Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (38^{th}cycle)

Quantum Machine Learning Applications and Algorithms

By

Marco Russo

******

Supervisor(s):
Prof. Bartolomeo Montrucchio, Supervisor

Doctoral Examination Committee:
Prof. Fiaz Gul Khan, Referee, COMSATS University Islamabad
Prof. Paolo Rech, Referee, Università di Trento
Dr. Riccardo Mengoni, Welinq

Politecnico di Torino
2026

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Marco Russo
2026

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

La scrittura scientifica è notoriamente fredda, austera, priva di emozioni. Qualsiasi risultato ottenuto da un esperimento, sia esso inconcludente o grandioso, viene presentato con la solita formula "in questa figura e in quella tabella si può vedere che", immancabilmente seguito da un "possiamo quindi dedurre che". Poi un abisso, oltre il quale c'è un altro mondo, quello dell'espressione dell'anima, dei nostri moti interiori e della consapevolezza di essi. Abbiamo deciso appunto di metterci un abisso in mezzo, e in fondo non è che si possa pretendere altrimenti; la scienza è questione di fatti e non di opinioni. Ma dentro colui che scrive con freddezza, nel momento in cui lo fa, il suo sentire non si spegne. Meglio allora non smettere mai di avere la testa nel secondo mondo anche se ci tocca lavorare nel primo.

Il sentire di chi scrive si nasconde tra ogni riga. E se dal punto di vista dell'utile conta quello che nelle prossime pagine sarà nero su bianco, quel che importa a me sono in realtà le righe invisibili. Ognuna di queste racchiude chi sono mentre scrivo, e le persone e le cose che in qualche modo, in fasi diverse di questo percorso, hanno avuto un impatto di cui magari non si rendono conto. È per loro che quelle righe bianche sono scritte.

Per mio fratello, in cui vedo molta più forza di quanta ne veda lui in sé stesso, e che, anche se ogni tanto la marea glielo nasconde e lui se lo dimentica, è così in gamba che alla fine si guarderà indietro ridendo.

Per mia madre, che non ha mai potuto fare a meno di essere preoccupata per qualsiasi cosa, che ha sempre saputo ascoltarmi in ogni momento mettendo da parte ogni impegno, e che quando sono andato in posti incredibili sembrava vivere quelle cose attraverso i miei occhi. Una vita passata a sentire più di noi le nostre stesse emozioni, sognando con noi.

Per mio padre, che ogni giorno dopo tante ore di lavoro logorante torna da mia madre con il sorriso chiedendole di noi e di lei, e che questo Natale ha passato tutto il tempo con me in garage per lavorare sulla macchina.

Per Davide, Gianluca, Leonardo, Marco, Salvatore, i pochi ma buoni e in ordine alfabetico per non offendere nessuno, che in questi anni a volte senza bussola sono stati compagni del mio viaggio, ognuno nel suo insostituibile modo.

Per Torino, la scenografia di un terzo della mia vita che ho amato sin dall'inizio per il suo essere così bella, calorosa anche nella morsa dei suoi inverni, e che sembrava sempre parlarmi.

Per te, Giulia, la mia bella. Anche se sei apparsa quasi alla fine, ci sei sempre stata, perché ti ho sempre cercata senza saperlo. Possiedi davvero tutta la dolcezza e la bellezza disarmante di cui ti accuso, me le dai ogni giorno, e sei sempre così orgogliosa di me, così attenta a me, a ogni lato del mio carattere e a ogni cosa io possa sentire, con quella capacità di vedermi così bene anche quando non parlo, da farmi dimenticare di qualsiasi tumulto mentre mi fai sentire a casa e mi dai una felicità che non mi sarei mai sognato. Sei la mia musa, piena di arte come l'arte che tanto ami, e di talento e di qualità incredibili. Avevo iniziato il dottorato per scoprire cose nuove, alla fine la scoperta più bella sei stata tu.

Acknowledgements

I would like to thank Bartolomeo Montrucchio for the opportunity to carry out this PhD, and for the resources and the suggestions provided during my time in the department. I am also grateful to Gabriel Perdue for his encouraging supervision at Fermilab and for the opportunity to conduct research in such a prestigious environment; to Giuseppe Di Guglielmo and Andy Li for their technical support during those months; and to Simone Donati, Emanuela Barzi, Giorgio Bellettini and Marco Mambelli for their contribution in making that opportunity possible.

Abstract

This dissertation investigates hardware-aware Quantum Machine Learning under Noisy Intermediate-Scale Quantum constraints, addressing the gap between idealized algorithm design and physical hardware limitations, including decoherence, gate infidelity, and restricted connectivity. The central claim is that practical progress depends on co-design between high-level models and low-level control.

First, the dissertation establishes the theoretical foundations of encoding, kernels, and trainability, then applies them to a neutral-atom Quantum Kernel Estimation pipeline. On the benchmark used, the proposed quantum kernel reaches a mean accuracy of $79.8\% \pm 8.8\%$ across 30 independent dataset seeds, significantly outperforming the classical radial-basis-function baseline ($52.8\% \pm 10.2\%$) with a mean accuracy gap of $27.0\% \pm 13.4\%$ (paired t-test p-value of 6.33×10^{-12}), while highlighting the role of geometric connectivity, pulse-level gate synthesis, and hardware clock quantization.

Second, for superconducting qubit control, the dissertation introduces Adapted Randomized Benchmarking (ARB) as a statistically rigorous method for both estimating gate-set fidelity and serving as a loss function. A compact neural network is first pre-trained with a standard MSE loss on simulation data to produce high-fidelity single-qubit pulse coefficients. The physical parameters of the simulated system are then deliberately altered, emulating realistic drifts that occur after deployment, such as changes in anharmonicity. Rather than retraining from scratch, the network is fine-tuned on the shifted parameters using the ARB fidelity estimate as the loss function. The fine-tuning procedure restores the fidelity in a statistically rigorous manner, demonstrating that lightweight on-device recalibration is feasible and confirming the viability of embedded real-time pulse calibration on resource-constrained platforms such as FPGAs.

Third, in the photonic domain, a gate-based quantum circuit is shown to reproduce with high fidelity the dynamics of a single-photon protective-measurement experiment. The simulation faithfully captures the Quantum-Zeno-inspired survival-probability enhancement observed in the laboratory, with the protective configuration reducing the percentage relative error by approximately 83% with respect to the non-protective one despite increased circuit depth. This validates the gate-model decomposition of the spatial translation and provides a tool for designing protocol variants or porting the scheme to platforms lacking native continuous-variable operations.

Overall, the results support a unified roadmap in which platform-specific control, model design, and measurement strategy jointly improve robustness and prepare the ground for future large-scale QML validation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The transition from classical to quantum computational paradigms represents one of the most significant shifts in the history of information theory. At the heart of this evolution lies the promise of exploiting the exponential Hilbert space dimensionality given by qubits to solve problems previously deemed intractable. Among the various fields that are most likely to have a revolutionary impact, QML (Quantum Machine Learning)stands out as a particularly fertile ground for innovation. By integrating the inductive biases of classical learning with the unique properties of quantum mechanics, such as superposition, entanglement, and interference, researchers aim to redefine the boundaries of data processing and pattern recognition.

QML is particularly relevant in the current phase of computational science because it addresses a dual pressure: on one side, the increasing complexity of data-driven models and optimization tasks; on the other, the slowing hardware scaling of conventional architectures. In this context, QML should not be interpreted as an immediate replacement for classical machine learning, but as a strategic research direction to identify which model classes may benefit from quantum-native representations and which hardware-software co-design choices are required to realize practical gains.

However, the current landscape of quantum computing is defined by the constraints of the Noisy Intermediate-Scale Quantum (NISQ) era. As discussed in Chapter 2, while the theoretical potential for quantum advantage is well-documented, practical hardware implementation remains vulnerable to environmental decoherence, high gate error rates, and restricted qubit connectivity. These limitations necessi-

tate a departure from purely abstract algorithmic design toward a hardware-aware methodology. This dissertation argues that a demonstrable quantum advantage is not merely a product of increasing qubit counts, but rather the result of a synergistic integration between high-level algorithms and low-level hardware control.

The structure of this dissertation follows a logical progression from theoretical foundations to platform-specific optimizations. After the comprehensive literature review provided in Chapter 2, the dissertation enters its technical core. In Chapter 3, we investigate the implementation of Quantum Support Vector Machines (QSVM) on neutral atom arrays. This platform is selected because its geometric flexibility directly targets one of the dominant QML bottlenecks, namely entangling connectivity overhead. By exploiting pulse-level synthesis and geometric optimization, we show how moving beyond the standard circuit abstraction can improve classification performance even on small-scale systems.

The focus then shifts toward the control layer in Chapter 4, where we address the critical challenge of real-time calibration for superconducting qubits. This platform is selected because superconducting devices currently offer a mature ecosystem for fast gate operations but remain strongly constrained by calibration overhead and drift. Developed during a research period at Fermilab, this chapter introduces an embedded machine learning framework designed to automate the optimization of control pulses. By deploying neural networks directly on FPGA-based controllers, we minimize the latency between measurement and correction, providing a scalable solution for maintaining high-fidelity operations in complex cryogenic environments.

Furthermore, Chapter 5 explores the fundamental dynamics of quantum states through a high-fidelity simulation of a single-photon laboratory experiment. Although this study diverges from explicit machine learning applications, it provides essential insights into state preservation and measurement paradigms. Specifically, we study a protective-measurement regime related to the Quantum Zeno Effect and show a finite-step survival-probability advantage that is useful in practice.

Taken together, these chapters are intentionally complementary: Chapter 3 addresses representational expressivity under connectivity constraints, Chapter 4 addresses control fidelity and deployment latency, and Chapter 5 addresses measurement robustness and state preservation. This organization clarifies the methodological choices of the dissertation and frames each contribution as a targeted step toward practical, hardware-grounded QML.

To ensure the dissertation remains a self-contained and rigorous reference, three appendices are included. Appendix A provides the mathematical and logical foundations necessary to follow the derivations presented in Chapter 5. Appendix B offers a technical overview of the diverse quantum computing technologies that support the experimental work, ensuring the reader possesses a comprehensive understanding of the physical constraints discussed. Finally, Appendix C provides an overview of Neuro-Symbolic Artificial Intelligence. Although this appendix represents a side research activity conducted during the doctoral program, its inclusion is justified by the strategic importance of integrating formal logic with statistical learning. As QML models often struggle with interpretability and "black-box" behaviors, the methodologies explored in this side research, such as Logic Tensor Networks, offer a theoretical foundation for future hybrid architectures where quantum advantage could be combined with human-readable logical constraints.

1.1 Research Questions and Contributions

To make the scope of this dissertation explicit, this section formalizes the main research questions, the working hypotheses, and the concrete contributions developed across the chapters.

1.1.1 Research Questions

- How can hardware-aware QML co-design reduce the gap between algorithmic proposals and the practical constraints of NISQ devices?
- To what extent can neutral-atom geometric flexibility improve QML kernel methods under entanglement and connectivity constraints?
- Can embedded ML-based control achieve high-fidelity gate synthesis while reducing calibration latency in superconducting platforms?
- Can protective-measurement strategies provide a practically useful state-preservation advantage in finite-step regimes relevant to QML workflows?

1.1.2 Working Hypotheses

- Hardware-aware design choices at pulse, layout, and control levels can improve practical robustness even when qubit counts remain limited.
- Neutral-atom layouts can reduce connectivity overhead and preserve quantum-kernel expressivity more effectively than constrained fixed-topology mappings.
- Data-driven quantum control, obtained considering qubit physical properties during training and deployed on high-speed electronics, can be used to optimize gate synthesis achieving high-fidelity quantum operations while mitigating the control bottleneck in superconducting systems.
- In finite operating regimes, QZE-related protective strategies can improve survival behavior and reduce error indicators.

1.1.3 Thesis Contributions

This dissertation offers four main contributions. First, it provides a unified hardware-aware QML perspective that connects algorithm design, control engineering, and measurement strategy under realistic NISQ constraints. Second, it introduces a neutral-atom QKE workflow based on pulse-level gate synthesis and geometric optimization, validated on supervised classification benchmarks, along with algorithms for deriving quantum gates for neutral atoms directly from pulse-level control. Third, it presents an embedded ML framework for superconducting quantum control, including Adapted Randomized Benchmarking for hardware-oriented adaptation. Finally, it delivers a high-fidelity photonic simulation study clarifying finite-step protective behavior and its relevance for robust state preparation and readout.

These elements define the narrative thread of the dissertation: Chapter 2 provides the conceptual basis, Chapters 3 and 4 test the hypotheses on complementary hardware axes, and Chapter 5 examines measurement robustness before the final synthesis in Chapter 6.

Chapter 2

From Classical to Quantum Machine Learning

2.1 Introduction

The emergence of QML as an interdisciplinary field represents a strategic response to the escalating computational demands of modern Artificial Intelligence (AI) and the approaching physical limitations of classical semiconductor hardware. This chapter provides a systematic analysis of the QML landscape, serving as the theoretical foundation for the specialized hardware investigations presented in the subsequent parts of this dissertation. The content of this chapter has been previously published by the author in [5].

We begin by establishing the fundamental quantum mechanical principles, such as qubits, superposition, and entanglement, essential for the encoding of classical data into quantum states. The chapter then examines the structural and efficiency differences between classical algorithms and their quantum-enhanced counterparts, covering both discriminative models (e.g., Quantum Support Vector Machines and Quantum Neural Networks) and generative or clustering approaches (e.g., Quantum K-Means and qCLUE). Furthermore, we provide a critical assessment of the challenges inherent to the Noisy Intermediate-Scale Quantum (NISQ) era, including data I/O constraints and the phenomenon of barren plateaus. By mapping the current state-of-the-art, this chapter delineates the path from hybrid quantum-classical models toward the long-term goal of fault-tolerant quantum advantage, and provides

the necessary context for the hardware-specific optimizations for Neutral Atoms and Superconducting qubits discussed in Chapters 3 and 4.

2.2 The Convergence of Quantum Computing and Machine Learning

The rapid advancements in machine learning (ML) have profoundly transformed numerous sectors, from computer vision to natural language processing. However, this progress has come with an escalating demand for computational resources. Modern ML models, particularly large language models (LLMs) with hundreds of billions of parameters, are extraordinarily compute-intensive, requiring massive investments in time and hardware that only a few organizations can sustain.

Traditional computing, which has historically driven exponential growth in transistor density as predicted by Moore's Law, is now confronting fundamental physical limits. This saturation restricts further increases in computational power, leading to bottlenecks in data processing, algorithmic development, and scientific discovery [6]. In response to these demands, Quantum Computing (QC) has emerged as a promising new paradigm. QC exploits the principles of quantum mechanics to process information in ways that are fundamentally unattainable for classical systems. QML represents the convergence of these two transformative fields, aiming to leverage quantum principles to enhance the efficiency and capabilities of ML algorithms [7]. This context positions QML not merely as an academic pursuit, but as a necessary evolutionary step to sustain the progress of AI, especially for tasks involving high-dimensional data.

