of shots per data point should be sufficient.
This performance gap is strictly connected to the stochastic nature of quantum mechanical measurements, a limited number of of samples introduces statistical fluctuations. This effect introduces a sampling overhead, which, in this context, remains moderate due to the use of local observables for generating the reservoir features. In practice, performance improves as more shots are used, up to a point where it stabilizes and becomes nearly indistinguishable from that of the noiseless simulation. Interestingly, this saturation point appears to be insensitive to the size of the quantum system, and instead is likely determined by factors such as the complexity of the classification task and the inherent noise in the dataset itself. This implies that once a sufficient number of measurements is reached, further increasing them has a negligible impact on the accuracy.

Tomato leaf classification

In the previous use cases, the QRC algorithm demonstrated solid performance across different tasks, achieving good agreement with test outputs in regression problems and outperforming linear baselines in classification tasks. These results highlight the non-linear transformations induced by the quantum reservoir. The authors initially tested the algorithm on small-sized reservoirs, observing a consistent advantage over linear methods and confirming the effectiveness of the QRC approach in both regression and classification settings. To further investigate the scaling behavior and predictive accuracy of the model, the algorithm was subsequently tested on larger quantum reservoirs, reaching up to 108 qubits, a substantial leap compared to previous quantum machine learning results e.g., [34].
For this purpose, the authors considered a tomato disease classification task based on leaf images. The dataset consisted of three classes of tomato leaf images, with a total of 498 samples, 400 of

which were used for training. All images were converted to grayscale and resized to a uniform resolution of 256×256 pixels. Feature extraction was performed by down-scaling each image to a lower resolution of $R_x \times R_y$ pixels, where each pixel value represents the average intensity over a square region centered at the corresponding location in the original image. These down-scaled features were then encoded into a two-dimensional Aquila array of $N_q = (R_x + 1) \times R_y$ qubits using the position encoding strategy, as illustrated in Fig. 4.3.7.

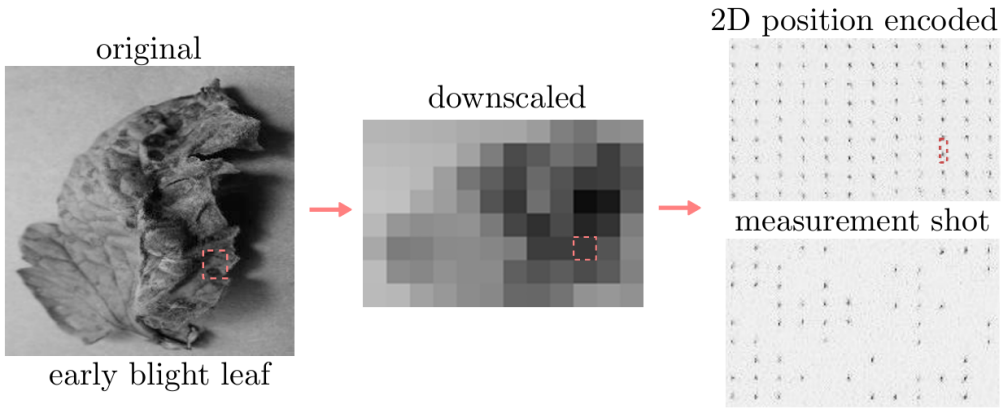

Figure 4.3.7: Down-scaling procedure of tomato leaf images [33].

In order to investigate the scaling behaviour of the QRC, several N_q has been tested, reaching the final configuration of a rectangural geometry of 9×12.
The results showing the test accuracy as a function of the system size, is shown in Fig. 4.3.8, where it is compared with classical linear and non-linear models. The experiment conducted with

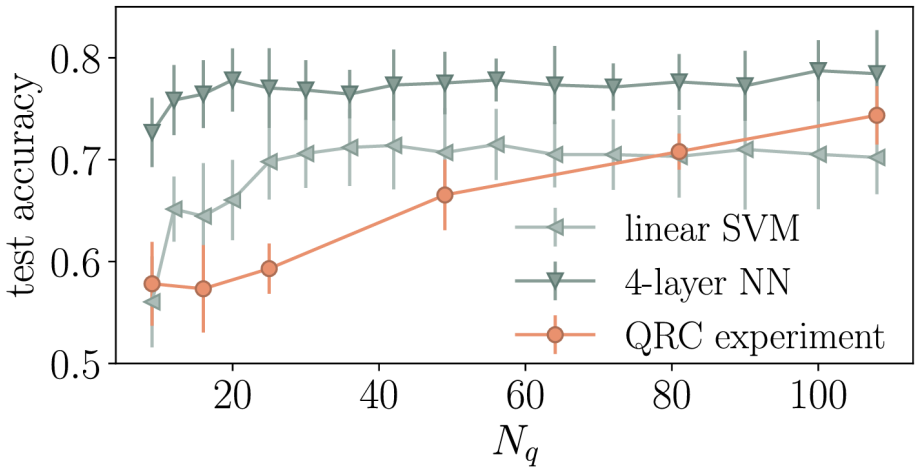

Figure 4.3.8: Test accuracy as a function of qubit numbers [33].

108 qubits represents a significant step forward in terms of scale for quantum machine learning implementations. At this size, the QRC not only outperforms the linear support vector machine baseline, but also begins to approach the performance of a fully classical 4-layer neural network containing approximately 20000 trainable parameters. Interestingly, the performance does not

appear to saturate at 100 shots per data point: results obtained with 1000 shots suggest a potential for further improvement, indicating that increased sampling may still offer benefits.

5 Results and Discussion

This chapter presents the original contributions of this thesis, focusing on the implementation and evaluation of the Quantum Reservoir Computing (QRC) algorithm on a neutral-atom quantum platform. Building on the theoretical foundations and algorithmic framework described in previous chapters, the aim here is to explore how quantum-enhanced embeddings can improve the performance of supervised learning tasks compared to fully classical approaches.

The discussion begins with an overview of Pasqal's Pulser library, the tool used to simulate and control neutral-atom devices at the pulse level.

Subsequently, the chapter details the problem setup, including the dataset selection, classical pre-processing steps, and the design choices for the quantum reservoir. This section establishes the parameters and constraints under which the QRC algorithm is applied.

Finally, the chapter presents the results obtained with different encoding strategies, highlighting how global and local encoding schemes affect the expressivity of the quantum reservoir. Comparisons with classical methods and insights gained from the simulations conclude the discussion.

5.1 Implementation on the Pasqal platform

Pasqal, a leading company in the development of neutral atoms quantum computers, have recently presented Pulser, an open-source Python library for programming neutral atom devices at the pulse level. The low-level nature of Pulser makes it a versatile framework for quantum control both in the digital and analog framework [35].
Pulser allows full control over all physically relevant parameters, enabling the creation of programs that can either be run locally using the built-in emulator, which faithfully reproduces the hardware behavior, or sent directly to the QPU to execute the sequence of pulses composing the simulation. In order to run a full simulation on a Pasqal device we need to carefully go through each needed step presented in Figure 3.4.1.

In Pulser the first step to run a simulation is the initialization of atoms in trapped arrays, this process goes under the name of *Register* creation. A register is a collection of atoms and their positions. For each atom in a register, the quantum information is encoded in specific electronic energy levels, which are reached through optical addressing techniques. In Pulser the addressing of single or multiple atoms is enabled by *Channels* and the emission of *Pulses*.
A pulse is defined as the modulation of a channel's output amplitude, detuning and phase over a finite duration. For a channel targeting the transition between energy levels a and b, with reso-

nance frequency $\omega_{ab} = |E_a - E_b|/\hbar$, the output amplitude determines the Rabi frequency $\Omega(t)$, and the detuning $\delta(t)$ is defined relatively to ω_{ab} and the frequency of the channel's output signal $\omega(t)$, as $\delta(t) = \omega(t) - \omega_{ab}$. Additionally, the phase ϕ of a pulse can be set to an arbitrary, constant value. Furthermore, Pulser enable the control over different basis for the quantum computation and thus, different types of interactions. In what follows we will focus on the so-called "Ising" configuration, obtained when the spin states are one of the ground state $|g\rangle$ and a Rydberg state $|r\rangle$, referred in Pulser as the *ground-rydberg* basis. A described in Eq. 3.2.1 the Hamiltonian describing the transition between those states is dependent on the constant parameter C_6 and the Rydberg blockade effect, the interaction prevents the simultaneous excitation of two atoms in the state $|r\rangle$ if $\hbar\Omega << C_6 R^6$.

Program structure

The elements necessary for executing a program on Pulser are illustrated in Fig. 5.1.1 and can be summarized as follows:

- **Device** consists of a series of specifications that characterize the hardware, including the chosen Rydberg level, the ranges of the amplitudes and frequencies of the lasers, the minimal and maximal distance between the atoms, and the different channels that can be declared.
- **Register** stores the information about the coordinates of the atoms and their respective ID's, which serve to identify them when targeting specific operations.
- **Channels** represent the action of the lasers and are organized by addressing (local or global) and the type of transition (Rydberg or Raman).
- **Waveforms** are the basic building blocks of a pulse. They can have custom or predetermined shapes, all of them indicating the specific profile of the waveform and its duration.
- **Pulses** consist of waveforms for the amplitude and the detuning. They can be further shifted by a phase. Once a Pulse is constructed it has to be added to the sequence indicating which atom(s) are targeted and what channel will implement it.
- **Sequence** contains the schedule of the pulses in each channel. It is also linked with a Register and the Device in which it is to be executed. This is the information that can be sent to a real neutral-atom QPU or simulated on a classical computer.
- **Simulation** is included to emulate results for the application of sequences. In Pulser, we have made use of the QuTiP library [36] for the simulation of quantum systems. Each simulation run returns a specialized object that holds the results and features methods for postprocessing them.