2.2.1 Motivation: Addressing Classical Limitations

The core motivation behind QML is to overcome the inherent constraints of classical hardware and algorithms, which process information sequentially and often struggle with the scale of high-dimensional datasets. QML aims to boost efficiency by harnessing unique quantum properties:

- **Superposition and Entanglement**: Unlike classical bits limited to binary states (0 or 1), qubits can exist in multiple states simultaneously and can be

interconnected through entanglement, regardless of physical separation. This allows for an inherent parallelism that enables the processing of vast amounts of data concurrently.

- **Quantum Advantage**: This concept refers to the potential for quantum systems to solve problems significantly faster than advanced classical supercomputers. We distinguish between *theoretical quantum advantage*, demonstrating superpolynomial speedup for an algorithm regardless of current hardware availability, and *practical quantum advantage*, which requires both the algorithm and the physical hardware to achieve a measurable gain on specific problems.

Beyond mere speed, quantum systems are inherently well-suited for identifying patterns within high-dimensional data that overwhelm classical algorithms. They can solve problems in fewer computational steps, sample probability distributions naturally for generative tasks, and compress large datasets into a relatively small number of qubits [8]. The ability of quantum algorithms to exploit constructive and destructive interference allows for the amplification of correct solutions while suppressing incorrect ones, addressing problems that are currently computationally impractical for classical systems.

2.3 Foundational concepts of Quantum Computing for Machine Learning

2.3.1 Quantum bits: superposition and entanglement

At the heart of quantum computing lies the quantum bit, or qubit, which fundamentally differs from the classical bit. While a classical bit can only exist in one of two definite states, i.e. either 0 or 1, a qubit leverages the principles of quantum mechanics to exist in a superposition of both 0 and 1 simultaneously. In particular, the state $|\psi\rangle$ of a single qubit is represented as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.1}$$

where $\alpha, \beta \in \mathbb{C}$, $s.t. |\alpha|^2 + |\beta|^2 = 1$ are called *amplitudes* and their squared moduli are called *magnitudes*. The magnitudes are real numbers representing the probability

of measuring either of the two bases, and they sum up to 1. The previous equation follows the *bra−ket* notation and $|\psi\rangle$, called *ket psi*, is a complex linear combination of the two basis states, $|0\rangle$ and $|1\rangle$, that are orthogonal vectors spanning a complex 2-dimensional Hilbert space. The physical meaning of these two bases depends on the physical realization of the qubit. For instance, in the case of photonic qubits, they represent the horizontal and vertical polarization of the photon. Since $|\psi\rangle \in \mathbb{C}^2$ belongs to a vector space, it can be represented by means of a column vector, i.e.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \implies |\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.2}$$

A *bra*, instead, is simply given by $\langle\psi| = |\psi\rangle^\dagger$, which is the Hermitian conjugate of $|\psi\rangle$, i.e. a row vector with conjugated entries, so that $\langle\psi| = \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix}$, and it is evident that

$$\langle\psi|\psi\rangle = 1, \ \langle 0|1\rangle = \langle 1|0\rangle = 0, \tag{2.3}$$

where the first equation represents the unitarity of $|\psi\rangle$ and the second one the orthonormality of the basis vectors. Note that the representation depends on the particular basis set that is chosen. Any two orthonormal vectors can form a basis, and a common alternative one is the *Hadamard basis*, given by $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Note that the fact that $\langle\psi|\psi\rangle = 1$ allows us to write

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle, \ \theta \in [0,\pi], \ \phi \in [0,2\pi), \tag{2.4}$$

giving us the Bloch sphere representation in Fig. 2.1.

Note that the Bloch sphere shows $|\psi\rangle$ in $\mathbb{R}^3$, not in $\mathbb{C}^2$, and that the vectors $|0\rangle$ and $|1\rangle$ are not orthogonal on the Bloch sphere.

In case of n qubits, each qubit is represented by

$$|\psi\rangle_i = \alpha_i|0\rangle_i + \beta_i|1\rangle_i, \ i \in [0,n-1], \tag{2.5}$$

where it is evident that not only the coefficients but also the bases are distinct for each qubit. The resulting system is given by

$$|\psi\rangle = \otimes_{i=0}^{n-1}|\psi\rangle_i, \tag{2.6}$$

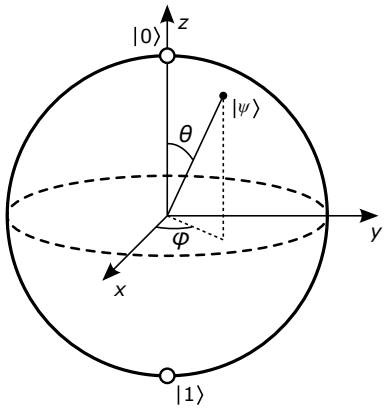


Fig. 2.1 Bloch sphere representation of a single qubit (Wikimedia Foundation, CC BY-SA 3.0)

where $\otimes$ represents the tensor product operator. The resulting state belongs to a complex $N = 2^n$-dimensional space, whose bases are $|0\rangle = |0\rangle_0 \otimes \cdots \otimes |0\rangle_n = |0\ldots00\rangle$, $|1\rangle = |0\rangle_0 \otimes \cdots \otimes |0\rangle_{n-1} \otimes |1\rangle_n = |0\ldots01\rangle$, $\ldots$, $|N-1\rangle = |1\rangle_0 \otimes \cdots \otimes |1\rangle_n = |1\ldots11\rangle$, which gives the final equation

$$|\psi\rangle = \sum_{j=0}^{N-1} \omega_j |j\rangle, \tag{2.7}$$

where the $|j\rangle$ are the new bases and the ω_j are the resulting coefficients coming from the tensor product. The overall state is now represented by a column vector belonging to $\mathbb{C}^N$. It is clear that, starting from an n-dimensional space, we end up in a 2^n-dimensional space with a state that is a linear combination of 2^n basis states instead of just one of them. This unique property, called *quantum superposition*, allows a single qubit to represent and process significantly more information than its classical counterpart. For instance, two qubits can represent four states simultaneously, and 100 qubits can represent approximately $1e30$ states concurrently, leading to an exponential increase in information processing capacity.

When a measurement is made, the superposition collapses, and the qubit settles into a definite classical state, given by just one of the N bases, i.e. $|j\rangle$, with a certain probability $|\omega_j|^2$.

Entanglement is a more perplexing quantum phenomenon, where two or more qubits become intrinsically linked after some operations on them. Their properties become interconnected in such a way that the state of one qubit instantaneously influ-

ences the state of the others, regardless of the physical distance separating them. The result is that measuring a qubit affects the state of the qubits it is entangled with. This connection allows for complex correlations between qubits, which can be harnessed for powerful computational tasks that are intractable for classical computers [8]. An example of a partially entangled state on 3 qubits is $|\psi\rangle = \frac{1}{\sqrt{3}}(|000\rangle + |001\rangle + |100\rangle)$: measuring just the first qubit, if the result is 0 the state of the remaining two qubits becomes $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$, whereas if the result is 1 their state collapses to $|00\rangle$. Note that, given an entangled state on m qubits, it is not possible to obtain it as a tensor product between the m single-qubit states.
The probabilistic nature of quantum programs, where each possible output has an associated probability rather than a deterministic 0 or 1, is a direct consequence of these quantum mechanical principles. This characteristic fundamentally dictates how quantum algorithms must be designed and how their results are interpreted. Also, since it is impossible to know the exact amplitudes of a state if not theoretically, and it is only possible to estimate the magnitudes by measuring the state multiple times, quantum computing necessitates the use of multiple samplings to identify the most likely answer, which can introduce overhead and affect efficiency in practical implementations. This inherent uncertainty is a core aspect of quantum mechanics that influences every facet of quantum algorithm design, making concepts like quantum mean estimation crucial for extracting meaningful information from quantum states [9].

2.3.2 Quantum gates and circuits

Just as classical computers rely on logic gates (such as AND, OR, and NOT) to manipulate bits, quantum computers utilize quantum gates to manipulate qubits. These quantum gates are unitary operations that transform the quantum states of qubits [7]. Unlike classical gates that operate on binary inputs, quantum gates operate on the principles of linear algebra and matrices, allowing them to process and transform superpositions of states [10]. Physically, a gate is obtained differently depending on the particular technology of the qubit. For instance, in the case of superconducting qubits, gates are obtained by means of electromagnetic waves of varying parameters acting on qubits, whereas photonic quantum computers utilize beam splitters, phase shifters, etc. to perform operations on qubits. It is interesting to note that, independently of the technology, all gates can be represented mathematically as unitary

operators U, i.e. such that

$$U^\dagger U = UU^\dagger = I, \tag{2.8}$$

and, representing U with complex matrices, the † operator is again the Hermitian conjugate operator, i.e. $U^\dagger = \overline{U^T} = \overline{U}^T$. If U is a single-qubit gate, then $U \in \mathbb{C}^{2,2}$, whereas if U acts on m qubits then $U \in \mathbb{C}^{2^n,2^n}$. After acting on a state $|\psi\rangle$ with a gate U, we obtain a new state $|\psi'\rangle = U|\psi\rangle$. Detailing the various gates would be out of scope ([11] is the standard reference about this subject), but it suffices to say that the most important single-qubit gates in QML are *rotation gates*, in particular $R_X(\theta_X)$, $R_y(\theta_Y)$ and $R_Z(\theta_Z)$, that respectively perform a rotation about the X axis by an angle θ_X, the Y axis by an angle θ_Y, and the Z axis by an angle θ_Z on the Bloch sphere. On the other hand, the most important 2-qubit gates are the *Controlled-NOT* gate *CX*, that allows for entanglement, and the *SWAP* gate, that swaps the state of two qubits.

Quantum algorithms are constructed by arranging sequences of these quantum gates into *quantum circuits*. These circuits are designed to perform specific computational tasks by evolving the initial quantum state through a series of transformations. A key characteristic that distinguishes quantum circuits from most classical circuits is their inherent reversibility [10]. This means that the input state of a quantum operation can, in principle, be recovered from its output, implying that quantum computations do not lose information. This reversibility is a direct consequence of the unitary nature of quantum mechanics, since the inverse operation of any gate U is given by $U^\dagger$, and $U^\dagger U|\psi\rangle = I|\psi\rangle = |\psi\rangle$. Regarding single-qubit rotation gates in particular, in general they can be represented as a generic rotation $U_k(\theta_k)$ around an arbitrary axis. If the rotation angles are not fixed, but are parameterized and subject to optimization, the circuit is called a *Variational Quantum Circuit (VQC)*, with a set of P parameters $\theta = \{\theta_0, \ldots, \theta_{P-1}\}$ where P is the number of parameterized rotation gates.

2.3.3 Principles of Quantum Data Encoding and Processing

A pivotal step in the application of QML, particularly for tasks involving classical data, is the process of encoding that classical information into a quantum computer's state. Note that the initial state is always $|0\ldots0\rangle$. Encoding is essential to make the data accessible for subsequent quantum information processing routines [7]. Various schemes exist for this purpose, each with its own advantages and limitations. For

instance, amplitude encoding represents data points as the amplitudes of quantum states, allowing for the compression of large datasets into a relatively small number of qubits. In particular, considering a single data point $x \in \mathbb{R}^N$. Then, given $n = \log_2 N$ qubits, the point can be encoded as $|\psi(x)\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$, after normalizing x [12]. The specific choice of encoding scheme is not trivial, as it can significantly impact the runtime and overall efficiency of the quantum circuit and expressivity of the QML model [13].

Once the classical data is encoded into quantum states, quantum information processing routines are applied. The processing stage is designed depending on the desired task, and a large branch of research in QML is dedicated to finding a way to design it in an optimal way, since it is the quantum equivalent of a classical algorithm. Prior to measurement, the whole circuit, comprising the encoding stage, parameterized by x (the input), and the processing stage, parameterized by θ (the optimized parameters), can be formulated as a single unitary operation $U(x, \theta)$. The final stage involves reading out the result of the quantum computation, which is achieved by measuring the quantum system. This measurement process causes the quantum superposition to collapse into a classical outcome, providing the final answer. The processes of getting classical data *into* the quantum computer (the "input problem") and extracting the quantum results *out* as classical information (the "output problem") present significant practical hurdles. Many quantum algorithms rely on Quantum Random Access Memory (QRAM) for fast data loading, which theoretically allows for logarithmic time complexity in data access [14]. Yet, QRAM faces substantial challenges in terms of experimental implementation and demonstrating a genuine computational advantage when compared to highly parallel classical architectures. Furthermore, if the data loading or result extraction steps scale polynomially or exponentially with the size of the dataset, they can effectively negate any theoretical exponential speedup offered by the quantum computation itself. This means that the potential quantum advantage in the core computational task might be lost due to these "ancillary" classical steps, highlighting that achieving practical quantum advantage is not solely about the quantum algorithm's efficiency but also about the entire workflow, including the efficiency of classical-quantum interfaces. This remains a significant practical limitation that researchers must actively address for QML to become truly useful in real-world applications.

2.4 QML paradigms and algorithms: a comparative analysis

2.4.1 Overview of quantum-enhanced Machine Learning

QML algorithms are designed to harness the power of quantum computing to solve complex problems with potentially superior efficiency and effectiveness compared to their classical counterparts.

In the current landscape, most QML implementations adopt hybrid quantum-classical strategies. In these models, a classical computer can use the result of the quantum circuit as an input to perform the easier calculations, leaving the more critical ones to the QC (like in the case of QSVMs), or it can be responsible for optimizing the parameters of a quantum circuit (VQCs), ensuring that the quantum processing segments are kept sufficiently short to mitigate the detrimental effects of noise inherent in current quantum hardware [15]. This hybrid approach is a pragmatic solution that allows researchers to leverage the nascent strengths of quantum computing while relying on the maturity and stability of classical machine learning techniques, thereby combining the power of both paradigms.

2.4.2 Quantum discriminative learning

Discriminative learning in machine learning focuses on creating models that can distinguish between different classes or predict continuous values based on input data. Quantum approaches aim to enhance these capabilities.

Quantum Support Vector Machines (QSVMs)

Support Vector Machines (SVMs) are a cornerstone of classical machine learning, providing robust classification by identifying an optimal separating hyperplane in a high-dimensional feature space. However, as the number of training instances or dimensions grows, the quadratic complexity of classical optimization often becomes a computational bottleneck [16]. Quantum Support Vector Machines (QSVMs) address this limitation by leveraging quantum algorithms to compute the kernel function, which represents the similarity between data points. In the Quantum Kernel

Estimation (QKE) paradigm, a unitary operation $U(x^i, x^j)$ is designed such that the overlap between quantum states encodes the kernel matrix entries $K_{i,j}$. This matrix is then processed by a classical solver to complete the optimization task, effectively combining quantum expressivity with classical stability.

The potential of this approach is evidenced by the research presented in Chapter 3, where it is demonstrated that even a 3-qubit system can outperform classical counterparts on synthetic datasets, achieving a mean accuracy of $79.8\% \pm 8.8\%$ against the classical RBF baseline of $52.8\% \pm 10.2\%$ over 30 independent dataset seeds. A paired t-test on the per-seed accuracy difference ($\bar{\gamma} = +27.0\%$, $\sigma_\gamma = 13.4\%$) yields $p = 6.33 \times 10^{-12}$ [17]. This finding is particularly relevant as it suggests that a quantum advantage in expressivity may be reachable before the era of large-scale fault-tolerant hardware. Nevertheless, the practical deployment of QKE remains hindered by the constraints of NISQ devices. Noise and decoherence often lead to kernel concentration, a phenomenon where the entries of the kernel matrix become exponentially close to a single value, rendering the classifier ineffective. Furthermore, the high execution times and limited qubit connectivity necessitate rigorous feature reduction and hardware-specific optimizations. Addressing these challenges through tailored pulse-level control and geometric optimization, especially on neutral atom arrays, is a primary focus of the following chapters.

Quantum neural networks (QNNs)

Classical Neural Networks are highly non-linear models that have demonstrated exceptional capabilities in complex pattern recognition, regression, classification, and generative tasks. They form the backbone of modern artificial intelligence, achieving state-of-the-art performance across various domains.

Quantum Neural Networks, instead, employ VQCs consisting of parameterized gates to emulate the structure and function of classical neural networks [18], and those parameters are subject to optimization, performed evaluating a cost function at each run of the circuit with the current parameters. The value of the cost function is usually obtained as the mean value of a quantum operator chosen in a way that encodes the cost for that particular task, and it is estimated running the same circuit multiple times and measuring the qubits to obtain a statistically accurate estimation. A fundamental distinction arises from the inherent linearity of quantum mechanics: operators that act on quantum states are always linear. This intrinsic linearity

presents a significant challenge for QNNs attempting to replicate the crucial non-linear activation functions that are central to the expressivity and learning capabilities of classical NNs. To circumvent this, most QNN designs integrate them through the use of quantum kernels or quantum feature encoding. A common scheme for a VQC is shown in Fig. 2.2. The processing stage, parameterized by θ, is called the *ansatz*.

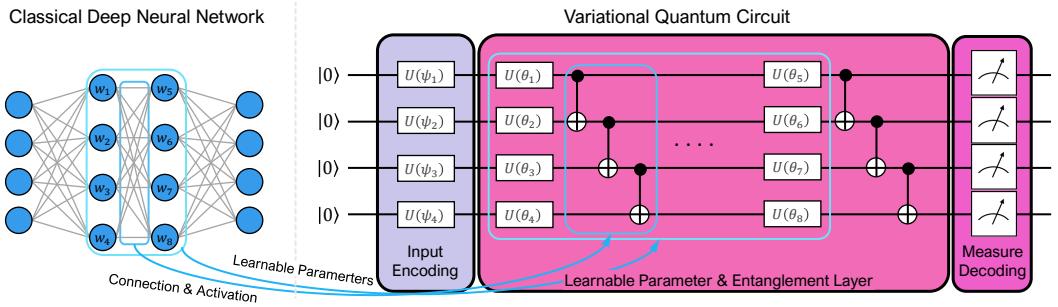


Fig. 2.2 On the left, a classical NN. On the right, its quantum equivalent. The x_i are here named ψ_i, and the gates connecting two adjacent qubits are the CX gates ([1], CC BY-NC-SA 4.0).

Hybrid Quantum-Classical Architectures

Hybrid Quantum-Classical Convolutional Neural Networks (HQCNNs) represent a promising integration where classical layers perform feature extraction and a VQC executes the final classification. While classical CNNs still outperform hybrid models on standard benchmarks like MNIST, HQCNNs demonstrate superior generalization on synthetic quantum datasets, suggesting an inherent advantage in capturing quantum-like patterns [18]. Recent studies indicate that QNNs can be significantly more resource-efficient than their classical counterparts; for instance, they have been shown to require approximately 7.5% fewer FLOPs and 42.3% fewer parameters when scaling to complex problem spaces [18]. This suggests that quantum layers provide a more compact and expressive representation of high-dimensional data, even when simulated on classical hardware.

However, the practical deployment of these models remains constrained by training instabilities. The estimation of gradients in deep quantum circuits is frequently hampered by the "barren plateau" phenomenon, where gradients vanish exponentially, rendering optimization ineffective [15]. Moreover, while hybrid models show a degree of robustness to noise in NISQ environments, the cumulative effect of

hardware errors and the overhead of data encoding continue to be significant barriers. Despite these challenges, the ability of QNNs to naturally manipulate entangled states positions them as a scalable alternative for modeling complex distributions that remain computationally intractable for purely classical architectures.

2.4.3 Quantum Unsupervised Learning

Beyond supervised paradigms, quantum computing offers significant advantages in unsupervised learning, particularly for clustering tasks where discovering latent structures in high-dimensional data is computationally intensive. Traditional algorithms, such as K-Means, are often constrained by a per-iteration complexity of $O(knd)$, making them less efficient as the dimensionality d increases [19]. In contrast, Quantum K-Means leverages quantum superposition and swap tests to estimate distances between data points and centroids, effectively reducing the complexity to $O(kn\log(d))$ quantum operations [20, 21]. This polynomial speedup relative to the feature space dimension highlights the potential of quantum-enhanced clustering for processing large-scale datasets that are otherwise bottlenecked by the "curse of dimensionality" in classical frameworks.

2.4.4 Comparative critical synthesis of the literature

To move beyond descriptive summaries, Table 2.1 contrasts representative QML directions along two criteria: claimed advantage and main empirical weakness.

Table 2.1 Comparative critical synthesis of representative QML directions in recent literature.

Direction	**Claimed advantage**	**Main weakness**
QSVM / QKE	Rich feature maps; potential kernel advantage [16, 17]	Low-dimensional or synthetic datasets; limited device validation
QNN / VQC	Compact, expressive hybrid models [18, 15]	Barren plateaus and unstable optimization; ansatz-dependent reproducibility
Quantum clustering	Asymptotic scaling gains [20, 21]	Benefits reduced by data-loading and readout overheads

The comparative evidence indicates a recurring pattern: current QML literature provides convincing *proof-of-feasibility*, but evidence remains uneven for *robust, reproducible, and scalable* superiority over strong classical baselines. This is particularly visible when studies are compared under consistent metrics (mean $\pm$ standard deviation), repeated runs, and matched resource budgets.

2.5 Real-World Applications of QML

The transformative potential of QML spans a wide array of industrial and scientific domains, primarily driven by its ability to navigate high-dimensional data spaces and optimize complex objectives more efficiently than classical frameworks. In the financial sector, QML is considered a primary candidate for early adoption, particularly in portfolio optimization, fraud detection, and risk analysis. The inherent complexity of multivariate financial environments aligns well with quantum-enhanced optimization strategies, where specialized tools like PennyLane and Qiskit are already facilitating the development of dedicated modules [22–24].

Similarly, in healthcare and materials science, quantum neural networks offer significant advantages for molecular property prediction and drug discovery, as quantum systems are naturally suited to simulate intricate molecular behaviors [25, 26]. However, a critical gap persists in this field: much of the current literature relies on idealized simulations, often overlooking the hardware-specific challenges of noise characterization and error mitigation that are essential for large-scale empirical validation [27]. Beyond these core areas, the reach of QML extends to climate modeling, image recognition, and natural language processing, underscoring its role as a versatile tool for future data-driven decision-making across diverse scientific challenges [28, 15].

2.6 Current Challenges and Practical Limitations

Despite the theoretical advantages of QML, its practical implementation is currently hindered by the constraints of the Noisy Intermediate-Scale Quantum (NISQ) era [29]. In these devices, qubits are highly susceptible to environmental noise and decoherence, which drastically limit the coherence time and, consequently, the maximum

circuit depth that can be executed before the computational signal is overwhelmed by errors [30]. Achieving high-fidelity quantum gates remains a primary bottleneck; while classical machine learning and control theory can be employed to optimize these operations, as detailed in Chapter 4, the physical constraints of the specific hardware technology, such as the sparse connectivity of superconducting transmons, often necessitate a significant overhead of SWAP gates, further aggravating the noise accumulation.

These hardware limitations directly translate into algorithmic challenges, particularly regarding scalability and data encoding. The process of mapping classical data into quantum states (e.g., via amplitude encoding) often requires deep subroutines that are prohibitively expensive for NISQ hardware. Although quantum autoencoders [31] offer a potential pathway for data compression, the overall computational cost of evaluating quantum circuits during training remains high. This constraint frequently limits empirical studies to downsampled or synthetic datasets, highlighting a significant gap between current experimental capabilities and the demands of real-world, large-scale applications. Consequently, the transition toward Fault-Tolerant Quantum Computing (FTQC) and the development of more efficient, platform-specific architectures, such as the neutral atom arrays discussed in Chapter 3, are essential to overcome these bottlenecks and realize a demonstrable quantum advantage in machine learning tasks.

2.6.1 Algorithmic hurdles: barren plateaus and trainability issues

One of the most critical barriers to effectively training quantum models is the *barren plateau* phenomenon [32]. This issue arises when the gradient variance of VQCs dramatically vanishes, or exponentially decreases, as the number of qubits or the depth of the circuit layers increases. When gradients become infinitesimally small across the vast parameter landscape, the model receives little to no effective training signal, rendering gradient-based optimization methods ineffective and preventing the model from learning. Even in scenarios where barren plateaus are less severe, VQCs can still suffer from poor local minima and flat regions in the loss landscape, which further hinder the learning process. This is not an insurmountable problem

but an active area of research with diverse, targeted solutions, demonstrating that the community is directly addressing this algorithmic bottleneck.

The existence of numerous, categorized mitigation strategies indicates that the field is actively moving beyond simply identifying problems to developing sophisticated solutions. This reflects a maturing research area where algorithmic innovation is recognized as being just as critical as hardware advancements for making current-generation quantum computers useful for machine learning.

2.7 Research Gap

Based on the literature synthesis, this dissertation does not aim to close all open QML issues; it focuses on four specific gaps that are directly addressed in the following chapters.

- **Methodologically aligned comparison for QKE**. A recurring limitation is the lack of controlled, repeatable comparisons between quantum-kernel methods and strong classical baselines under consistent metrics and experimental protocols. This dissertation addresses this gap through a unified evaluation setup for the QSVM study.
- **Explicit link between algorithmic gains, hardware constraints, and gate-level realizability**. Reported QML improvements are often detached from concrete platform constraints (native connectivity, circuit depth limits, control overhead), and the literature frequently assumes gate availability without detailing how missing gate primitives are obtained when software libraries do not provide direct implementations; this is an obstacle when applying these methods to technologies that are less explored in the literature, such as neutral atoms, that do not immediately provide gate primitives unlike the superconducting platforms, that are the usual choice for QML experiments. This gap is addressed in this dissertation by evaluating QML performance with hardware-aware criteria and by making the gate-construction pathway explicit for neutral-atom settings.
- **Statistically rigorous validation of data-driven quantum control**. Machine-learning-based approaches to quantum gate synthesis are gaining traction, yet

the reported fidelity figures seldom come with rigorous statistical backing such as confidence intervals or systematic error propagation. This dissertation proposes a complete data-driven pipeline for high-fidelity single-qubit gate synthesis on superconducting hardware, validated through an Adapted Randomized Benchmarking protocol that provides 95% confidence intervals on per-gate fidelity via binomial error propagation and Student's t-test.

- **Measurement-stability evidence for state preservation**. The literature still offers limited hardware-proximate evidence on how protective-measurement strategies can preserve quantum states during repeated readout interactions. This gap is here addressed by providing a photonic simulation benchmark that compares protective and non-protective regimes to quantify finite-step state-survival benefits relevant to robust NISQ workflows.

In this way, the contribution is framed around the gaps effectively covered by this work: reproducible QKE-oriented comparison, hardware-grounded interpretation of results, a clear gate-construction pathway when neutral-atom software stacks lack direct gate implementations, statistically rigorous validation of data-driven gate control on superconducting hardware, and measurement-stability evidence for protective state-preservation mechanisms.

2.8 Summary

This chapter established the theoretical and methodological baseline of the dissertation by defining the core QML paradigms, practical constraints, and research gaps that matter in the NISQ regime. Its contribution to the overall thesis is to provide the criteria used later to evaluate whether a hardware-aware strategy is only conceptually appealing or also operationally viable.

The next chapter (Chapter 3) operationalizes this foundation on neutral-atom hardware through a full QKE implementation, while Chapter 4 addresses the complementary challenge of statistically validated, data-driven gate control on superconducting devices. In this way, the dissertation transitions from framework-level analysis to platform-specific validation.

Chapter 3

Quantum Kernel Estimation With Neutral Atoms For Supervised Classification

3.1 Introduction

As detailed in the previous chapter, QML is an emerging paradigm that seeks to exploit the fundamental principles of quantum mechanics to achieve computational advantages over classical learning algorithms. In recent years, significant theoretical progress has been made in defining the boundaries of this convergence [33–35]. However, at the current stage of technological development, the low number of available qubits restricts experimental benchmarks to problems of limited scale, often making direct comparisons with classical counterparts less than definitive. As noted by Schuld and Killoran [36], research in this era should not focus exclusively on immediate "speedups" but rather on identifying the structural nuances, similarities, and unique capabilities of quantum models. Building these foundations is essential for the future evolution of the field.

3.1.1 Quantum Kernel Estimation and Hilbert Space Mapping

Among the various machine learning architectures, the Support Vector Machine (SVM) stands out as a primary candidate for quantum enhancement. The most

computationally demanding phase of the SVM is the calculation of a kernel function, which corresponds to a feature map of data into a high-dimensional feature space. When this space is sufficiently complex, the mapping becomes classically intractable. Quantum systems are naturally suited for this task due to the exponential growth of their state space with the number of qubits; specifically, the tensor product of N 2-dimensional complex Hilbert spaces $\mathscr{H} = \mathbb{C}^2$ results in a 2^N-dimensional complex Hilbert space:

$$\mathscr{H}_N = (\mathbb{C}^2)^N \tag{3.1}$$

The advantage of Quantum Kernel Estimation (QKE) lies in the ability of the quantum processor to directly compute the inner products that constitute the kernel matrix. This allows the classical optimizer to operate on the high-dimensional mapping without ever needing to explicitly represent the statevectors classically.

3.1.2 The Necessity of Entanglement and the Connectivity Barrier

The efficacy of a quantum feature map is intrinsically linked to the presence of 2-local entangling operations. If a quantum circuit's evolution could be decomposed solely into a tensor product of 1-local (single-qubit) operations, the system could be efficiently simulated on a classical computer. In such a scenario, the global statevector $|\Psi\rangle$ would remain separable:

$$|\Psi\rangle = \bigotimes_{i=N}^{1} |q_i\rangle \tag{3.2}$$

To ensure classical intractability at higher qubit counts, entangling gates (such as the *CX* gate) are mandatory. These gates generate states that cannot be expressed as a tensor product of single-qubit states. For instance, considering a Bell state $|\Phi^+\rangle$:

$$|\Phi^+\rangle = \frac{1}{2}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) \tag{3.3}$$

it can be demonstrated that:

$$\nexists\, \alpha, \beta, \gamma, \delta \in \mathbb{C} : |\Phi^+\rangle = (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma|0\rangle + \delta\,|1\rangle) \tag{3.4}$$

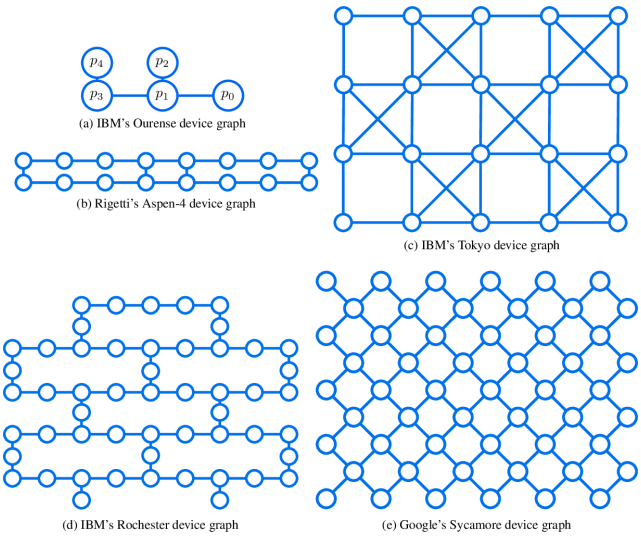


Fig. 3.1 Layout of various superconducting quantum processors (source: [2])

While methodologies for implementing these maps exist for superconducting architectures [37], they are constrained by fixed physical layouts (see Fig. 3.1 [2]). As the number of features increases, the requirement for long-range entanglement necessitates a high density of SWAP gates. This overhead significantly increases circuit depth, thereby exacerbating the system's susceptibility to decoherence.