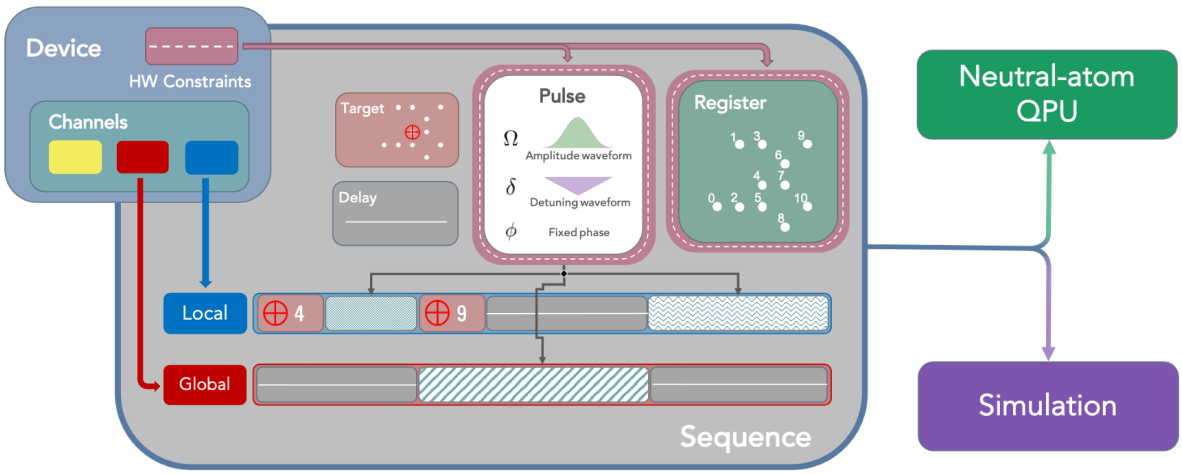

Figure 5.1.1: Relationship between the main Pulser classes. The central object is the Sequence, which is linked to a Device. The Device holds the available Channels, which are selected and declared in the Sequence, and information of the hardware constraints. These constraints are enforced upon the Register, where the neutral-atom array is defined, and upon the Pulses. Each Pulse, defined by its amplitude and detuning Waveforms and a fixed phase, populates the declared channels alongside other commands like target, which points local addressing channels to specific qubits, and delay, which idles the channel. The resulting Sequence can then be sent for execution on the neutral-atom QPU or emulated through Pulser's Simulation class [35].

5.2 Problem set-up

In this section, we will describe the specific problem addressed in this thesis. The main objective is to evaluate the performance of the QRC algorithm on a 10-class supervised machine learning task and to compare the performances with those obtained using a fully classical pipeline.

5.2.1 Dataset

The reference dataset used is the "Modified National Institute of Standards and Technology", also known as MNIST dataset [37]. The dataset is a collection of 70000 handwritten digits from 0 to 9, with each image being 28x28 pixels.
In more details the dataset is composed by:

- **Number of instances**: 70000 images divided in 60000 representing the training set and 10000 the test set.
- **Number of Features**: 784.
- **Pixel**: each image is a composed by 784 pixels with values ranging from 0 to 255 representing the grayscale intensity of the corresponding pixel.

This dataset is well suited for benchmarking classical supervised machine learning. However, given the current hardware limitations of NISQ devices and the actual Pasqal QPU (see Section 3.4), we decided to use a reduced version of the original MNIST dataset that is compatible with these constraints. This reduced version allows us to map each feature to a controllable parameter on

the Pulser without exceeding hardware limitations.

This alternative version is composed:

- **Number of instances**: 1797.
- **Feature number**: 64.
- **Pixel**: 8x8 image of integer pixels in the range 0 to 15.

is provided by scikit-learn and can be loaded via the function `load_digits()` [38].
An intuitive understanding of the dataset is provided in 5.2.1 that shows a selection of handwritten digits from the scikit-learn digits dataset.

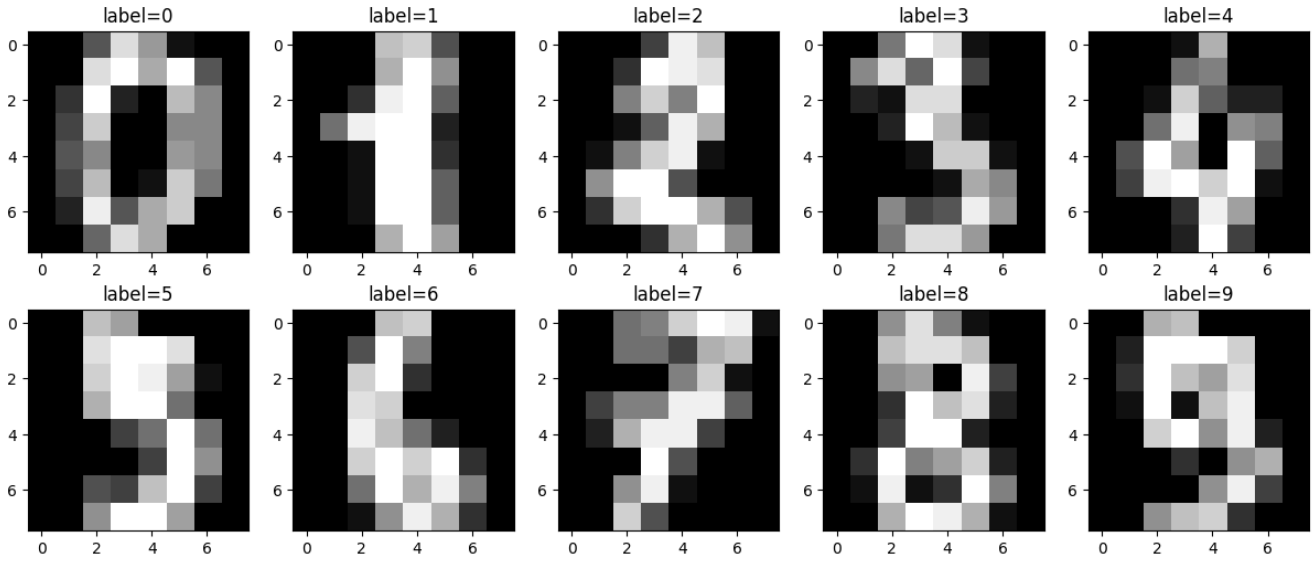

Figure 5.2.1: Examples of handwritten digits from the scikit-learn digits dataset. Each image comes with his associated label and are composed by 8x8 pixels, with grayscale values ranging from 0 to 15.

5.2.2 Pre-processing

The reduced dataset contains 64 features per instance. Mapping this amount of feature to the quantum reservoir is not feasible, as there is no encoding techniques that enable the simulation of 64 features classically, this would exceed the classical computational resources available. Therefore, to address this limitation we applied PCA to reduce the dimensionality of the inputted feature.
PCA is a linear dimensionality reduction technique that linearly transform the original data onto a new coordinate system such that the directions (principal components) capture the largest variation in the original dataset. By projecting high-dimensional data onto the directions of maximum variance, PCA reduces model complexity while preserving the most relevant features.
As showed in Fig. 5.2.2, for this work we selected the first 5 principal components. From the 6^{th} component onward, there is significant overlap between classes, indicating that these directions add minimal useful information for the classification task. Once completed PCA reduction and

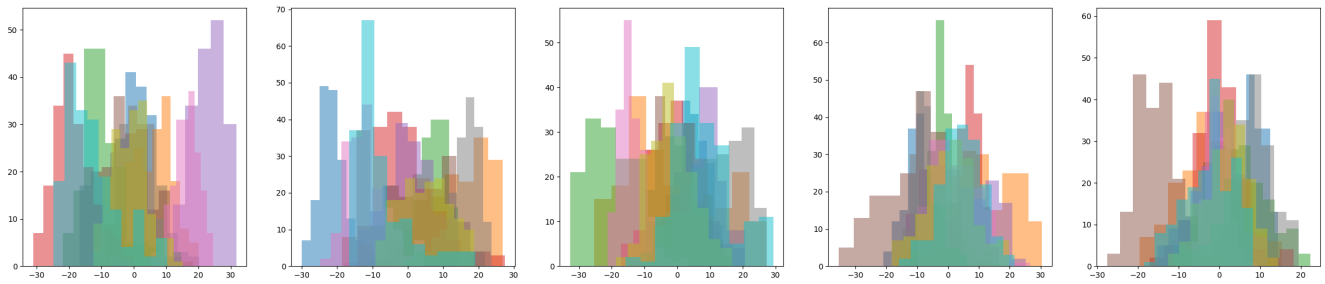

Figure 5.2.2: Histograms of the PCA-transformed features for the reduced digits dataset. Each subplot corresponds to one principal component obtained from PCA. The histograms display the distribution of values for each component, separated by digit class (from 0 to 9). Overlapping colors indicate contributions from different classes, allowing comparison of their distributions along each principal component.

obtained the new coordinate system the next step is to split the original dataset of 1797 instances into a train and test set. In order to do so, we used the `train_test_split()` function `from sklearn.model_selection` and defined the train set as 70% out of the full dataset and the test set as 30%.
The last step required in order to avoid bias in the model is to normalize the data feature to the range $[0, 1]$.
The preprocessed features are now prepared for encoding into the quantum reservoir.