3.1.3 Neutral Atom Computing: A Flexible Paradigm

Neutral atom quantum computers, typically utilizing Rubidium atoms, offer a compelling alternative to superconducting devices. In these platforms, qubits are not fixed but can be arranged arbitrarily using optical tweezers, subject to specific physical constraints [38]. This geometric freedom allows for more direct connections between qubits, effectively reducing circuit depth and mitigating decoherence.

While previous literature has explored QKE on neutral atoms, for instance in the context of graph-structured data [39], such efforts have largely relied on an analogue approach where the multi-qubit Hamiltonian is obtained via optimization. In this chapter, we propose a different, universal paradigm: the direct implementation of gate-model circuits (specifically those inspired by [37]) on neutral atom hardware. Taking platforms like Pasqal [38] or QuEra [40] as references, we investigate the deployment of QKE for general-purpose datasets.

3.1.4 Chapter Contributions and Organization

This chapter details the transition from low-level pulse control to high-level algorithmic execution. The primary contributions include:

- **Gate Synthesis from Pulses**: We generalize the methods found in [41] to derive a universal set of 1-qubit and 2-qubit gates starting from fundamental laser pulses, creating a layer of abstraction for parameterized gate design.
- **Geometric Optimization**: We discuss the exploitation of neutral atom arrangement for feature mapping, using a 3-qubit configuration as a benchmark due to the computational limits of current simulators based on QuTiP.
- **Empirical QKE Implementation**: We present the formalisms for SVM and QKE, followed by experimental results demonstrating high accuracy on datasets with low linear separation.

The content of this chapter has been previously published by the author in [17].

3.2 Methodology

3.2.1 Quantum gate design in neutral atom quantum computers

In this subsection, since there are currently no functions that implement gates on Pasqal quantum computers, a general method for deriving 1-qubit and 2-qubit gates from pulses is shown, and the algorithms corresponding to the primitives for obtaining each gate are illustrated. The algorithms shown in this section will be used in this chapter for converting a gate circuit into a sequence of pulses, as in Fig. 3.2. When calling a function, if the name of the parameter is not clear from context it is indicated with the notation $parameter \leftarrow parameterValue$.

Single-qubit gates

It is first shown how to obtain single-qubit gates and then it is derived how to obtain the CZ and the CX gates, allowing to achieve a universal gate-based toolset.

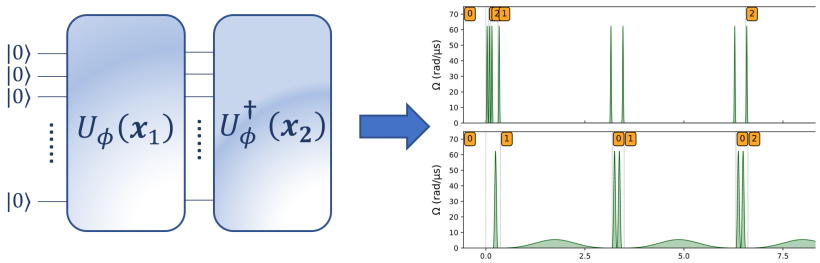


Fig. 3.2 The process of converting a gate circuit into a sequence of pulses for a neutral atom device is achieved using the algorithms presented in this chapter, used for the specific case of Quantum Kernel Estimation.

An arbitrary single-qubit unitary rotation can be expressed as a Z-X-Z rotation by means of Euler angles. This makes it possible to represent the rotation as

$$U(\gamma,\theta,\phi) = R_z(\gamma)R_x(\theta)R_z(\phi), \tag{3.5}$$

and the reason for using a Z-X-Z rotation will be evident later, following the approach presented in [41].
Since

$$R_z(\phi) = \begin{bmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{bmatrix} \tag{3.6}$$

and

$$R_x(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{3.7}$$

then U is equal to

$$U(\gamma,\theta,\phi) = \begin{bmatrix} e^{-i\frac{\phi+\gamma}{2}}\cos\frac{\theta}{2} & -ie^{i\frac{\phi-\gamma}{2}}\sin\frac{\theta}{2} \\ -ie^{i\frac{\gamma-\phi}{2}}\sin\frac{\theta}{2} & e^{i\frac{\phi+\gamma}{2}}\cos\frac{\theta}{2} \end{bmatrix}. \tag{3.8}$$

Up to global phase, i.e. up to a multiplication by a scalar $e^{i\lambda}, \lambda \in \mathbb{R}$ that doesn't make any actual difference on the representation of a quantum state, these equalities hold:

$$R_z(\phi) = U\left(\frac{\phi}{2}, 0, \frac{\phi}{2}\right), \tag{3.9}$$

$$R_x(\theta) = U(0,\theta,0). \tag{3.10}$$

The Hadamard gate, represented by its matrix

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

can be obtained from U as

$$H = U\left(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}\right).$$

Similarly, the Z gate and the X gate, respectively represented by their Pauli matrices

$$\sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$\sigma_X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

can both be obtained from U, using Eq. 3.9 and 3.10, as

$$\sigma_Z = R_z(\pi) = U\left(\frac{\pi}{2}, 0, \frac{\pi}{2}\right),$$

$$\sigma_X = R_x(\pi) = U\left(0, \pi, 0\right).$$

Focusing on Pasqal neutral atom quantum computers, where Rubidium atoms are used, in [41] it is shown that there are generally two different channels, the Rydberg channel and the Raman channel. The Raman channel, also referred to as the digital channel, allows for the discrete manipulation of single qubits, allowing to realize single-qubit gates. There are two possible states, the ground state $|g\rangle$ and the hyperfine state $|h\rangle$, respectively corresponding to $|0\rangle$ and $|1\rangle$. A pulse can drive the transition between these two levels of a qubit by means of the drive Hamiltonian

$$H^D(t) = \frac{\hbar}{2}(\Omega(t)\cos(\phi), -\Omega(t)\sin(\phi), -\delta(t)) \cdot (\sigma_X, \sigma_Y, \sigma_Z) \tag{3.11}$$

or, more compactly,

$$H^D(t) = \frac{\hbar}{2}\Omega(t) \cdot \sigma, \tag{3.12}$$

where $\Omega(t)$ is the Rabi frequency, i.e. the amplitude of the signal measured in $rad\ \mu s^{-1}$, ϕ is the phase of the pulse and $\delta(t) = \omega(t) - \frac{|E_a - E_b|}{\hbar}$, being $\omega(t)$ is the frequency of the signal and E_a, E_b the two energy levels of the qubit. In this chapter only resonant pulses ($\delta = 0$) of duration T and phase ϕ will be considered, so that $\Omega(t) = (\Omega(t)\cos(\phi), -\Omega(t)\sin(\phi), 0)$, generating a rotation along the axis $\hat{u}(\phi) = (\cos\phi, -\sin\phi, 0)$ of the Bloch sphere, by an angle equal to

$$\theta = \int_0^T \Omega(t)dt, \tag{3.13}$$

which corresponds to the following unitary

$$R_{\hat{u}_\phi}(\theta) = \cos\left(\frac{\theta}{2}\right)\mathbb{I} - i\sin\left(\frac{\theta}{2}\right)(\hat{u}\cdot\sigma), \tag{3.14}$$

and this corresponds to

$$R_{\hat{u}_\phi}(\theta) = R_z(-\phi)R_x(\theta)R_z(\phi). \tag{3.15}$$

The difference between Eq. 3.15 and Eq. 3.5 lies in the fact that the leftmost operation in the former is $R_z(-\phi)$, whereas in the latter it is $R_z(\gamma)$. This is irrelevant if this rotation is performed right before measurement, whereas, if other rotations are performed later, a phase shift equal to $\gamma + \phi$ has to be added to the channel, so that the next rotation $R_{\hat{u}_{\phi_2}}(\theta_2)$ will consider the phase shift, i.e.

$$R_{\hat{u}_{\phi_2}}(\theta_2) = R_z(-\phi_2 - \gamma - \phi)R_x(\theta_2)R_z(\phi_2 + \gamma + \phi),$$

which, after the first rotation, corresponds to

$$[R_z(-\phi_2 - \gamma - \phi)R_x(\theta_2)R_z(\phi_2)][R_z(\gamma)R_x(\theta)R_z(\phi)],$$

and this process can be reiterated.

As for the amplitude of the pulse $\Omega(t)$, the Blackman window function is chosen so that the spectral leakage is minimized. This function, with maximum amplitude A $[rad/\mu s]$ and in discrete time, is defined as

$$\Omega(t) = A\left(0.42 - 0.5\cos\left(\frac{2\pi t}{T}\right) + 0.08\cos\left(\frac{4\pi t}{T}\right)\right), \tag{3.16}$$

where $t = n\Delta t$, $T = N\Delta t$ and Δt is the time step. As $\Delta t \to 0$, the area under $\Omega(t)$, which in turn is the rotation angle θ, can be calculated as

$$\theta = \int_0^T \Omega(t)dt = 0.42AT,$$

which means that, to get an angle θ out of a pulse with maximum amplitude A, T must be equal to

$$T = \frac{\theta}{0.42A}. \tag{3.17}$$

The maximum amplitude A is the maximum output of the driving channel and it depends both on the chosen device and channel; in this chapter, the Chadoq2 device was used, with $A = 62.83\ rad\ \mu s^{-1}$. A plot of T depending on θ is shown in Fig. 3.3, with this value of A.

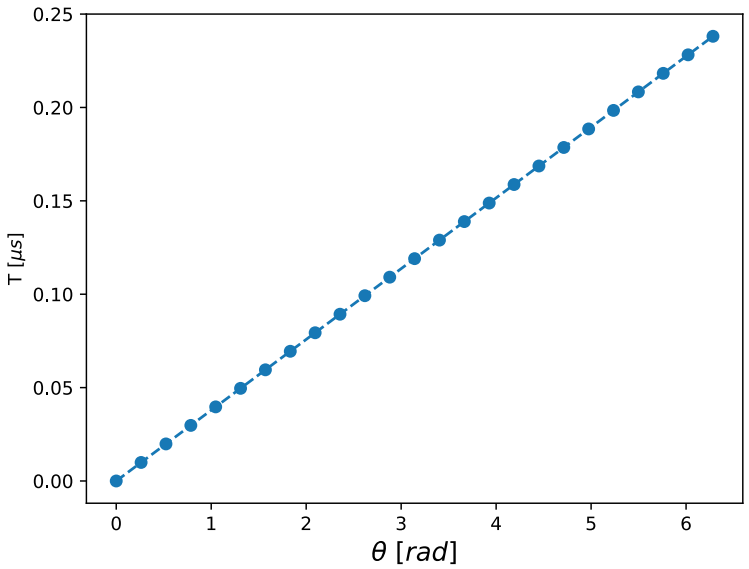


Fig. 3.3 A plot showing the duration T of a pulse corresponding to a single-qubit rotation $U(\gamma, \theta, \phi)$ depending on θ on the Chadoq2 device ($A = 62.83\ rad\ \mu s^{-1}$).

The other gates can be obtained subsequently, as shown in the various algorithms in this chapter.

Algorithm 1 U_{ZXZ}

Parameters: γ, θ, ϕ, max, sequence, channel, qubit
channel.target $\leftarrow$ qubit
$\gamma \leftarrow \gamma \pmod{2\pi}$
$\theta \leftarrow \theta \pmod{2\pi}$
$\phi \leftarrow \phi \pmod{2\pi}$
if $\theta \neq 0$ **then**
    blackman $\leftarrow$ Blackman(max_amplitude $\leftarrow$ max, area $\leftarrow \theta$)
    pulse $\leftarrow$ Pulse(waveform $\leftarrow$ blackman, detuning $\leftarrow$ 0, phase $\leftarrow \phi$, post_phase_shift $\leftarrow (\gamma+\phi) \pmod{2\pi}$)
    sequence.add(pulse, channel)
else
    sequence.phase_shift(shift $\leftarrow \gamma+\phi$, channel, qubit)
end if
return sequence

Algorithm 2 R_X

Parameters: θ, max, sequence, channel, qubit
return $U_{ZXZ}(0, \theta, 0, \text{max}, \text{sequence}, \text{channel}, \text{qubit})$

Algorithm 3 R_Z

Parameters: ϕ, sequence, channel, qubit
sequence.phase_shift(shift $\leftarrow \phi$, channel, qubit)
return sequence

Algorithm 4 X

Parameters: max, sequence, channel, qubit
return $R_X(\pi, \text{max}, \text{sequence}, \text{channel}, \text{qubit})$

Algorithm 5 H

Parameters: sequence, channel, qubit
return $U_{ZXZ}(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \text{max}, \text{sequence}, \text{channel}, \text{qubit})$

Two-qubit gates

In neutral atom devices, what allows entanglement and, in turn, two-qubit gates, is the Rydberg blockade, which is not possible considering only the $|g\rangle$ and $|h\rangle$ states; hence, the Raman channel cannot be used for this purpose. Using the Rydberg channel, instead, a transition between the ground state $|g\rangle$ and the Rydberg state $|r\rangle$ can be driven, in addition to the hyperfine state $|h\rangle$, obtaining a 3-level system. In this configuration, the global Hamiltonian becomes

$$\mathscr{H}(t) = \sum_i \left(H_i^D(t) + \sum_{j<i} \frac{C_6}{R_{ij}^6} \hat{n}_i \hat{n}_j \right), \tag{3.18}$$

where to the drive Hamiltonian of each qubit is added a term that accounts for the van der Waals forces between each pair of atoms. For the device Chadoq2, $C_6\hbar^{-1} = 5008\ GHz\ \mu m^6$.

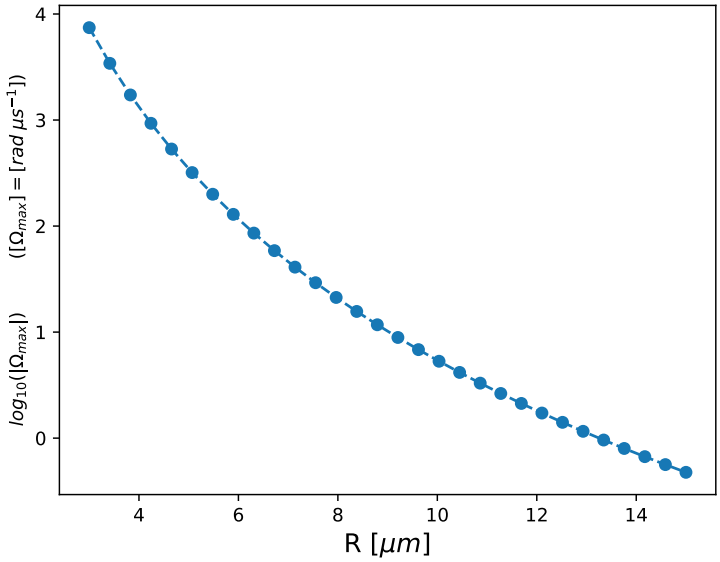


Fig. 3.4 A plot showing the log_{10} of the value of Ω_{max} with respect to the atom distance R for making the Rydberg blockade possible.

Furthermore, R_{ij} is the distance between the i-th and the j-th atoms, and $\hat{n}_i$ is the projector $|r\rangle\langle r|_i$ acting on the i-th atom. When a pulse with $\delta = 0$ and maximum amplitude Ω_{max} is driven through the Rydberg channel, the entanglement between

two atoms at distance R can happen only if

$$R \ll R_b = \left(\frac{C_6}{\hbar\Omega_{max}}\right)^{\frac{1}{6}}, \tag{3.19}$$

being R_b the Rydberg blockade radius, which corresponds to a constraint on Ω_{max} given R. The $\pi - 2\pi - \pi$ sequence proposed in [41] can be used for obtaining a CZ gate with global phase -1, where the considered states are still $|g\rangle = |0\rangle$ and $|h\rangle = |1\rangle$, while $|r\rangle$ is used only for implementing the CZ. However, since the CX gate is needed too, the sequence must be modified. By observing that $X = HZH$, i.e. $Z \sim X$ with respect to the Hadamard basis $\{|+\rangle, |-\rangle\}$, it is sufficient to add a pulse corresponding to the H gate on the target qubit right before and after the 2π pulse, using the Raman channel. On the Rydberg channel, the constraint on the maximum Ω depending on R is only applied for the 2π pulse. In Fig. 3.4, a plot of the logarithm base 10 of Ω_{max} depending on R for obtaining the Rydberg blockade is shown. A detailed explanation of the Blockade effect can be found in the literature (see [42], [43], [44]).

Algorithm 6 *CX*

Parameters: sequence, rydbergChannel, ramanChannel, max, maxRyd, controlQubit, targetQubit
$\pi\_wave \leftarrow$ Blackman(max_amplitude $\leftarrow$ max, area $\leftarrow \pi$)
$2\pi\_wave \leftarrow$ Blackman(max_amplitude $\leftarrow$ maxRyd, area $\leftarrow 2\pi$)
$\pi\_pulse \leftarrow$ Pulse(waveform $\leftarrow \pi\_wave$, detuning $\leftarrow 0$, phase $\leftarrow 0$)
$2\pi\_pulse \leftarrow$ Pulse(waveform $\leftarrow 2\pi\_wave$, detuning $\leftarrow 0$, phase $\leftarrow 0$)
rydbergChannel.target $\leftarrow$ controlQubit
sequence.add(π_pulse, rydbergChannel)
ramanChannel.target $\leftarrow$ targetQubit
sequence $\leftarrow$ H(sequence, rydbergChannel, targetQubit)
rydbergChannel.target $\leftarrow$ controlQubit
sequence.add($2\pi\_pulse$, rydbergChannel)
sequence $\leftarrow$ H(sequence, rydbergChannel, targetQubit)
rydbergChannel.target $\leftarrow$ controlQubit
sequence.add(π_pulse, rydbergChannel)
return sequence

3.2.2 Arrangement of neutral atoms

If the chosen device allows to position the atoms in three-dimensional space and not just on a plane, a lattice could be ideally created where the innermost atoms are arranged as in Fig. 3.5.

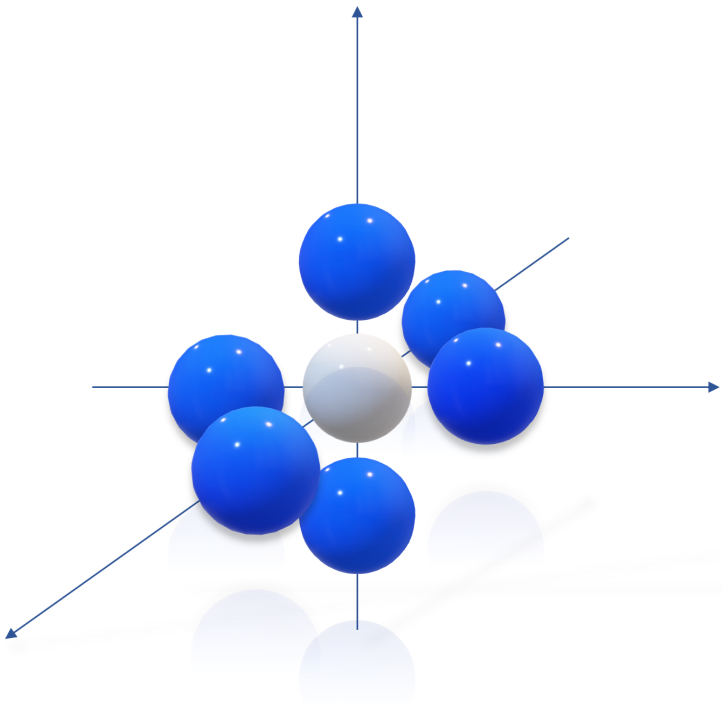


Fig. 3.5 A possible pattern of arranging the neighbors of each atom, if a fixed distance d is wanted. In 3D space, only 6 neighbors are possible with this constraint.

In particular, assuming that a fixed distance d between atoms is wanted, it is clear that, considering a cube of length $2d$ and an atom placed at its center, all of the atoms that are at the center of each of the 6 faces of the cube will be at distance d from it. This pattern can be repeated in space, hence creating a lattice of atoms where, inside of it, each atom is at distance d from its 6 neighbors. If it is not important that the distance be constant, it can be seen that, building a lattice following the pattern in Fig. 3.6, the central atom (white) will have a minimum distance of d and a maximum distance of $d\sqrt{3}$ from its neighbors, with the advantage that now it has 26 neighbors instead of just 6, which allows for direct entanglement between the central qubit and all the other qubits in that neighborhood without using swap gates, which is a noticeable reduction considering the fact that, with superconducting devices, a qubit

can generally be put in direct entanglement with at most 2 other qubits. Naturally, the minimum distance requirement of the device and the maximum distance according to the Rydberg blockade radius must be respected. In this chapter, however, a device that only allows planar arrangement is considered, because as of today there is no possibility to simulate a device with spatial arrangement, but the same arguments hold for either scenarios.

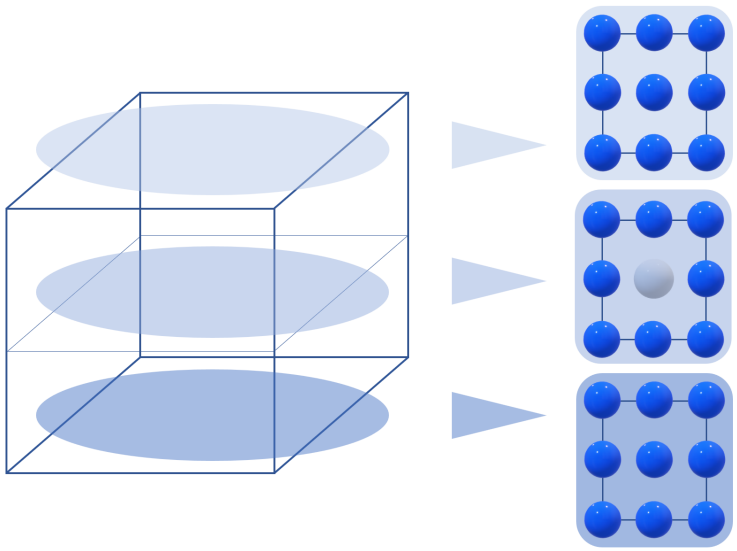


Fig. 3.6 Another possible pattern of arranging the neighbors of each atom. A cube of length $2d$ is considered, and the top face, the bottom face and the face resulting from a central cut are expanded on the right to show the atom arrangements. The central atom (in white) is at a minimum distance d and at a maximum distance $d\sqrt{3}$ from all its neighbors. With this pattern, it is possible to have a total of 26 neighbors for each atom.

3.2.3 Support Vector Machines

In supervised binary classification, the objective is to determine whether a given sample belongs to one class or the other. SVM is a machine learning technique that consists in finding a separating hyperplane between the samples of the two distributions.

Formally, each sample with f features can be seen as a vector $x \in \mathscr{X} \subset \mathbb{R}^f$, paired

with a label $y \in \mathscr{Y} = \{+1, -1\}$. The training set is therefore the set of all points [1]

$$M = \{\omega_{M,i} = (x_{M,i}, y_{M,i}) \in \Omega = \mathscr{X} \times \mathscr{Y}\}_{i \in [m]},$$

used in the training phase, whereas the test set is

$$S = \{\omega_{S,i} \in \Omega\}_{i \in [s]},$$

used in the testing phase.

The training step consists in solving the following optimization problem,

$$\min_{w,b} \left(\lambda ||w||^2 + L_S^{hinge}((w,b))\right), \tag{3.20}$$

where $L_S^{hinge}((w,b))$ is the average of the hinge loss $l^{hinge}((w,b),(x,y))$ over all the samples (x_M, y_M) of the training set, and the hinge loss is defined as

$$l^{hinge}((w,b),(x,y)) = \max(0, 1 - y(\langle w,x\rangle + b)) \tag{3.21}$$

so that the resulting parameters define the linearly separating hyperplane (see [45] for an in-depth derivation). After this step, a classifier f can be derived such that $f : \mathbb{R}^f \to \mathscr{Y} \cup \{0\}$, where the value 0 can be associated with the label $+1$.
However, since data is frequently non-linearly separable, a nonlinear function is applied to data in order to account for nonlinearities, which results in an increased number of dimensions. This is allowed by means of a mapping $\Phi : \mathscr{X} \to \mathscr{F}$, where $\mathscr{F}$ is a Hilbert space called the *feature space*. Since the mapping can be computationally hard to calculate for each sample if $\mathscr{F}$ has a high dimension, the *kernel trick* is used. Defining the *kernel* as a function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$, where

$$K(x_a, x_b) = \langle \Phi(x_a), \Phi(x_b) \rangle, \tag{3.22}$$

it is often the case that, given two points, their kernel, i.e. the scalar product of their feature maps, is much easier to calculate than their respective feature maps separately. Due to the *representer theorem* (see [45]), there exists an α such that the

[1] $[n] = \{1, \ldots, n\}$

optimal solution of the problem can be written as

$$w^* = \sum_{i=1}^{M} \alpha_i \Phi(x_{M,i}) \tag{3.23}$$

and, introducing $K_M \in \mathbb{R}^{m,m}$ as the *training kernel matrix* defined as

$$(K_M)_{i,j} = K(x_{M,i}, x_{M,j}), \tag{3.24}$$

it can be derived that

$$\langle w, \psi(x_{M,i}) \rangle = (K_M \alpha)_i \tag{3.25}$$

and

$$\|w\|^2 = \alpha^T K_M \alpha, \tag{3.26}$$

so that the optimization problem is now expressed as

$$\min_{\alpha,b} \left(\lambda \alpha^T K_M \alpha + \frac{1}{m} \sum_{i=1}^{m} \max\left(0, 1 - y_i (K_M \alpha)_i + b\right) \right). \tag{3.27}$$

At this point, to classify an instance x_S belonging to the test set, it is sufficient to calculate the quantity

$$\hat{y}_S = sign\left(\sum_{i=1}^{m} y_{M,i} \alpha_i K(x_S, x_{M,i}) + b \right). \tag{3.28}$$

It is therefore natural to calculate the *test kernel matrix* $K_S \in \mathbb{R}^{s,m}$, defined as

$$(K_S)_{i,j} = K(x_{S,i}, x_{M,j}), \tag{3.29}$$

to speed up the evaluation of the accuracy on the test set.

3.2.4 Quantum Kernel Estimation

When the kernel is hard to compute classically, a quantum computer can be used instead, leaving the training part to the classical computer once the kernel is cal-

culated. In particular, a unitary $U_\Phi(x)$ can be realized such that, starting from the state $|0\rangle = |0\ldots0\rangle$, it performs the feature mapping of x directly on the quantum computer, obtaining $U_\Phi(x)|0\rangle$. Following the method presented in [37], given a state ψ, the density matrices $|\psi\rangle\langle\psi|$ are considered instead of their statevector $|\psi\rangle$, so that inner products of two different states do not depend on the global phase of their representation. This is why the kernel is calculated as $K(x_1,x_2) = |\langle\Phi(x_2)|\Phi(x_1)\rangle|^2$. This is equivalent to

$$K(x_1,x_2) = |\langle 0|U_\Phi^\dagger(x_2)U_\Phi(x_1)|0\rangle|^2, \tag{3.30}$$

which is the square of the probability of measuring all 0 in the Pauli-Z base after performing $U_\Phi^\dagger(x_2)U_\Phi(x_1)|0\rangle$. In other words, the kernel matrix $(K_M)_{i,j} = K(x_{M,i},x_{M,j})$ can be estimated by performing many times each $U_\Phi^\dagger(x_{M,j})U_\Phi(x_{M,i})|0\rangle$ and calculating the square of the counts of 0 measurements. In the same way, K_S can be estimated. Naturally, the higher the number of runs for each (i,j) couple, the higher the accuracy of the estimation.
In summary, the optimization problem is finally expressed by substituing in Eq. 3.27 the K_M that is estimated.

As shown in the previously cited paper, considering the training step and being $|M|$ the cardinality of the training set, a total number of executions equal to $O(\delta^{-2}|M|^4)$ allows to obtain a training kernel matrix $\hat{K}_M$ that differs in operator norm from the true K_M by at most $\|K_M - \hat{K}_M\|^2 \leq \delta$.
Finally, the training and the testing steps of the SVM are performed on a classical computer by using the kernel matrices calculated in the previous step.

3.2.5 Experimental setup

The experiment is performed simulating the Pasqal Chadoq2 device. According to [46], only planar arrangement of atoms is possible, with a maximum of 100 atoms, a minimum distance of $4\mu m$ between each pair and a maximum distance from the origin equal to $50\mu m$. Both the local Rydberg channel and the local Raman channel share the specifications listed in Table 3.1.

Since only resonant pulses will be used, i.e. with $\delta = 0$, the constraint about the maximum $|\delta|$ does not interfere with the experiment. It is however worth to notice that every time a channel changes the target qubit there is a loss of at least $0.22\ \mu s$.