5.2.3 Quantum Reservoir design

Once the data have been classically pre-processed, they are ready to be fed either into the quantum reservoir or directly into a classical ML model for performance comparison.

Device

The first step in setting up a proper Quantum Reservoir is the choice of the encoding technique. As discussed in Sec. 4.3.1, different encoding strategies are, in principle, possible. However, Pasqal QPU has limitations that prevents control of single atoms. Therefore, positional and local encodings are not allowed on the QPU.
In the following, although we work with emulators of the real QPU, we enforce the same hardware restrictions in order to ensure consistency with potential results obtained on the actual device. In Pulser, these restrictions are encoded in the device specification. In particular, we adopt the constraints of the real device `FRESNEL` as reference:

- **Dimensions**: 2
- **Minimum atom distance**: 5 μm
- **Maximum number of atoms**: 61
- **Maximum layout filling fraction**: 0.5
- **Maximum sequence duration**: 6000 ns

- **Pre-calibrated layout**: triangular lattice
- **Channels**: global addressing
- **Maximum absolute detuning**: 48.69 rad/μs
- **Maximum Rabi amplitude**: 12.56 rad/μs

This device is not the one directly used for the simulations, but it provides reference values that guide the emulated experiments. The distinction between global and local addressing corresponds to two different device types, which will be discussed in Sec. 5.3.

Register

Once the hardware limitations imposed by the QPU are defined, the next step is to build a register starting from the available pre-calibrated layout. As shown in Fig. 5.2.3, the pre-calibrated layout at this stage corresponds to a triangular lattice.

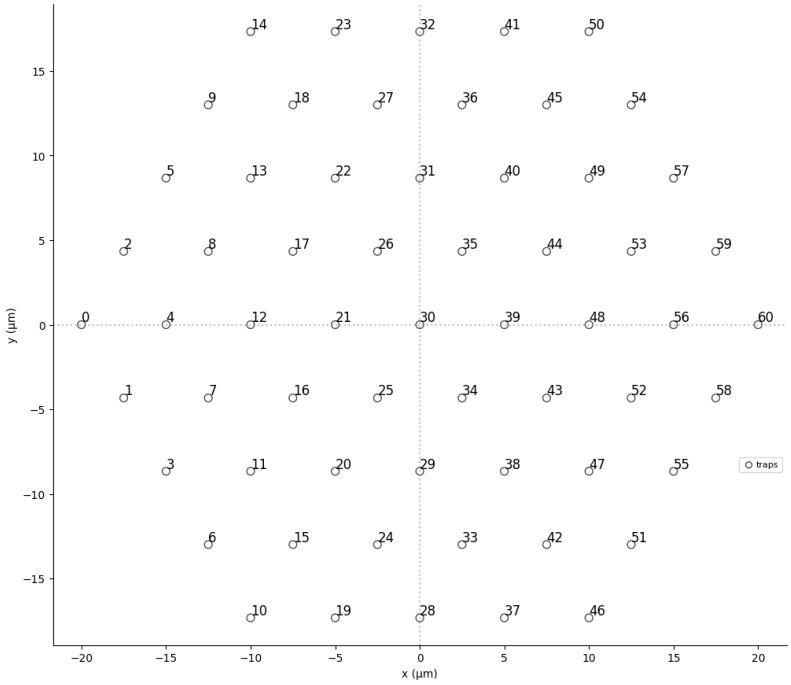

Figure 5.2.3: The triangular lattice layout as available for QPU emulation. Each ∘ is a trap that can be filled with an atom.

According to the pre-processing step, we retain 5 PCA-reduced features out of the full set of 64 per instance. This information is used to build the register on top of the pre-calibrated layout. Consistently with the PCA reduction, for both global and local encoding, the register will consist of 5 atoms selected from the available traps, forming a linear chain in which the PCA-reduced features are encoded.
Along with the definition of the register, another fundamental parameter that must be specified in order to enable interactions and entanglement among atoms is the Rydberg blockade radius. According to the principle outlined in Sec. 4.3.2, the blockade radius should not be too large (which would lead to excessive interactions and unrealistic simulation times), nor too small (in

which case no interaction would occur). Given the spatial separation of 5μm between adjacent atoms along the same row, a blockade radius of 7μm was adopted in the simulations, as shown in Fig. 5.2.4.
Based on the blockade radius, a built-in function is used to determine the magnitude of the Rabi frequency required to construct the sequence according to the selected encoding technique.

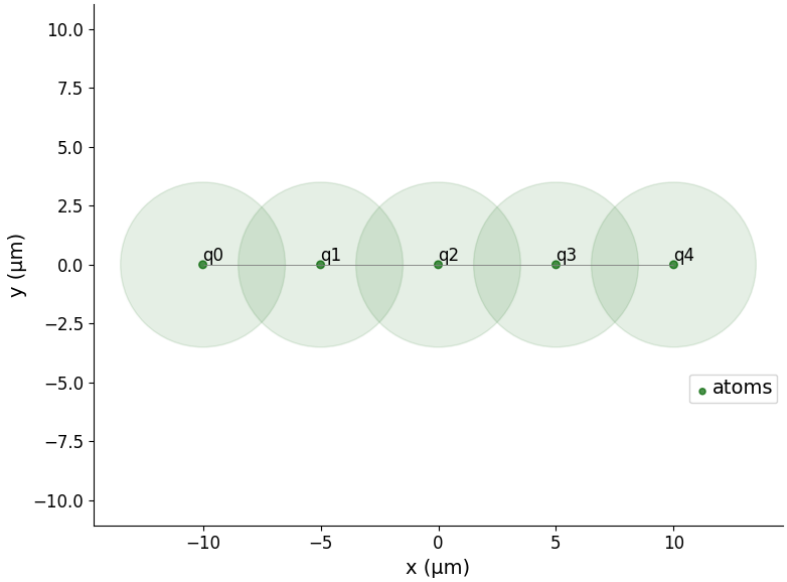

Figure 5.2.4: Register used for the QRC algorithm. Green dots represent qubits while light green circles denote the interaction range between them, defined by the blockade radius.

Simulation

The final step before running the simulation is the definition of the sequence and channel, which depend on the chosen encoding technique and will be detailed in the following sections.
Once the register has been defined and the constraints of the real device enforced, the simulation of the dynamics can be performed through the Pulser emulator based on QuTiP. The emulator is initialized as:

```
emulator = QutipBackend(sequence=sequence, mimic_qpu=True)
results = emulator.run()
```

where the option `mimic_qpu=True` ensures that the same hardware constraints of the Pasqal QPU are applied during the emulation.
In order to extract meaningful observables from the quantum evolution, we define both single-qubit and two-qubit operators. The single-qubit operators correspond to the Pauli-Z expectation values on each atom, while the two-qubit operators encode correlations between pairs of atoms within a cutoff distance, here set to 7 μm to match the blockade radius:

$$Z_i = \sigma_z^{(i)}, \qquad Z_i Z_j = \sigma_z^{(i)} \otimes \sigma_z^{(j)} \quad \text{if } \|r_i - r_j\| \leq 7 \ \mu\text{m}.$$

For each input sample, the corresponding sequence is run on the emulator and the quantum state is obtained at different time steps. At every step, we compute the expectation values of the Z

and ZZ operators, which together form the embedding vector associated with that time step. Formally, for input x and time t, the embedding reads:

$$\mathbf{e}(x,t) = \big(\langle Z_1\rangle, \ldots, \langle Z_N\rangle, \langle Z_1 Z_2\rangle, \ldots\big).$$

By concatenating the embeddings across all time steps, we obtain a high-dimensional representation for each input, which is then used for the subsequent machine learning tasks:

```
embeddings_train = build_embeddings(x_train, QRC_parameters, positions)
embeddings_test = build_embeddings(x_test, QRC_parameters, positions)
```

5.2.4 Post-processing

After obtaining the embeddings from the quantum reservoir, we evaluate their performance on a supervised classification task. The main objective of this post-processing step is to assess how well the quantum embeddings capture the relevant features of the input data, and to compare the results with a classical baseline.

Model selection and training

To this end, a logistic regression classifier is trained on the embeddings. Logistic Regression is a supervised machine learning algorithm used for classification tasks, it predicts the probability that an observation belongs to one or more discrete classes. The model belongs to the family of Generalized Linear Models (GLM) and relies on the logistic (or sigmoid) function to map a linear combination of inputs into a value within the range $[0, 1]$.
In the binary case, given an input $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, the probability that the output is $y = 1$ is modeled as:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) \tag{5.2.1}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function.
The model does not estimate the probability directly, but the log-odds (the logarithm of the odds ratio):

$$\log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \tag{5.2.2}$$

The coefficients β_i are estimated via Maximum Likelihood Estimation, i.e., by maximizing the likelihood of the observed data.
For our 10-class classification problem, we use Multinomial Logistic Regression. In this case, the probability that an observation belongs to class $k \in \{1, \ldots, K\}$ is given by:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\beta_{k0} + \beta_{k1} x_1 + \cdots + \beta_{kp} x_p)}{\sum_{j=1}^{K} \exp(\beta_{j0} + \beta_{j1} x_1 + \cdots + \beta_{jp} x_p)}, \tag{5.2.3}$$

The predicted class is then the one with the highest probability:

$$\hat{y} = \arg\max_k \; P(y = k \mid \mathbf{x}). \tag{5.2.4}$$

To reduce the risk of overfitting, logistic regression is often combined with regularization terms. In particular, one can use:

- **L1 (Lasso)**: promotes sparsity of the coefficients;
- **L2 (Ridge)**: penalizes large coefficients and stabilizes the solution;
- **Elastic Net**: combines both approaches.