Table 3.1 Chadoq2 local channels specifications

Quantity	***Value***
Maximum Ω	62.83 $rad\ \mu s^{-1}$
Maximum $\lvert\delta\rvert$	125.7 $rad\ \mu s^{-1}$
Minimum time between retargets	220 ns
Clock period (τ_{clk})	4 ns

Additionally, the Chadoq2 channels enforce a *clock period* τ_{clk}: every pulse duration must be an integer multiple of τ_{clk}. If the ideal duration T obtained from Eq. (3.17) is not a multiple of τ_{clk}, it is rounded up to the nearest valid value $T' = \lceil T/\tau_{\text{clk}} \rceil \tau_{\text{clk}} \geq T$. Crucially, this quantization does not alter the rotation angle θ. Given the desired area θ and the new duration T', the Blackman waveform is regenerated with a reduced peak amplitude $A' \leq A$ such that the integral $\int_0^{T'} \Omega(t)\, dt$ remains exactly equal to θ. In other words, stretching the pulse in time is compensated by lowering its peak, preserving the area and therefore the rotation angle. Since the peak is only decreased, the hardware constraint $\Omega(t) \leq A$ is never violated.

A register of $N = 3$ qubits is prepared as in Fig. 3.7, with a minimum distance of $4\mu m$ and a maximum distance of 4.47 μm between atoms.

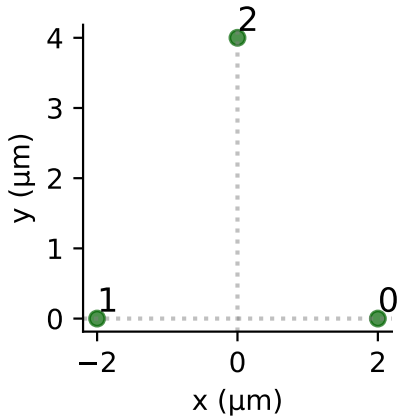


Fig. 3.7 Arrangement of the atoms in this experiment with the Chadoq2 device. The distance $|q_0 - q_1|$ is 4 μm, while the distance $|q_0 - q_2|$ and $|q_1 - q_2|$ is 4.47 μm

The dataset is generated using Qiskit's function `ad_hoc_data` inside the package `qiskit_machine_learning.datasets`, varying the seed, with a training set size equal to 20, a test set size equal to 10, 3 features and a gap equal to $\Delta = 0.1$.

Since there are two possible classes, the function actually generates a dataset with $m = |M| = 40$ training samples and $s = |S| = 20$ test samples, where half samples are of one class and half samples are of the other one.

More formally, the dataset is prepared by choosing a unitary $V \in SU(2^3)$ and $f = Z_1Z_2Z_3$, so that the ground truth labels of each sample are assigned in such a way that, given a randomly generated sample $x \in (0, 2\pi]^3$, its label is 1 if $\langle\Phi(x)|V^\dagger fV|\Phi(x)\rangle \geq \Delta$, while if $\langle\Phi(x)|V^\dagger fV|\Phi(x)\rangle \leq -\Delta$ the label -1 is assigned (the unitary $\Phi(x)$ is the same used in the QKE circuit for performing the feature mapping). To give and idea of the complexity of the separation of the classes, in Fig. 3.8 the ground truth regions are shown for the frontier of the $(0, 2\pi]^3$ volume, where red corresponds to label -1, blue to $+1$ and white to the separation gap.

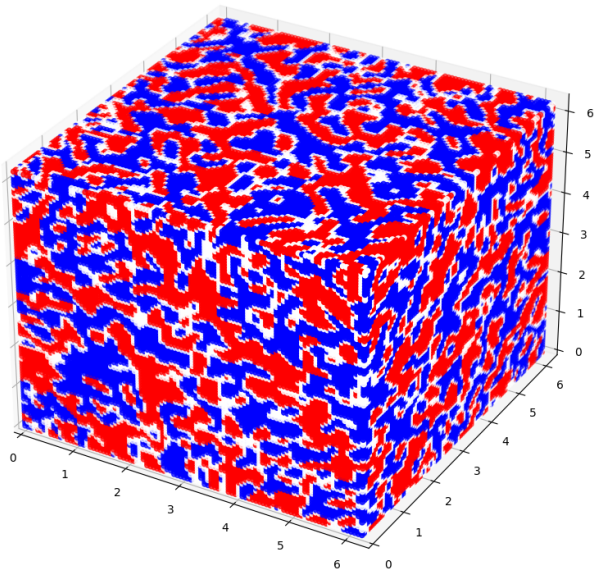


Fig. 3.8 The ground truth regions are shown for the frontier of the $(0, 2\pi]^3$ volume. The blue regions are the points x such that $\langle\Phi(x)|V^\dagger fV|\Phi(x)\rangle \geq \Delta$, whereas the red regions are such that $\langle\Phi(x)|V^\dagger fV|\Phi(x)\rangle \leq -\Delta$. The separation gap is in white. The function `ad_hoc_data` was modified to return a $100 \times 100 \times 100$ uniform grid instead of a $20 \times 20 \times 20$ one when $n = 3$, to obtain a clearer visualization. A seed equal to 10000 was used. Inside the volume, the regions of each section change with continuity.

To perform the feature mapping on the neutral atom device, the circuit generated by the instruction `ZZFeatureMap` inside the package `qiskit.circuit.library`, with $N = 3$ features is taken as a reference. In Fig. 3.9 the feature map circuit with 1 repetition is shown.

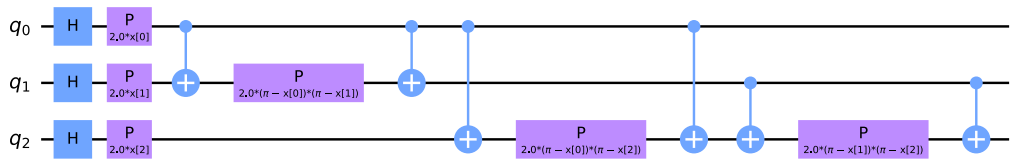


Fig. 3.9 The gate circuit that implements the feature map $U_\phi(x)$ with 1 repetition. The actual circuit uses a $U_\phi(x)$ with 2 repetitions, followed by a $U_\phi^\dagger(x)$.

This circuit is parameterized on each possible x.

The actual $U_\Phi(x)$ is obtained with 2 repetitions of the feature map circuit. Furthermore, the circuit needed for performing the QKE is obtained by concatenating $U_\Phi^\dagger(x)$ to $U_\Phi(x)$.

By using the methods explained in Section II, a sequence is built for each x_i with Pulser, where each pulse corresponds to a gate. For the estimation of the training kernel matrix, a total of m^2 sequences that represent the evolutions $\xi_M(i,j)$ are obtained, where

$$\forall(i \in [m], j \in [m]), \xi_M |0\rangle : (i,j) \mapsto U_\Phi^\dagger(x_{M,i}) U_\Phi(x_{M,j}) |0\rangle .$$

For the estimation of the test kernel matrix, instead, a total of sm sequences representing $\xi_S(i,j)$ are obtained, where

$$\forall(i \in [s], j \in [m]), \xi_S |0\rangle : (i,j) \mapsto U_\Phi^\dagger(x_{S,i}) U_\Phi(x_{M,j}) |0\rangle .$$

As an example, the sequence corresponding to $\xi_M(1,1)$ is shown in Fig. 3.10.

The training kernel matrix entries $(K_M)_{i,j}$ are estimated by sampling 1000 times from the Z measurement distribution out of each sequence $\xi_M(i,j)$ and taking the square of the frequency of the 0 outcome. Similarly, the testing kernel matrix entries $(K_S)_{i,j}$ are estimated by sampling 1000 times from the Z measurements distribution out of each sequence $\xi_S(i,j)$ and taking the square of the frequency of the 0 outcome. $K_M \in [0,1]^{m,m}$ and $K_S \in [0,1]^{s,m}$ are then used by a classical computer for training and testing the SVM algorithm on the dataset, using the Scikit-Learn SVC function with default parameters and the precomputed kernel matrices. The performance is

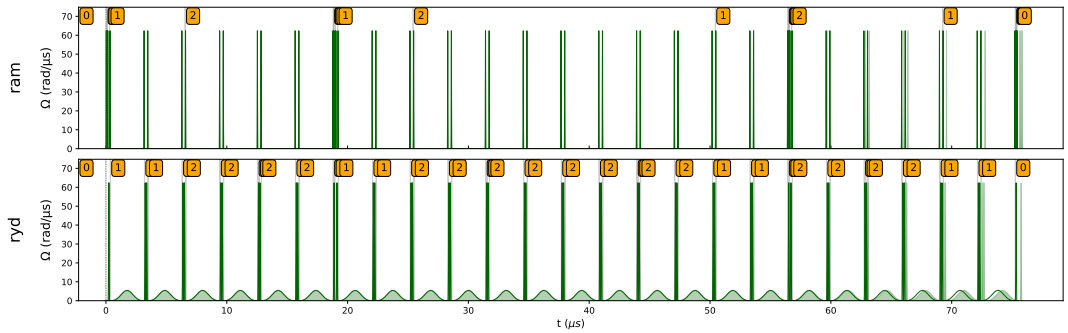


Fig. 3.10 The resulting sequence of pulses corresponding to the QKE circuit ($U_\phi(xU_\phi^\dagger(x))$. The channel at the top is the local Raman channel, whereas the channel at the bottom is the local Rydberg channel. The highlighted numbers indicate the target qubit of each pulse, that changes at each transition.

compared with the one obtained with a radial basis function kernel as a classical counterpart.

3.3 Results

The Hadamard, Phase (R_z) and CX gates are obtained with pulses using the methods shown previously. Given Eq. 3.17 with $A = \Omega_{max} = 62.83\ rad\ \mu s^{-1}$ and $\theta \in (0, 2\pi]$, then, growing linearly with the desired θ, it holds $T \in (0, 0.238]\ \mu s$, which is the interval of the possible duration of a pulse for single-qubit gates. For the Rydberg 2π pulse, since a conservative blockade radius equal to 10 μm is considered, the corresponding maximum Rabi frequency is $\Omega_{max} = 5.42\ rad\ \mu s^{-1}$ (only for the 2π pulse), corresponding to a duration $T_{2\pi} = 2.76\ \mu s$. It was seen that reducing the considered blockade radius made the CZ pulse more imprecise. The average QKE sequence on each pair of samples lasts approximately $75\mu s$.

After running each sequence $\xi_M(i, j)$ and $\xi_S(i, j)$, the matrices K_M and K_S are estimated as shown in the previous sections. A heatmap of the estimated matrix K_M, obtained on a dataset generated with the pseudorandom seed equal to 10000, is shown in Fig. 3.11 to illustrate the values that are obtained. The closer two samples i and j are in the feature space, the closer the corresponding $K_M(i, j)$ is to 1. Naturally,

$(K_M)_{i,i} = 1$ since

$$\xi_M(i,i)\,|0\rangle = U_\Phi^\dagger(x_{M,i})U_\Phi(x_{M,i})\,|0\rangle = |0\rangle\,,$$

and $|\,\langle 0|0\rangle\,|^2 = 1$. After training, all of the training samples become support vectors.

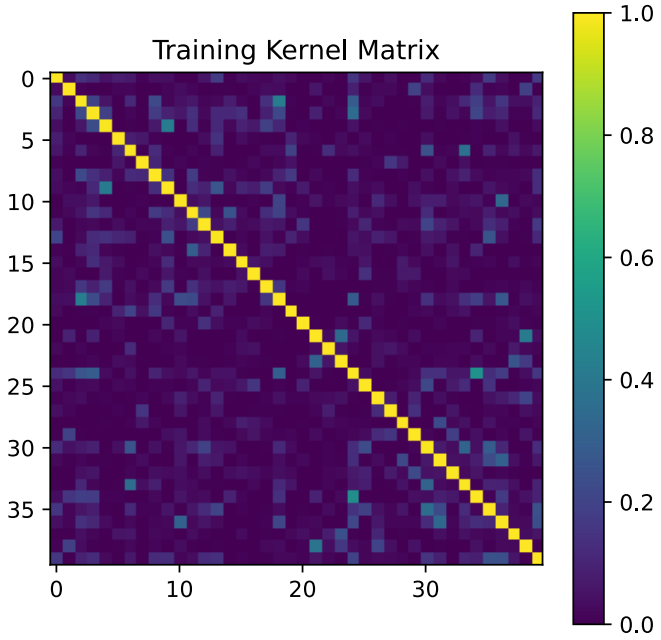


Fig. 3.11 A heatmap of the estimated training kernel matrix, where at row i and column j the color corresponding to the value $K_M(i,j)$ is found. The diagonal elements are all equal to 1. A seed equal to 10000 was used

To account for statistical fluctuations depending on different seeds and on sampling noise, the actual experiments are described in the following subsections.

3.3.1 Experiment 1: Multi-seed statistical comparison

A single dataset instance is insufficient for drawing robust conclusions about the relative performance of the quantum and classical classifiers, since accuracy may depend on the specific train/test split. To address this, the full QKE pipeline was repeated across 30 independent dataset instances, generated by varying the pseudo-

random seed from 10000 to 10029. For each seed, a fresh training set ($m = 40$, i.e. 20 samples per class) and test set ($s = 20$, i.e. 10 samples per class) are generated using `ad_hoc_data` with $N = 3$ features and gap $\Delta = 0.1$. All other experimental parameters are kept identical: $N_{\text{shots}} = 1000$ per kernel entry, Chadoq2 device simulation, $r = 2$ feature-map repetitions.

For each of the 30 runs, both a QSVM (with the precomputed quantum kernel) and a classical SVC with radial basis function (RBF) kernel are then trained and evaluated on the same data split. This paired design ensures that differences in accuracy are attributable to the kernel type rather than to data variability. All accuracy values in the following tables are reported as $\mu \pm \sigma$, where μ is the sample mean and σ the sample standard deviation.

The results across 30 runs are summarized in Table 3.2.

Table 3.2 Multi-seed experiment: QSVM vs. classical RBF SVC over 30 dataset seeds

Metric	**Value**
Number of seeds	30
QSVM accuracy	0.7983 ± 0.0876
RBF accuracy	0.5283 ± 0.1023
γ (QSVM $-$ RBF)	$+0.2700 \pm 0.1336$
Paired t-statistic	11.065
p-value (two-sided)	6.33×10^{-12}

A two-sided paired Student t-test on the per-seed accuracy differences $\gamma_i = \text{acc}_{\text{QSVM},i} - \text{acc}_{\text{RBF},i}$ yields $t = 11.065$ with $p = 6.33 \times 10^{-12}$, rejecting the null hypothesis $H_0 : \mu_\gamma = 0$ with overwhelming statistical significance (well below the conventional $\alpha = 0.01$ threshold). The mean accuracy advantage of $+27.00$ percentage points, with a standard deviation of 13.36 percentage points, is both statistically significant and practically relevant: across all 30 seeds, the quantum kernel never underperforms the classical one.

It is worth noting that, because the test set contains $s = 20$ samples, individual accuracy values are discrete multiples of $1/20 = 0.05$. This discretization does not affect the validity of the t-test, since the sample means over 30 runs are continuous estimators of the true expected accuracy, and the Central Limit Theorem ensures approximate normality of the mean at this sample size.