In our case, the logistic regression has been employed as the final classifier to compare the classical data and the quantum-generated embeddings [39].

Hyper-parameters tuning

To improve the performance of the logistic regression model and avoid underfitting or overfitting, we perform hyperparameter tuning using *Grid Search*.
Grid Search is a systematic approach to explore a predefined set of hyper-parameters and identify the combination that maximizes the model's performance. The parameters of the estimator are optimized via a cross-validation procedure: the original dataset is divided in 5 equal parts, at each interaction 4 of them are used to train and the remaining 1 for test. Every possible combination of train and test split is tested in order to reduce the risk of overfitting.
In our case, the hyper-parameters to tune are:

- **Penalty**: type of regularization, such as L1, L2, or Elastic Net;
- **Regularization strength** (C): inverse of regularization strength, controlling the amount of shrinkage applied to coefficients;
- **Solver**: the optimization algorithm used to fit the model (e.g., `saga`);
- **L1 ratio**: used for Elastic Net, balancing L1 and L2 contributions.

After identifying the best combination of hyper-parameters through cross-validation, the selected logistic regression model is applied to both the classical and the quantum-enhanced datasets. The performance of the model is then evaluated using the accuracy score, and the classification results are further analyzed with a confusion matrix.
A confusion matrix is a table that summarizes the performance of a classification model. More specifically, a confusion matrix C is such that C_{ij} is equal to the number of observation known to be in group i but predicted to be in group j. Thus in binary classification, the count of true negatives is C_{00}, false negatives is C_{10}, true positives is C_{01} and false positives is C_{11} [40].
Having established the framework of the algorithm, we next investigate two distinct encoding strategies and their outcomes.

5.3 Global encoding

In the context of neutral-atom quantum computing, a natural strategy for encoding data consists in mapping the input features onto the time-varying profile of the global detuning pulse:

$$\Delta_g(t_i) = \lambda \Delta_g^{\max} x_i, \tag{5.3.1}$$

where $\Delta_g(t_i)$ denotes the global detuning at different time intervals, λ is the encoding scale (properly tuned as discussed in Section 4.3.2), $\Delta_g^{\max}$ is the maximum absolute detuning supported by the `FRESNEL` device, and x_i are the pre-processed and normalized data features.
As explained in Section 5.2.3, the QRC algorithm has been executed locally, with reference parameters derived from the `FRESNEL` device specifications. In this framework, the simulation of the global encoding technique is enabled by the `ANALOG` device, which allows the simultaneous control and evolution of all the atoms in the defined chain register.
The main parameters defining the simulation are the following:

- Number of atoms: 5, matching the reduced dimensionality of the PCA input
- Encoding scale: $\lambda = 1$, ensuring that all relevant energy scales remain comparable
- Blockade radius: $7\,\mu m$
- Rabi frequency: $\Omega = 7.35\,\mathrm{rad}/\mu s$
- Maximum global detuning: $\Delta_g^{\max} \approx 48.7\,\mathrm{rad}/\mu s$, in accordance with `FRESNEL` specs
- Time interval: $t_{int} = 500\,\mathrm{ns}$ for the encoding and probing of each feature
- Total evolution time: $T = 2500\,ns$

The atomic register is initialized on a `TriangularLatticeLayout` of the `ANALOG` device, selecting five traps at fixed positions, as shown in Fig. 5.2.4. Each trap is assigned to a qubit identifier q_i, resulting in a chain register with known inter-atomic distances. The blockade radius is then used to compute the effective Rabi frequency, $\Omega = 7.35\,\mathrm{rad}/\mu s$, which fixes the amplitude of the driving pulse.
For each input vector $x = (x_1, \ldots, x_f)$, we build *one* pulse sequence whose detuning is determined segment-by-segment by the components of x. Specifically, we define target detunings

$$\Delta_g(t_i) = \lambda\, \Delta_g^{max}\, x_i \qquad (i = 1, \ldots, d),$$

and construct a piecewise-linear waveform by concatenating d segments of equal duration t_{int}. For each consecutive pair (Δ_i, Δ_{i+1}) we create a `RampWaveform` of duration t_{int} that interpolates linearly from Δ_i to Δ_{i+1}, while the last segment holds the final value Δ_d for the same duration. The full, input-conditioned detuning profile is assembled as a `CompositeWaveform` with total duration

$$T = d \cdot t_{\mathrm{interval}}.$$

In the case $d = 5$ and $t_{\mathrm{interval}} = 500\,\mathrm{ns}$, the resulting sequence lasts $2.5\,\mu\mathrm{s}$.
For each input x, a fresh sequence is compiled on the `rydberg_global` channel, which drives all atoms simultaneously. A constant-amplitude pulse is applied with

$$\Omega(t) = \Omega, \qquad \Delta_g(t) = \mathrm{CompositeWaveform}(\Delta_g(t_1), \ldots, \Delta_g(t_d)), \qquad \phi = 0,$$

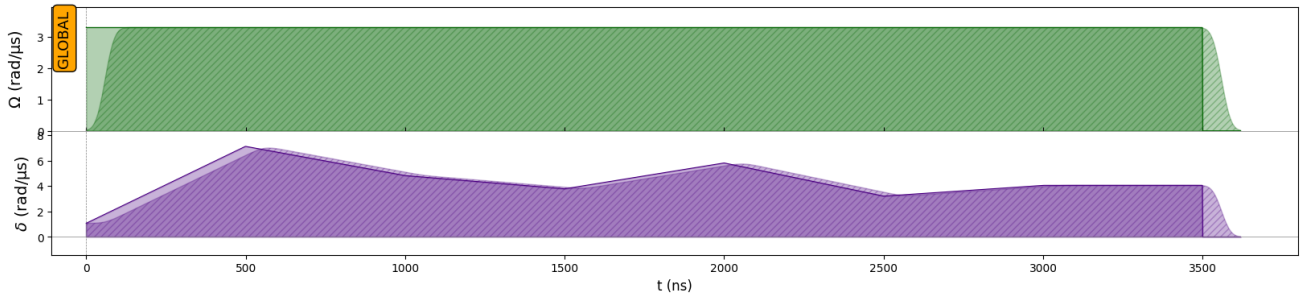

Figure 5.3.1: Typical pulse sequence employed in the global encoding scheme. The plot shows the constant Rabi frequency Ω (green) and the global detuning Δ_g (purple) over time. The data features are encoded in the detuning amplitude. A short initialization delay of approximately 50 ns is present between the trigger signal and the onset of the detuning pulse. This delay is an intrinsic feature of the experimental setup, accounting for hardware response times.

so that the amplitude remains fixed at the Rabi frequency while the detuning follows the input-dependent, piecewise-linear trajectory defined above.
As described in Sec. 5.2.3, the simulation is executed by inputting the desired pulse sequence, with the option to mimic the QPU. The observables used as input for the classical post-processing are single and two-qubit Pauli-Z operators. At each time step and for each feature vector, these measurements are collected into the embeddings matrix, which constitutes the output of the QRC.

Before presenting the results obtained with this global encoding scheme, it is important to highlight a subtlety inherent to this approach. Since the encoding is global, the same detuning profile is applied to all atoms simultaneously. If the register is arranged in a symmetric chain, as in our current setup, significant information can be lost or become redundant: atoms located symmetrically with respect to the chain's center produce identical embeddings, as evident from the time-resolved expectation values shown above (see Tab. 5.1).

	$t = 500$ns	$t = 1000$ns	$t = 1500$ns	$t = 2000$ ns	$t = 2500$ns
$\langle Z \rangle_0$	-0.785	-0.529	-0.513	-0.655	-0.761
$\langle Z \rangle_1$	-0.593	-0.417	-0.820	-0.615	-0.549
$\langle Z \rangle_2$	-0.777	-0.605	-0.625	-0.614	-0.574
$\langle Z \rangle_3$	-0.593	-0.417	-0.820	-0.615	-0.549
$\langle Z \rangle_4$	-0.785	-0.529	-0.513	-0.655	-0.761
$\langle ZZ \rangle_{01}$	0.395	-0.033	0.344	0.284	0.325
$\langle ZZ \rangle_{12}$	0.390	0.034	0.450	0.243	0.147
$\langle ZZ \rangle_{23}$	0.390	0.034	0.450	0.243	0.147
$\langle ZZ \rangle_{34}$	0.395	-0.033	0.344	0.284	0.325

Table 5.1: Time-resolved expectation values of $\langle Z \rangle$ single-qubit and $\langle ZZ \rangle$ nearest-neighbor two-qubit operators for a 5-qubit chain under the sequence showed in Fig. 5.3.1. Symmetry in the chain leads to some redundant embeddings.