3.3.2 Experiment 2: Sampling noise sensitivity

The kernel entries are estimated via finite measurement shots ($N_{\text{shots}} = 1000$), which introduces sampling noise into the kernel matrices. To quantify the impact of this noise on classification accuracy, we fixed the dataset seed to 10000 and performed $K = 30$ independent samplings for each entry of the kernel matrix.

Concretely, for each pair (i, j), the QuTiP-based emulation of the sequence $\xi(i, j)$ is performed once, yielding the exact final state. From this state, $K = 30$ independent measurement rounds of $N_{\text{shots}} = 1000$ shots each are drawn. The k-th measurement round ($k = 1, \ldots, K$) provides the entry (i, j) of the k-th kernel matrix. This procedure is applied to both the training and test sequences, yielding K training kernel matrices $K_M^{(1)}, \ldots, K_M^{(K)}$ and K test kernel matrices $K_S^{(1)}, \ldots, K_S^{(K)}$. For each pair $(K_M^{(k)}, K_S^{(k)})$, a QSVM is trained and evaluated, producing K accuracy values. This design isolates the sampling noise from other sources of variability (dataset generation, gate synthesis).

The results are summarized in Table 3.3.

Table 3.3 Sampling noise analysis: $K = 30$ repeated kernel samplings on seed = 10000

Metric	**Value**
Dataset seed	10000
Samplings per entry (K)	30
Shots per sampling (N_{shots})	1000
QSVM accuracy	0.7617 ± 0.0409

The QSVM accuracy fluctuates due to the stochastic nature of finite-shot sampling, yielding a standard deviation of $\sigma \approx 0.041$. This is a relatively small variability: the coefficient of variation is $\approx 5.4\%$, confirming that $N_{\text{shots}} = 1000$ provides a sufficiently stable kernel estimate for this problem size.

Importantly, even under sampling noise, the QSVM accuracy remains concentrated around its mean, with all $K = 30$ values falling within a narrow band. This confirms that the finite-shot kernel estimation is reliable enough for classification at this problem scale.

3.3.3 Experimental details

All experiments were executed using the Pulser library [41] for pulse-level sequence construction, QuTiP [47] for full-state emulation, and Scikit-Learn for the SVM training and evaluation. The dataset was generated using Qiskit's `ad_hoc_data` function from the `qiskit_machine_learning` package. Table 3.4 summarizes the key experimental parameters common to both experiments.

Table 3.4 Experimental parameters for both experiments

Parameter	**Value**
Device	Chadoq2 (simulated)
Number of qubits / features (N)	3
Feature map	`ZZFeatureMap`, $r = 2$ reps
Training samples per class	20 ($m = 40$ total)
Test samples per class	10 ($s = 20$ total)
Separation gap (Δ)	0.1
Shots per kernel entry (N_{shots})	1000
Max Rabi frequency (Raman)	62.83 rad μs^{-1}
Blockade radius	10 μm
Dataset seeds (Exp. 1)	10000–10029
Fixed seed (Exp. 2)	10000
Kernel repetitions K (Exp. 2)	30

Since each of the 30 runs in Experiment 1 requires the construction and emulation of $m^2 + sm = 40^2 + 20 \times 40 = 2400$ pulse sequences, the total computational cost is substantial. To make the experiments tractable, the kernel matrix computation was parallelized across multiple CPU cores using a process-based pool with the `fork` multiprocessing context. This choice was dictated by the fact that the QuTiP ODE solver (`ZVODE`) is not thread-safe; process-level isolation avoids race conditions while still exploiting multi-core hardware.

The full source code, including the parallelized kernel evaluator, the statistical analysis pipeline, and the reproducible experiment notebooks, is publicly available at [48].

3.3.4 Discussion

The two experiments above provide complementary evidence. Experiment 1 (Section 3.3.1) demonstrates that the quantum kernel advantage is not an artifact of a particular dataset instance: over 30 independent data splits, the QSVM consistently and significantly outperforms the RBF baseline ($p < 10^{-11}$). Experiment 2 (Section 3.3.2) shows that, for a fixed dataset, the finite-shot estimation of the quantum kernel introduces only modest variability ($\sigma \approx 0.04$), insufficient to erode the accuracy gap.

Despite the very low separation ($\Delta = 0.1$) on the dataset and the small number of samples, a high accuracy is reached. The main advantage of the quantum approach is that, for a high number of features, the classical kernel computation would become not only less accurate, at least for this particular problem, but also unfeasible, whereas the quantum one does not have this critical issue. Due to the impossibility of running the sequences on real hardware, the simulation was restricted to the case of only $N = 3$ features because of the computational effort required by QuTiP when the number of qubits increases. The method presented in this chapter can be used for any number of features, if the chosen technology allows it, especially if spatial arrangement is permitted so that it can be fully exploited. Clearly, as the number of qubits increases, the advantage brought by the neutral atom technology is more evident, given the fact that much more 1-to-many connections among qubits are needed.

It is worth noticing that the purpose of quantum feature kernels makes sense for datasets such as the one used in this experiment, i.e. for data that needs to be brought into a high-dimensional feature space to be actually separable. Regarding the specific dataset used in this experiment, it is evident that it was designed to make a high classification accuracy possible using the classification method illustrated in this chapter and in [37], and it is just an artificial example for theoretical purposes. However, it can still be the case that in a real scenario a high-dimensional feature space could be needed, in which case quantum feature kernels present an exponential computational advantage over classical kernel computation methods. Future research should focus on the study of which real scenarios are actually of this kind, since as of today it is not clear, making quantum feature kernels only potentially advantageous in purely theoretical cases.

On a final note, while it is true that neutral atom devices allow for better connectivity, a pulse corresponding to a gate usually takes some microseconds, whereas on superconducting devices a gate takes a few nanoseconds. Since the duration T of a pulse is inversely proportional to the maximum channel output of the device, improving the latter can drastically reduce T. Furthermore, a sequence such as the one used in this chapter (Fig. 3.10) takes almost 80 μs to run, while, as of today, Pasqal devices only support sequences that last up to 10 μs due to decoherence.

3.3.5 Scalability analysis: scaling with qubit count

We now derive how the quantum resources of the QKE pipeline scale with the number of qubits (features) N. The analysis is grounded in the specific gate structure of the `ZZFeatureMap` used in this chapter.

Gate count derivation from the ZZFeatureMap

One repetition of the `ZZFeatureMap` on N qubits consists of two layers:

1. A *single-qubit layer*: N parameterized $R_z(x_i)$ rotations, one per qubit. This contributes N single-qubit gates.

2. An *entangling layer*: for every pair (i, j) with $i < j$, a block $CX_{i,j}-R_z(x_i x_j)-CX_{i,j}$ is applied. The number of such pairs is $\binom{N}{2} = \frac{N(N-1)}{2}$, and each block uses exactly 2 CX gates and 1 R_z gate. Hence the entangling (CX) and single-qubit (R_z) gate counts per repetition are

$$G_{CX}(N) = 2\binom{N}{2} = N(N-1), \qquad G_{R_z}^{\text{ent}}(N) = \binom{N}{2} = \frac{N(N-1)}{2}. \quad (3.31)$$

The total single-qubit gate count per repetition (combining both layers) is therefore

$$G_{1Q}(N) = N + \frac{N(N-1)}{2} = \frac{N(N+1)}{2}. \quad (3.32)$$

Since $U_\Phi(x)$ uses r repetitions and the full QKE circuit is $U_\Phi^\dagger(x_j)U_\Phi(x_i)$, the total gate counts per kernel entry are

$$G_{CX}^{\mathrm{QKE}}(N,r) = 2r\cdot N(N-1) = O(rN^2),$$
$$G_{1Q}^{\mathrm{QKE}}(N,r) = 2r\cdot\frac{N(N+1)}{2} = rN(N+1) = O(rN^2). \tag{3.33}$$

In this chapter $r=2$, giving $G_{CX}^{\mathrm{QKE}}(N,2) = 4N(N-1)$ and $G_{1Q}^{\mathrm{QKE}}(N,2) = 2N(N+1)$.

Neutral atoms: routing overhead

For small N, all atoms can be placed within the Rydberg blockade radius of each other, achieving all-to-all connectivity with zero routing overhead and a total gate count of $O(N^2)$. However, for large N this is no longer possible: each atom has a bounded number of neighbors within the blockade radius. As discussed in Section 3.2.2, in the 3D cubic lattice arrangement each atom has at most 26 neighbors, i.e. the degree is $deg = 26$. On this regular 3D lattice of side $\lceil N^{1/3}\rceil$, the graph distance between two atoms scales as $O(N^{1/3})$. Since a CX between non-adjacent atoms requires routing via SWAP gates (each decomposing into 3 CX gates), the overhead per pair is $O(N^{1/3})$ additional CX gates. Summing over all $O(N^2)$ pairs, the total SWAP overhead on a 3D neutral-atom lattice is

$$G_{\mathrm{SWAP}}^{\mathrm{3D}}(N) = O\Big(N^2\cdot N^{1/3}\Big) = O(N^{7/3}). \tag{3.34}$$

In conclusion, the all-to-all claim $O(N^2)$ with no routing overhead holds strictly only when $N \le deg+1$.

Comparison with fixed-connectivity devices

On a linear-chain superconducting device, a CX between non-adjacent qubits i and j requires routing via SWAP gates. Each SWAP decomposes into 3 CX gates. For a pair at distance $d = |i-j| > 1$, at least $d-1$ SWAPs are needed to bring the qubits adjacent, giving $3(d-1)$ additional CX gates. On a linear chain of N qubits, the number of pairs at distance exactly d is $(N-d)$. The total SWAP overhead (in

additional CX gates) per repetition is therefore

$$G_{\text{SWAP}}^{\text{linear}}(N) = \sum_{d=2}^{N-1} (N-d) \cdot 3(d-1) = O(N^3), \tag{3.35}$$

since the summand grows as $O(d^2)$ over $O(N)$ terms. The total gate count per repetition is dominated by this SWAP overhead and is therefore $O(N^3)$.

On a 2D $\sqrt{N} \times \sqrt{N}$ grid topology, the Manhattan distance between qubits is on average $O(\sqrt{N})$. Routing a single pair therefore requires $O(\sqrt{N})$ SWAPs, i.e. $O(\sqrt{N})$ additional CX gates. Summing over all $\binom{N}{2} = O(N^2)$ pairs, the total SWAP overhead becomes

$$G_{\text{SWAP}}^{\text{2D}}(N) = O\left(N^2 \cdot \sqrt{N}\right) = O(N^{5/2}). \tag{3.36}$$

In summary, neutral atoms retain a routing advantage over fixed-topology superconducting devices due to higher connectivity and, when available, 3D geometry: $O(N^{7/3})$ on a 3D lattice versus $O(N^{5/2})$ on 2D grids and $O(N^3)$ on linear chains.

Note that the limited coherence times of current neutral atom devices (on the order of 10 μs) impose a practical limit on the number of gates that can be executed before decoherence dominates, and therefore on the maximum N for which the QKE pipeline can be successfully implemented. This means that scaling QKE beyond a few qubits requires either faster gate operations, longer coherence times, or error-mitigation strategies.

3.4 Summary

In this chapter it was shown that, using neutral atom devices, it is not only possible to work completely with the gate model, but the advantage given by the arbitrary arrangement of atoms in space can be exploited for optimizing highly entangling circuits, showing this advantage on the Quantum Kernel Estimation technique that is traditionally implemented on superconducting devices. A general method for obtaining single-qubit and multi-qubit gates was formalized, generalizing the technique proposed in [41], and it was then used for implementing the QKE algorithm on a neutral atom device. A mean accuracy of $79.8\% \pm 8.8\%$ with 3 features was

reached across 30 independent dataset seeds, significantly outperforming a classical RBF-kernel baseline ($52.8\% \pm 10.2\%$). A paired t-test on the per-seed accuracy difference ($\bar{\gamma} = +27.0\%$, $\sigma_\gamma = 13.4\%$) yields $p = 6.33 \times 10^{-12}$. Additionally, a sampling noise analysis on a fixed dataset confirmed that the kernel estimation at $N_{\text{shots}} = 1000$ is sufficiently stable, with a QSVM accuracy standard deviation of only ≈ 0.04. It was also shown how to geometrically exploit the possibility to arbitrarily arrange the atoms in space when the number of qubits is high, in order to maximally reduce the SWAP gates needed for allowing entanglement. This chapter aims to serve as a reference for future works that either wish to use the gate model on neutral atom quantum computers for any kind of problem or to further explore the possibilities that this technology allows in the field of QML, and to provide an object of discussion and further exploration on the possible utility of quantum computing for real classification problems.

Within the dissertation narrative, this chapter converts the theoretical framework of Chapter 2 into a concrete hardware-aware learning pipeline. The next chapter accompanies this contribution by shifting the focus from connectivity-limited expressivity to low-latency control and calibration in superconducting platforms.

Chapter 4

Machine Learning for Quantum Control of Superconducting Qubits

4.1 Introduction

Building on the hardware-aware implementation perspective introduced in Chapter 3, this chapter analyzes the complementary challenge of real-time control in superconducting systems. The realization of quantum computational advantage hinges on the ability to execute algorithms with a degree of efficiency that surpasses classical limits [49]. Among the leading hardware candidates, superconducting transmon qubits have emerged as a particularly robust platform, offering a viable pathway toward fault-tolerant architectures [50–52]. However, as these systems scale, they face a critical "control bottleneck" imposed by the cryogenic environment. Specifically, the necessity of maintaining qubits at millikelvin temperatures within dilution refrigerators introduces severe constraints on communication latency and bandwidth between the quantum processor and the classical control systems located at room temperature.

This chapter investigates a novel machine learning (ML) framework designed to overcome these limitations by deploying control logic directly proximate to the qubits. By utilizing field-programmable logic, such as Field-Programmable Gate Arrays (FPGAs) and embedded FPGAs (eFPGAs), we can achieve high-performance hardware acceleration within the restricted power and space budgets of a refrigerator [53]. While standalone FPGAs offer post-deployment flexibility, eFPGAs integrated

into System-on-Chips (SoCs) provide a specialized path for reducing power consumption and optimizing the hardware-software interface for real-time quantum control.

4.1.1 Pulse-Level Control and the Role of Machine Learning

The control of transmon qubits is fundamentally an analog problem, where electromagnetic microwave pulses are tailored to drive specific unitary transformations, represented as rotations on the Bloch sphere. While any arbitrary rotation can be decomposed into a sequence of Z and X gates, transmon architectures allow for "virtual" Z rotations simply by shifting the phase of subsequent pulses [54]. Consequently, the primary challenge lies in the synthesis of high-fidelity $R_x(\theta)$ gates.

Traditional methods for pulse synthesis rely on resource-intensive simulations using tools such as `Juqbox`, `Qiskit Pulse`, or `QuTiP` [55–58]. While accurate, these solutions are typically executed on external workstations, creating a communication bottleneck that hinders scalability. Conversely, simpler alternatives like lookup tables suffer from interpolation errors and lack the flexibility required to adapt to hardware noise and drift.

To address these shortcomings, we propose a multi-stage ML-based approach for single-qubit gate synthesis. Our methodology begins with "bootstrapping" a neural network in a simulated environment, training it to infer optimal B-spline pulse coefficients based on full statevector information. This approach builds upon prior research [59] but extends it to address the practicalities of real-world hardware, where the full quantum state is not directly observable and system parameters, such as qubit anharmonicity, exhibit instability over thermal cycles.