In practical hardware such as `FRESNEL`, where atoms are constrained in fixed positions, this symmetry-induced redundancy limits the expressivity of the reservoir. In principle one could mitigate this effect by slightly perturbing the atomic positions to break perfect symmetry. However, the inability to freely place atoms remains a fundamental limitation for the performance of global encoding schemes.
Once the embeddings matrix has been obtained from the QRC simulation, it is used as input for a logistic regression model to perform classification. As described in Sec. 5.2.4, the hyper-parameters of the logistic regressor are optimized via a grid search procedure. Importantly, this optimization is carried out separately for the QRC-enhanced embeddings and for the fully classical embeddings, in order to account for potential differences in feature scale and correlations between the two representations. The resulting best hyper-parameters for each case are reported in Tables 5.3 and 5.4, respectively.

Penalty	C	Test Accuracy
l2	10	0.849
elasticnet	10	0.847
elasticnet	10	0.847
l1	100	0.844
elasticnet	100	0.844

Table 5.4: Top 5 hyperparameter combinations for the logistic regression model using classical features and the saga solver, selected from the full grid search. Test accuracy is reported for each combination.

Model evaluation

After hyperparameter optimization, the QRC-enhanced features are fed into the best logistic regression model, an l_1-regularized regressor, as reported in Tab. 5.3. Figure 5.3.2 shows the test accuracy evaluated on the QRC-enhanced training and test sets as a function of the training set size.
As shown in the plot, the QRC embeddings allow the model to achieve high accuracy even with a small number of training samples, indicating that the quantum encoding captures relevant information for the classification task.
To further assess the performance of the QRC-enhanced model, the confusion matrix was computed on the test set (Fig. 5.3.3). The confusion matrix confirms that the QRC embeddings improve class separability, reducing mis-classification errors, even for classes with subtle differences in the input space.
In order to compare the performance of QRC-enhanced features with a purely classical approach, we evaluated the logistic regression model on PCA-reduced classical data, using the top hyperparameter combinations shown in Tab. 5.4. Figure 5.3.4 shows the learning curve for the classical dataset.

To provide a direct comparison between QRC-enhanced and classical features, Fig. 5.3.5 reports the test accuracy for both models across different training set sizes.

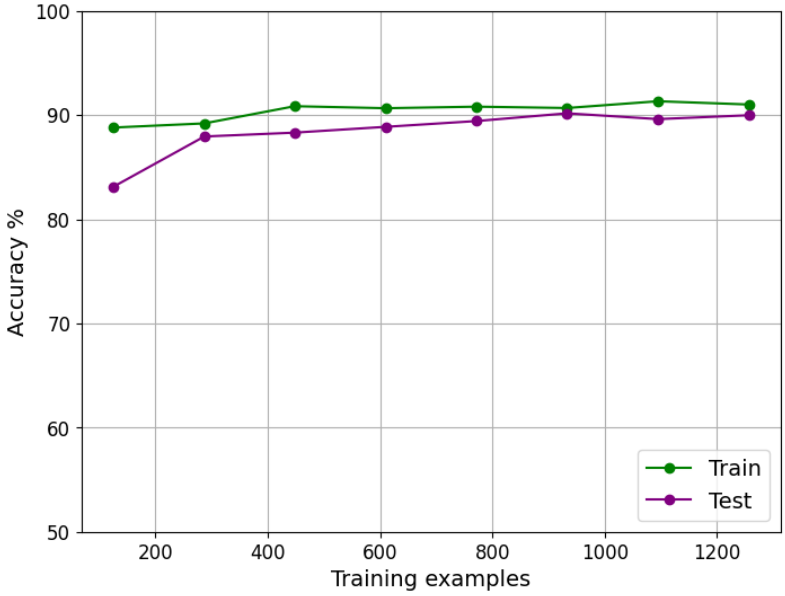

Figure 5.3.2: Learning curve for QRC-enhanced features. The model accuracy was evaluated on a fixed test set, corresponding to 30% of the 1797 samples, achieving a maximum accuracy of 90.17%.

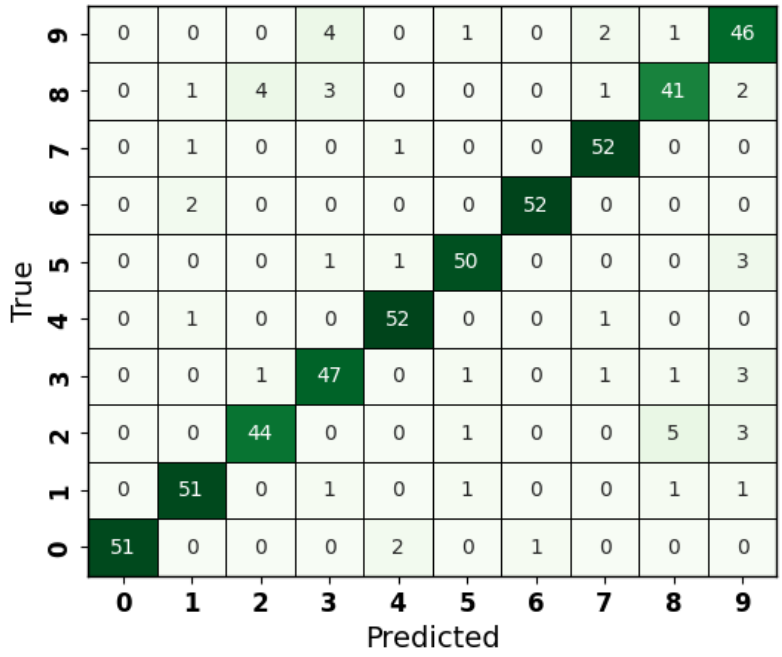

Figure 5.3.3: Confusion matrix for QRC-enhanced features. The matrix highlights the model's ability to correctly classify each class and identifies potential confusions between similar classes, such as 2/8 and 3/9.

From the comparison, it is evident that the QRC model consistently achieves higher accuracy than the classical model across all training sizes. This is possible thanks to the non linearity induced by the quantum reservoir step. Despite the fully classical model is able to reach great

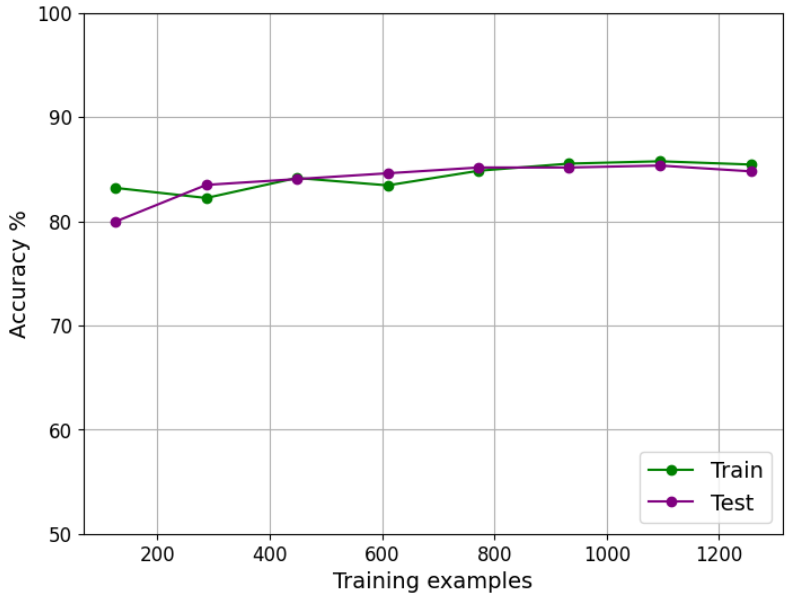

Figure 5.3.4: Learning curve for classical PCA-reduced features. The model accuracy was evaluated on a fixed test set, corresponding to 30% of the 1,797 samples, achieving a maximum accuracy of 85.15%.

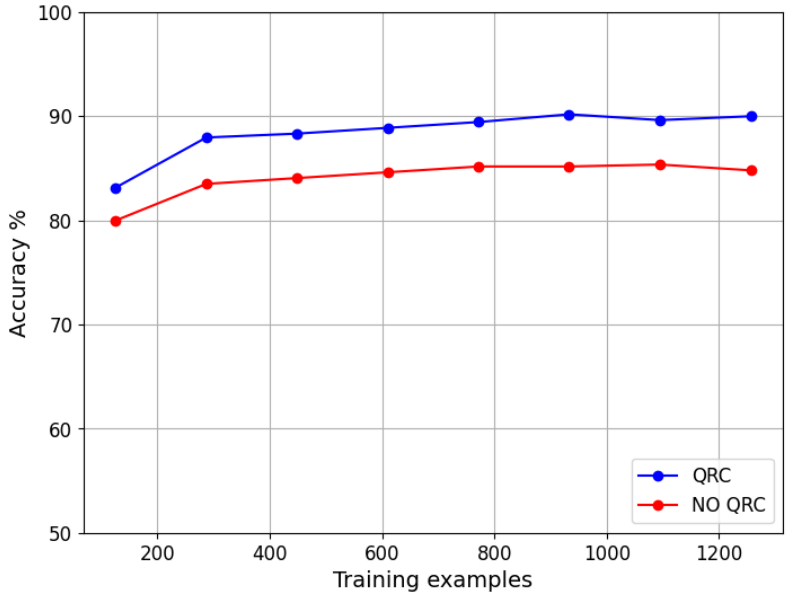

Figure 5.3.5: Comparison of learning curves for QRC-enhanced and classical features. The models were evaluated on the test set for varying training sizes with QRC consistently outperforming the classical approach.

performances over the test set ($\approx 85\%$), those are significantly improved by introducing a quantum step that takes into account the non linear correlation among the classical input feature. This analysis shows that, despite the symmetry-induced redundancy in the register, the global encoding scheme enables QRC embeddings that significantly outperform the purely classical baseline, confirming the role of the quantum reservoir in enhancing feature expressivity.

5.4 Local encoding

An alternative strategy for data injection into the quantum reservoir is the so-called *local encoding*, where each input feature x_i is directly mapped onto the detuning of an individual qubit:

$$\Delta_{q_j} = \lambda \Delta^{\max} x_j, \tag{5.4.1}$$

with Δ_{q_j} the detuning applied to qubit q_j, λ the encoding scale, and $\Delta^{\max}$ the maximum local detuning value.

Unlike the global encoding scheme (Sec. 5.3), this method requires the ability to independently address each atom in the register with a local detuning channel.
In the current generation of neutral-atom devices such as `FRESNEL`, this type of control is not physically available, since only global channels are supported and the detuning cannot be targeted at individual qubits. For this reason, the local encoding scheme cannot be directly implemented on the actual hardware. However, it can be emulated using the `MockDevice`, a virtual device without hardware constraints. To preserve physical relevance, we nevertheless constrained the parameters of the mock simulation to values consistent with the `FRESNEL` specifications, thus ensuring that the numerical study remains physically meaningful.
As showed in Fig. 5.2.4, the atomic register consists of 5 atoms placed on a one-dimensional chain with coordinates

$$\texttt{qubit\_coords} = \{q_0 = (-10, 0),\ q_1 = (-5, 0),\ q_2 = (0, 0),\ q_3 = (5, 0),\ q_4 = (10, 0)\},$$

matching the reduced dimensionality of the PCA input.
The main parameters are:

- Number of atoms: 5, equal to PCA feature vector dimension
- Encoding scale: $\lambda = 1$
- Blockade radius: $7\,\mu m$
- Maximum local detuning: $\Delta_l^{\max} \approx 48.7\,\mathrm{rad}/\mu s$, in accordance with `FRESNEL` specs
- Rabi frequency: $\Omega = 7.35\,\mathrm{rad}/\mu s$
- Time interval: $t_{int} = 100\,\mathrm{ns}$ for the probing of each feature
- Total evolution time: $T = 500\,ns$ per qubit

Each feature x_j of the input vector is mapped to a detuning value Δ_{q_j} acting only on qubit q_j. For each input $x = (x_1, \ldots, x_5)$, the simulation builds a sequence where all qubits are driven with constant-amplitude pulses at frequency Ω, while the detuning of each channel follows the locally assigned value:

$$\Omega(t) = \Omega, \qquad \Delta_{q_j}(t) = \lambda\, \Delta_l^{\max}\, x_j, \qquad \phi = 0.$$

As described in Sec. 5.2.3, the simulation is executed by inputting the desired pulse sequence, with the option to mimic the QPU. The observables used as input for the classical post-processing are single and two-qubit Pauli-Z operators. At each time step and for each feature vector, these

measurements are collected into the embeddings matrix, which constitutes the output of the QRC.

Once the embeddings matrix has been obtained from the QRC simulation, it is used as input for a logistic regression model to perform classification. As described in Sec. 5.2.4, the hyper-parameters of the logistic regressor are optimized via a grid search procedure. Importantly, this optimization is carried out separately for the QRC-enhanced embeddings and for the fully classical embeddings, in order to account for potential differences in feature scale and correlations between the two representations. Differently from the global encoding case, here no redundancy arises from symmetries in the register, since each qubit is assigned a distinct detuning profile. The resulting best hyperparameters for the QRC-enhanced and fully classical method are reported in Tables 5.4 and 5.7, respectively.

Penalty	C	Test Accuracy
l2	10	0.847
elasticnet	10	0.848
elasticnet	10	0.847
l1	100	0.846
l2	100	0.844

Table 5.7: Top 5 hyperparameter combinations for the logistic regression model using classical features, selected from the full grid search. Test accuracy is reported for each combination.

Model evaluation

After hyperparameter optimization, the QRC-enhanced features obtained with the local encoding scheme are fed into the best logistic regression model, an l_2-regularized regressor, as reported in Tab. 5.4. Figure 5.4.1 shows the test accuracy evaluated on the QRC-enhanced training and test sets as a function of the training set size.
As illustrated in the plot, the locally encoded QRC embeddings allow the model to reach high accuracy levels even with a relatively small training set, confirming that the quantum encoding provides an expressive feature representation even with a restricted training set. Compared to the global encoding case, the local detuning reaches similar performances across the different sets. This result is consistent with the observation in literature, see Fig. 4.3.2 especially in the emulation case where performances across the different encoding schemes are similar.

To further assess the performance of the locally encoded QRC model, the confusion matrix was computed on the test set (Fig. 5.4.2). The confusion matrix confirms that the local encoding provides a rich embedding space, with few overlap between classes.
To benchmark these results against a fully classical baseline, we again trained the logistic regression model on PCA-reduced classical features, using the top hyperparameter combinations shown in Tab. 5.7. Figure 5.4.3 shows the corresponding learning curve.

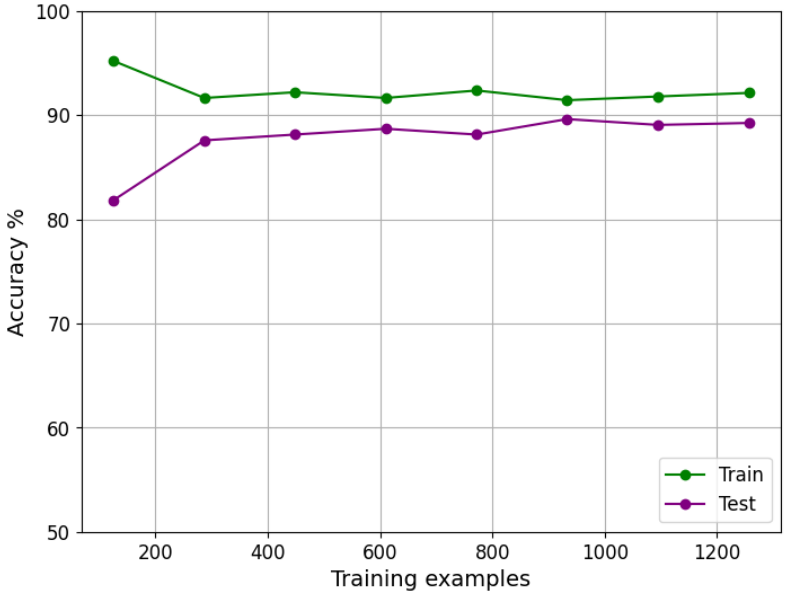

Figure 5.4.1: Learning curve for QRC-enhanced features with local encoding. The model accuracy was evaluated on a fixed test set, corresponding to 30% of the 1,797 samples, achieving a maximum accuracy of 89.61%.

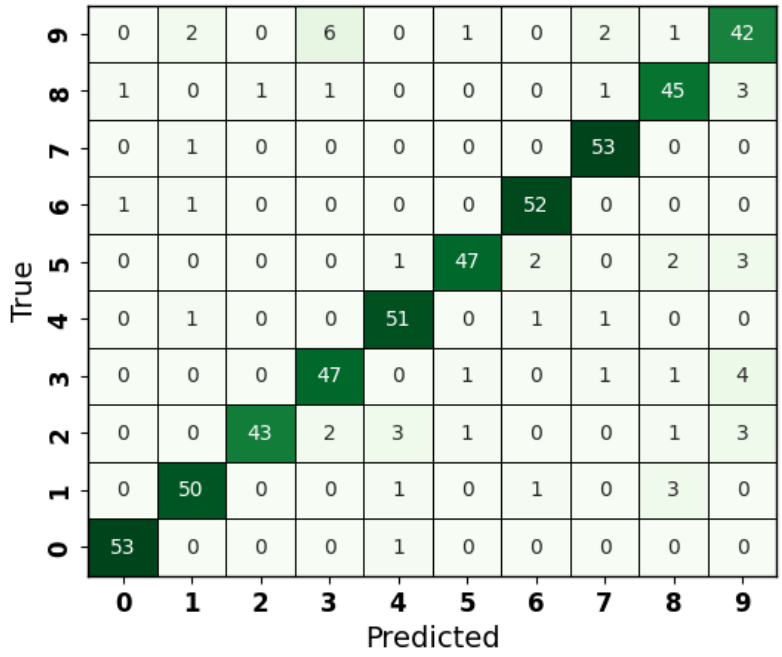

Figure 5.4.2: Confusion matrix for QRC-enhanced features with local encoding. The matrix highlight the model's accuracy reached in each classes, showing potential confusions between classes 9 and 3.

Finally, a direct comparison between QRC-enhanced embeddings with local encoding and the purely classical features is reported in Fig. 5.4.4. From the comparison, the local encoding scheme achieves performance improvements over classical features, with no redundancy issues arising from

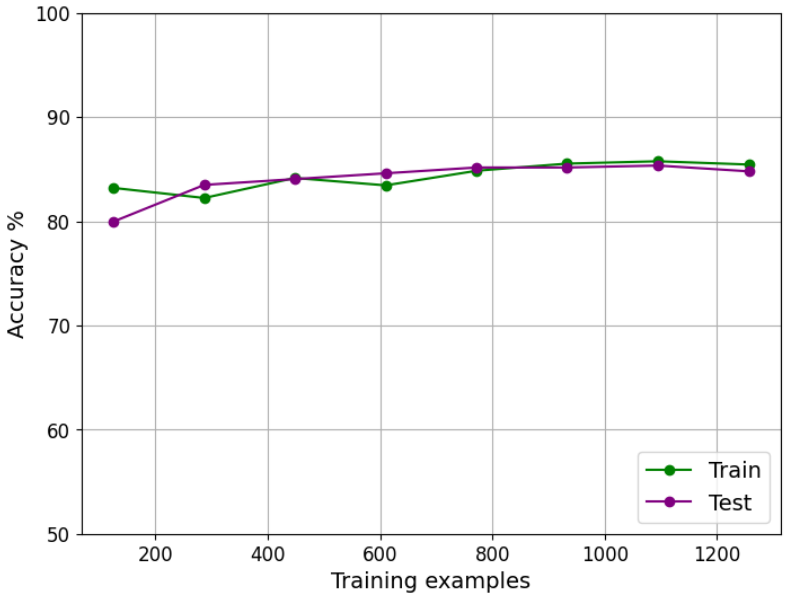

Figure 5.4.3: Learning curve for classical PCA-reduced features (local encoding baseline). The model accuracy was evaluated on a fixed test set, corresponding to 30% of the 1,797 samples, achieving a maximum accuracy of 85.15%.

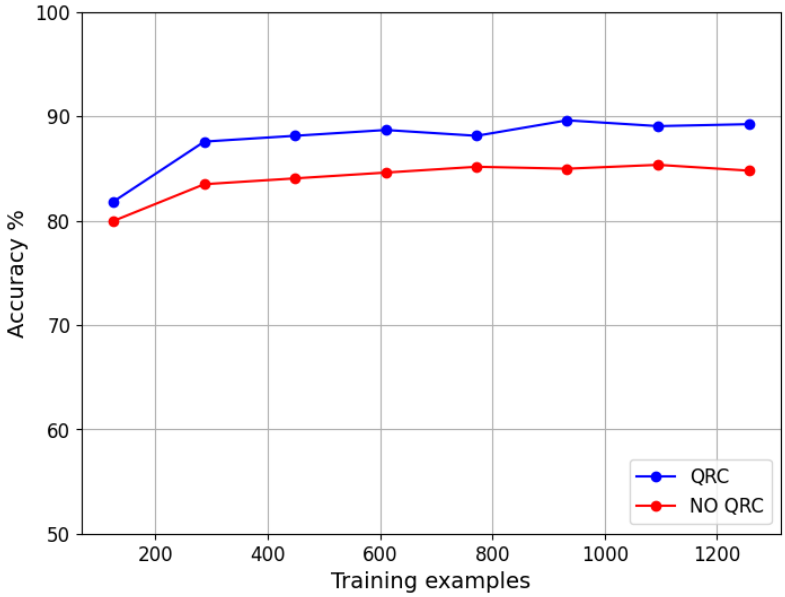

Figure 5.4.4: Comparison of learning curves for QRC-enhanced and classical features with local encoding. The QRC consistently achieves higher accuracy across training sizes.

register symmetry. This demonstrates that QRC consistently extracts non-linear correlations, regardless of the chosen encoding strategy.

5.5 Future works

The work presented in this thesis has focused on the simulation of the Quantum Reservoir Computing algorithm on neutral-atom devices, implementing both the global and the local encoding schemes, emulating the behavior of the `FRESNEL` quantum processor. A natural future development consists in extending the described procedure to direct execution on a real QPU, in order to experimentally validate the results obtained in simulation.
Since the current `FRESNEL` hardware does not yet support local qubit addressing, such an extension is currently feasible only in the global encoding mode.

The procedure for running the QRC protocol on real hardware remains conceptually identical to the simulated one: for each input x, a sequence of global detuning pulses is constructed according to the linear register configuration already described in this work. Measurements are then performed on single and two-qubit Pauli-Z operators. The key difference is that embeddings are no longer computed from Schrödinger-based simulations (e.g., QuTiP), but directly from bit-strings obtained through quantum measurements on the device.
To enable execution on a QPU, the sequences are serialized and submitted to the Pasqal SDK. For each evolution time t, a batch is created, containing as many jobs as there are input instances. Each job specifies the detuning parameters for the sequence together with the number of repetitions N, which controls the statistical accuracy of the measurement.
Once the jobs have been processed by the QPU, each of them returns a distribution of bit-strings in the computational basis. These bit-strings are mapped to ± 1 values, from which the expectation values $\langle Z_i \rangle$ and correlations $\langle Z_i Z_j \rangle$ are derived as averages over the available repetitions.
By repeating the measurement at each time step, it is possible to estimate embeddings with controlled accuracy, at the cost of increased computational resources.
The overall software workflow can thus be summarized as follows:

1. Construction and serialization of the global detuning pulse sequences corresponding to the inputs;
2. Submission of batches of jobs to the QPU via the Pasqal SDK;
3. Collection of bit-strings returned for each input and evolution time;
4. Conversion into expectation values $\langle Z \rangle$ and $\langle ZZ \rangle$ through averaging over repeated measurements;
5. Assembly of the embedding matrix, concatenating the features obtained at different time steps, to be used as input to the machine learning model.

In this way, the simulated framework can be seamlessly extended to execution on real quantum hardware, with the embedding quality explicitly depending on the number of shots and on the intrinsic noise of the device.

Beyond experimental validation, several research directions naturally emerge from the results of this thesis:

- **Dependence on observables.** A systematic analysis may clarify to what extent measurements of $\langle Z \rangle$ and $\langle ZZ \rangle$ are sufficient to saturate performance, or whether the inclusion of higher-order correlators provides a significant contribution to the expressivity of the reservoir.

- **Thermalization phenomena.** Since embeddings are collected as a function of time, it is important to investigate whether expectation values converge to a stationary regime after a certain evolution time, thereby reducing the effective temporal window available for computation. This aspect is crucial for optimizing hardware resources and for the design of more efficient reservoirs.

- **Improvement of global encoding.** The main limitation observed in global encoding is redundancy arising from the symmetry of the atomic register. A possible improvement would consist in introducing controlled asymmetries in the spatial arrangement of the atoms, in order to break degeneracies and enrich the embedding space. This direction represents a promising development both at the simulation level and in hardware design.

- **Extensions to complex tasks.** A further step will be to apply the QRC framework to more complex problems, such as recognition of excited states in quantum chemistry or multiclass classification tasks in real-world domains, in order to assess to what extent the advantages observed in this thesis can be generalized beyond the reduced MNIST dataset.

In conclusion, the transition from simulation to experimentation on real hardware, together with the exploration of richer observables, dynamics, and configurations, constitute the main future research directions, with the ultimate goal of consolidating QRC as a promising paradigm for quantum data processing.

6 Conclusions

This thesis investigated Quantum Reservoir Computing (QRC) as a distinct, non-variational paradigm for quantum machine learning, suited for near-term quantum hardware. Motivated by the limitations of current NISQ devices (noise, limited qubit connectivity and trainability issues such as barren plateaus), the work developed, implemented and benchmarked two encoding strategies, global and local detuning encodings, within a QRC pipeline simulated using Pasqal's Pulser environment.

The main technical contributions are threefold. First, a QRC pipeline for Pasqal devices was designed and implemented to map classical inputs into time-dependent pulses and to collect single- and two-qubit Pauli-Z expectation values as embeddings; the implementation adheres to realistic device constraints (5 atoms, blockade radius 7 μm, $\Omega = 7.35$ rad/μs, $\Delta_g^{\max} \approx 48.7$ rad/μs) and employs Pulser primitives to construct physically plausible sequences. Second, two concrete encoding strategies were compared: a global detuning waveform affecting all atoms simultaneously, and a local per-qubit detuning scheme. Finally, a systematic evaluation against a classical PCA baseline was carried out using logistic regression with grid-searched hyper-parameters, reporting learning curves, confusion matrices and model-selection statistics.

The empirical findings show that QRC-enhanced embeddings consistently outperform the classical PCA baseline across encoding schemes and training-set sizes. Quantitatively, the global-encoding QRC achieved a peak test accuracy of 90.17% (vs. 85.15% for the classical baseline), while the local encoding delivered 89.61%. The improvement is observed throughout the learning curves and is particularly pronounced in low-data regimes: QRC embeddings provide richer, more separable feature representations that allow simple linear readouts to reach higher accuracy with fewer training examples. Confusion matrices indicate improved class separability and reduced mis-classification for closely related classes.
These results support two key conclusions. First, the quantum reservoir step effectively induces nonlinear embeddings that are difficult to reproduce with simple classical PCA pre-processing; this enriches the feature space in a way that benefits linear classifiers. Second, the advantages of QRC are not limited to a single encoding strategy: both global and local encodings produce meaningful gains, although they come with different practical constraints.

The work also identifies clear limitations and practical challenges.
The global encoding suffers from symmetry-induced redundancy when the atomic register is arranged symmetrically: symmetric atoms produce nearly identical embeddings and reduce effective expressivity. The local encoding avoids this redundancy but could only be studied here in emulation because the current `FRESNEL` hardware does not support per-qubit detuning. More generally, all results presented are simulated: embedding quality on a real QPU will depend on finite-shot

statistics, device noise, and compilation overhead.

As immediate next steps, we prioritize experimental validation of the global-encoding sequences on the Pasqal QPU and targeted strategies to mitigate symmetry-induced redundancy in the reservoir layout. Additionally, a focused study on the choice of observables and on optimal evolution times would clarify the trade-off between expressivity and measurement cost. For a detailed road-map and further technical directions see Sec.5.5

In summary, this thesis demonstrates that QRC, implemented with realistic neutral-atom primitives, is a promising and experimentally grounded route to harness near-term quantum dynamics for supervised learning. The observed accuracy gains over classical baselines, together with the non-variational, training-free nature of the reservoir step, indicate that QRC can offer practical benefits on NISQ hardware. The transition to real-device experiments and the systematic study of observables, asymmetries and scaling will be decisive to confirm and extend the advantages demonstrated here. Overall, QRC appears as a robust and flexible framework worth further experimental investment, well positioned to exploit the dynamical richness of neutral-atom platforms in the near term.

Bibliography

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982.

[2] Matthias Klusch, Jörg Lässig, Daniel Müssig, Antonio Macaluso, and Frank K. Wilhelm. Quantum artificial intelligence: A brief survey. *KI - Künstliche Intelligenz*, 38(3):257–276, 2024.

[3] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91:045002, Dec 2019.

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

[5] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Quantum Science and Technology. Springer, 2018.

[6] Daniel F. Perez-Ramirez. Variational quantum algorithms for combinatorial optimization. arXiv:2407.06421 [quant-ph], `https://arxiv.org/abs/2407.06421`, 2025. arXiv preprint.

[7] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.

[8] I. M. Georgescu, S. Ashhab, and Franco Nori. Quantum simulation. *Rev. Mod. Phys.*, 86:153–185, Mar 2014.

[9] David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9–11):771–783, 2000.

[10] J. Ignacio Cirac and Peter Zoller. Goals and opportunities in quantum simulation. *Nature Physics*, 8:264–266, 2012.

[11] Nicholas S. DiBrita, Daniel Leeds, Yuqian Huo, Jason Ludmir, and Tirthak Patel. Recon: Reconfiguring analog rydberg atom quantum computers for quantum generative adversarial networks. arXiv:2408.13389 [quant-ph], 2024. To appear in the Proceedings of the International Conference on Computer-Aided Design (ICCAD), 2024.

[12] R. Trivedi, A. Franco Rubio, and J. I. Cirac. Quantum advantage and stability to errors in analogue quantum simulators. *Nature Communications*, 15:6507, 2024.

[13] R. Shaffer, E. Megidish, J. Broz, et al. Practical verification protocols for analog quantum simulators. *npj Quantum Information*, 7:46, 2021.

[14] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin. Fast quantum gates for neutral atoms. *Phys. Rev. Lett.*, 85:2208–2211, Sep 2000.

[15] Manuel Endres, Hannes Bernien, Alexander Keesling, Harry Levine, Eric R. Anschuetz, Alexandre Krajenbrink, Crystal Senko, Vladan Vuletić, Markus Greiner, and Mikhail D. Lukin. Atom-by-atom assembly of defect-free one-dimensional cold atom arrays. *Science*, 354(6315):1024–1027, 2016.

[16] Hannes Bernien, Sylvain Schwartz, Alexander Keesling, Harry Levine, Ahmed Omran, Hannes Pichler, Soonwon Choi, Alexander S. Zibrov, Manuel Endres, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551:579–584, 2017.

[17] M. Morgado and S. Whitlock. Quantum simulation and computing with rydberg-interacting qubits. *AVS Quantum Science*, 3(2):023501, 05 2021.

[18] Daniel Barredo, Vincent Lienhard, Sylvain de Léséleuc, Thierry Lahaye, and Antoine Browaeys. Synthetic three-dimensional atomic structures assembled atom by atom. *Nature*, 561:79–82, 2018.

[19] Jonathan Wurtz et al. Aquila: QuEra's 256-qubit neutral-atom quantum computer. 6 2023.

[20] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020.

[21] Pere Mujal, Rodrigo Martínez-Peña, Johannes Nokkala, Jorge García-Beni, Gian Luca Giorgi, Miguel C. Soriano, and Roberta Zambrini. Opportunities in quantum reservoir computing and extreme learning machines. *Advanced Quantum Technologies*, 4(8):2100027, 2021.

[22] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, Bonn, Germany, 2001.

[23] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[24] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[25] David Verstraeten, Benjamin Schrauwen, Michiel D'Haene, and Dirk Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.

[26] Martín Larocca, Siripong Thanasilp, Shao-Hua Wang, et al. Barren plateaus in variational quantum computing. *Nature Reviews Physics*, 7:174–189, 2025.

[27] Sanjib Ghosh, Andrzej Opala, Michał Matuszewski, Tomasz Paterek, and Timothy C. H. Liew. Reconstructing quantum states with quantum reservoir networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):3148–3155, 2021.

[28] Hiroki Kawai and Yuya O. Nakagawa. Predicting excited states from ground state wavefunction by supervised quantum machine learning. *Machine Learning: Science and Technology*, 1(4):045027, 2020.

[29] L. Domingo, G. Carlo, and F. Borondo. Optimal quantum reservoir computing for the noisy intermediate-scale quantum era. *Physical Review E*, 106(L043301):L043301, 2022.

[30] Laia Domingo, M. Chehimi, S. Banerjee, S. He Yuxun, S. Konakanchi, L. Ogunfowora, S. Roy, S. Selvaras, M. Djukic, and C. Johnson. A hybrid quantum-classical fusion neural network to improve protein-ligand binding affinity predictions for drug discovery, 2023. arXiv:2309.03919 [quant-ph].

[31] Laia Domingo. *Classical and quantum reservoir computing: development and applications in machine learning*. Doctoral thesis, arXiv, 2023. arXiv:2310.07455 [quant-ph].

[32] Alberto Di Meglio, Karl Jansen, Ivano Tavernelli, Constantia Alexandrou, Srinivasan Arunachalam, Christian W. Bauer, Kerstin Borras, Stefano Carrazza, Arianna Crippa, Daniel J. Egger, Vincent Croft, Roland de Putter, Andrea Delgado, Vedran Dunjko, Elias Fernández-Combarro, Daniel González-Cuadra, Stefan Kühn, Michele Grossi, Denis Lacroix, Elina Fuchs, Lena Funcke, Jad C. Halimeh, Randy Lewis, Zoë Holmes, Donatella Lucchesi, Miriam Lucio Martinez, Federico Meloni, Antonio Mezzacapo, Simone Montangero, Vincent R. Pascuzzi, Voica Radescu, Enrique Rico Ortega, Lento Nagano, Alessandro Roggero, Julian Schuhmacher, Joao Seixas, Pietro Silvi, Francesco Tacchino, Panagiotis Spentzouris, Cenk Tüysüz, Kristan Temme, Koji Terashi, Jordi Tura, Sofia Vallecorsa, Uwe-Jens Wiese, Shinjae Yoo, and Jinglei Zhang. Quantum computing for high-energy physics: State of the art and challenges. *PRX Quantum*, 5:037001, 2024.

[33] Milan Kornjača, Hong-Ye Hu, Chen Zhao, Jonathan Wurtz, Phillip Weinberg, Majd Hamdan, Andrii Zhdanov, Sergio H. Cantu, Hengyun Zhou, Rodrigo Araiza Bravo, Kevin Bagnall, James I. Basham, Joseph Campo, Adam Choukri, Robert DeAngelo, Paige Frederick, David Haines, Julian Hammett, Ning Hsu, Ming-Guang Hu, Florian Huber, Paul Niklas Jepsen, Ningyuan Jia, Thomas Karolyshyn, Minho Kwon, John Long, Jonathan Lopatin, Alexander Lukin, Tommaso Macrì, Ognjen Marković, Luis A. Martínez-Martínez, Xianmei Meng, Evgeny Ostroumov, David Paquette, John Robinson, Pedro Sales Rodriguez, Anshuman Singh, Nandan Sinha, Henry Thoreen, Noel Wan, Daniel Waxman-Lenz, Tak Wong, Kai-Hsin Wu, Pedro L. S. Lopes, Yuval Boger, Nathan Gemelke, Takuya Kitagawa, Alexander Keesling, Xun Gao, Alexei Bylinskii, Susanne F. Yelin, Fangli Liu, and Sheng-Tao Wang. Large-scale quantum reservoir learning with an analog quantum computer, 2024. arXiv:2407.02553 [quant-ph].

[34] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, and Jarrod R. McClean. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 2022.

[35] Henrique Silvério, Sebastián Grijalva, Constantin Dalyac, Lucas Leclerc, Peter J. Karalekas, Nathan Shammah, Mourad Beji, Louis-Paul Henry, and Loïc Henriet. Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays. *Quantum*, 6:629, January 2022.

[36] J.R. Johansson, P.D. Nation, and Franco Nori. Qutip: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8):1760–1772, 2012.

[37] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, Nov 2012.

[38] Scikit-learn developers. load_digits — scikit-learn documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html`, 2025. Accessed: 2025-09-02.

[39] IBM. Logistic regression. `https://www.ibm.com/think/topics/logistic-regression`. Accessed: 25-Aug-2025.

[40] Scikit-learn Developers. sklearn.metrics.confusion_matrix. `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`. Accessed: 20-Aug-2025.

List of Figures

List of Tables