4.1.2 Fidelity Estimation and Hardware Adaptation

A critical contribution of this chapter is the development of a feedback loop that allows the ML model to adapt to specific hardware characteristics. Since the fidelity of a gate is the primary metric for success, we implement an "Adapted Randomized Benchmarking" (ARB) algorithm, based on the principles of statistical fidelity estimation [60]. This enables the fine-tuning of gate parameters directly on the device, accounting for noise sources that are not captured in idealized simulations.

The techniques presented herein, while demonstrated on transmon-based systems, are designed with portability in mind. The integration of neural network inference with programmable logic holds potential for diverse quantum architectures, including high-fidelity control of superconducting cavities [61, 62], trapped-ion systems [63], and neutral-atom qubits [17]. Through the exploration of design spaces and resource requirements, this chapter establishes a framework for efficient, hardware-proximate quantum control in the NISQ era and beyond.

4.1.3 Prior work

A number of papers can be found in the literature proposing methods to calibrate quantum gates. For instance, [64] and [65] use variations of Quantum Process Tomography (QPT); however, the computational complexity of QPT makes it unusable for a large set of gates to be optimized. In [66], instead, the system is modeled with a Hamiltonian and the gates are numerically optimized, performing a simulation of the system. Such a method is certainly useful as a starting point, but cannot be implemented on its own on a real device, as the actual system is inherently more complicated than the model and there is no proposed method to adjust to the real qubits. Our method starts in a similar way using Juqbox for optimizing the gates during a numerical simulation of the system, but that is only used as a warm-up; the parameters are then fine-tuned considering physical variations that model the differences that occur at the passage from the model to the real system. Finally, there are also papers that apply RB to non-Clifford gates, such as [67], but they do not provide any statistical backing to the fidelity estimates.
This chapter builds on previous work by some of the authors of [59]. Our primary extensions here are to adapt the algorithm to use a cost function based on measurements from a real quantum computer. While we use simulated data in this chapter, we hide unobservable "truth values" (i.e., the wavefunction) from the simulation and only use the results of projective measurements of quantum states, as with real quantum hardware. We develop a benchmark for assessing the fidelity of our gate synthesis algorithm and estimate the training costs in shots on a quantum computer. The content of this chapter has been previously published by the author in [68].

4.2 Methods

4.2.1 Data generation: creating quantum control samples with `Juqbox`

Quantum control sample generation via `Juqbox` [55] is a cornerstone of our methodology, as it defines the parameters within which we validate our hypotheses. Our objective was to produce samples comprising a single input angle in the range of $-\pi$ to π, paired with output pulse coefficients of a predefined size (20 in this case). The qubit is controlled by a microwave control channel capacitively coupled to the qubit. we use quadratic B-splines as the basis functions to decompose the envelop of the microwave control drive with the wave frequency set to be the qubit frequency. The first/last 10 pulse coefficients are the B-spline coefficients of the real/imaginary part of the control. In this study, we assume the qubit anharmonicity to be 200 MHz unless otherwise specified. We set the pulse duration time to be 125 ns and the maximum pulse amplitude to be 20 MHz for optimal control. To implement the control pulse with specific hardware, the control pulse has to be converted to the voltage level of the control line. The conversion is typically determined by calibrating the quantum hardware [69, 70]. The calibration's details are highly specific to the experimental setup and are outside the scope of this chapter.

The generation of these samples was facilitated through a Jupyter Notebook script, which in turn invoked several `Julia` scripts for configuration and generation using `Juqbox`. Aiming for a high-fidelity threshold, we achieved a fidelity greater than 0.9999 (referred to as "four 9s" of Fidelity), signifying an excellent quantum state overlap. This level of precision, quantified by fidelity, highlights the data's precision and the system's effectiveness in maintaining the desired state.

The `Juqbox` mathematical model uses a seed value that alters the output pulse coefficients even when the input angle value remains unchanged. This creates a challenging dataset for model building due to large variations in outputs for neighboring input angles. To address this, we generate datasets using 100 seeds, each containing 4,096 input angles uniformly distributed from $-\pi$ to π, which helps reduce erratic variation as discussed in the next section. In addition to the general variation across different seeds, we observed that the pulse coefficients would invert at around -3.118 when initialized randomly. We addressed this by including a fixed

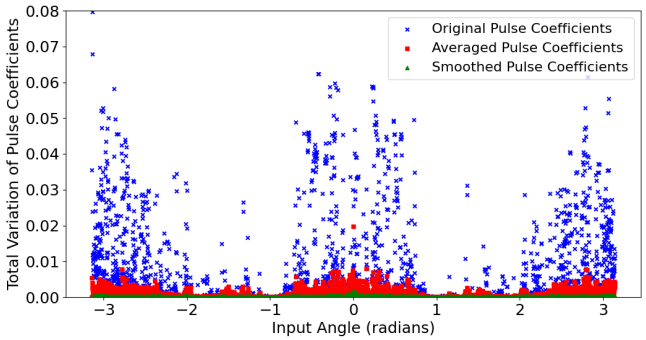


Fig. 4.1 Total Variation of Pulse Parameters Over Angle. This overlay plot compares the original, averaged, and smoothed datasets, highlighting changes in parameter stability across different processing stages.

positive or negative baseline initial pulse based on the sign of the rotation angle, in addition to the random initialization in the `Julia` script responsible for generating these pulses by optimal pulse control. This fixed baseline serves as an educated guess to force the optimized pulses to carry the same sign as the baseline. Empirical analysis determined that approximately 4,096 samples within the $-\pi$ to π range were ideal. Excess samples introduced unwanted noise, while too few samples reduced the model's ability to generalize.

Data analysis: refining and optimizing quantum control samples

Refining and optimizing data samples is crucial for enhancing the performance and efficiency of model development. The process began with the organization and improvement of data generated from the 100 varied seeds, with each one producing slightly varied outputs.

The uniformity of the data was improved by averaging the outcomes from all seeds, resulting in a single set of 4,096 samples. This averaging process is performed across the 100 seeds for each of the 4,096 input angles. Our objective is to mitigate the noise associated with any particular seed. For each input angle, the averaging can be expressed as:

$$A_i = \frac{s_{i1} + s_{i2} + s_{i3} + \cdots + s_{i100}}{100} \tag{4.1}$$

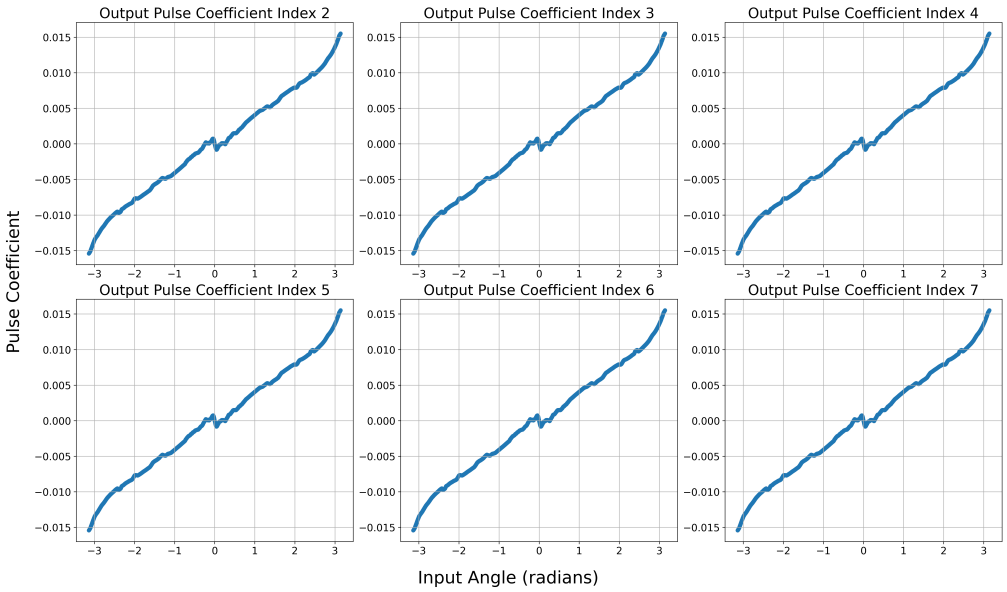


Fig. 4.2 Comparative subplots of output B-spline coefficients 2 to 7 out of 20 coefficients for X gates, demonstrating parallel trends. The six coefficients shown have similar values (y-axis) for a given X gate rotation angle (x-axis), and hence they can be replaced by the average value for data smoothing as explained in the main text.

where A_i is the averaged value for the i-th input angle, and s_{ij} represents the value from the j-th seed for the i-th input angle. This process is repeated for all 4,096 input angles.

This step was critical to eliminate outliers and anomalies inherent in any individual seed. The data was then smoothed by using a sort of convolution that works by averaging groups of 50 close samples and using this average for the middle sample. This method helps the values change gradually instead of suddenly, leading to much less variation in the data, as shown in Fig. 4.1. We further simplified our data for X gates by reducing the number of coefficients from 10 to 5. We observed that several coefficients exhibited minimal variation across different input angles, allowing us to replace them with their average values without significant loss of information. Figure 4.2 illustrates this pattern for X gates. Once the data was smoothed in this manner, it was divided into training, testing, and validation sets for use in subsequent stages. The script detailing these processes is available for reference.

Through these modifications, the dataset was condensed while maintaining four nines of fidelity. Despite a negligible decline in overall fidelity of the dataset

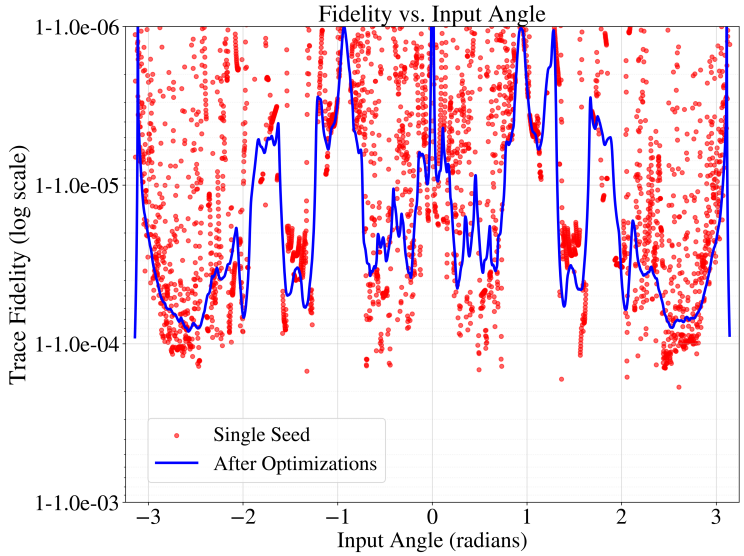


Fig. 4.3 Trace Fidelity comparison between a single seed and after dataset optimizations over varying input angles.

(< 0.00002), the fidelity for some individual angle measurements actually improved, particularly in previously worst-case scenarios, as evidenced by Fig. 4.3.

4.2.2 Model training: optimizing for efficiency and fidelity

For our model architecture, we opted for a multi-layer perceptron neural network due to the relatively straightforward nature of pulse coefficient prediction and its capability to efficiently translate to hardware. Our Keras-based neural network model's training process involves progressive stages. We begin with pre-training using 'ground truth' values from quantum simulations, helping the model recognize similarities in pulse coefficients. Subsequently, we refine the model by focusing on quantum state output fidelity, using infidelity (1 - fidelity) as the loss function. This stage incorporates quantum simulation directly into training but is slower due to the non-analytical nature of the infidelity loss function. These steps are detailed in Section 4.2.2.

We then make training quantization-aware (Section 4.2.3), translate the model for field programmable logic (Section 4.2.4), and fine-tune based on measurement-only information (Section 4.3.1).

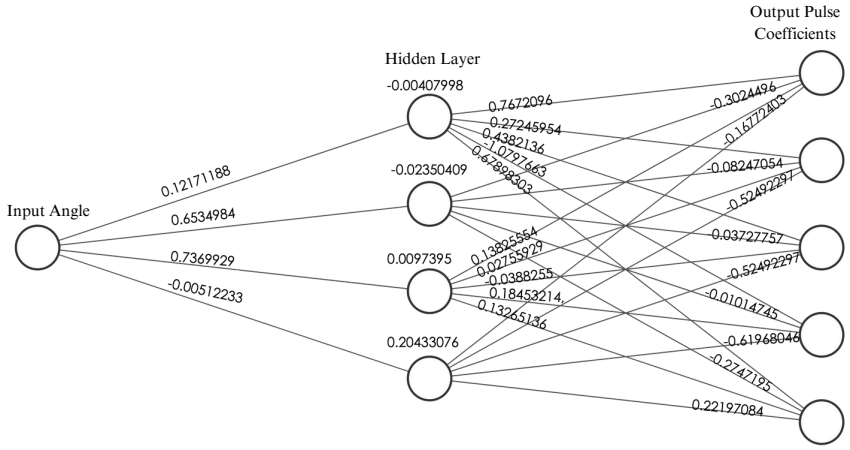


Fig. 4.4 Illustration of the smallest Keras model configuration with 33 parameters achieving four nines of fidelity.

Our focus is on creating a compact, efficient Keras model maintaining four nines of fidelity. This involves optimizing architecture, activation functions, learning rate, and loss function. The model is trained and evaluated using separate training and validation sets.

Pre-training involved 10,000 epochs with a 0.0001 learning rate, using MSE as the primary loss function. For fine-tuning, we employed an infidelity-based loss function, introducing controlled perturbations to the model's weights and recalculating infidelity. This process, detailed in Section 4.2.2, significantly improved performance but is considerably slower than MSE training.

Through these optimizations, we reduced model complexity from over 2000 to just 33 adjustable parameters. Fig. 4.4 illustrates the final model. This neural network model has a single input with a single hidden layer of size 4 and an output layer of size 5 which is reduced from the 20 pulse coefficients as explained earlier in Section 4.2.1.

Simulation-based training algorithm

The algorithm is built on a set of functions that are available in our software package [71]. See Algorithm 7.

- **Function** `infidelity_loss_parallelized(x, y_preds, y_orig):`
  - Description: Calls an external quantum simulation to compute infidelities in parallel.
  - Input: x - input data, *y_preds* - predicted values, *y_orig* - original values.
  - Output: $infidelities\_y\_preds$ - infidelities of *y_preds*, $infidelities\_y\_orig$ - infidelities of *y_orig*.
- **Function** `infid_grad(x, model, epsilon):`
  - Description: Applies a small epsilon perturbation to each weight/bias of the model and computes the gradients. The gradients are obtained by calculating the infidelities using infidelity_loss_parallelized, computing the difference before and after the perturbation, and dividing that by the perturbation epsilon.
  - Input: x - input data, *model* - trained model, $epsilon$ - small perturbation value.
  - Output: *gradients* - computed gradients, $infidelity\_original$ - original infidelity.
- **Function** `train_step(x_batch, y_batch):`
  - Description: Computes gradients and loss for the batch using infid_grad, updates model trainable variables using optimizer, updates loss metric to track loss.
  - Input: *x_batch* - batch of input data, *y_batch* - batch of output data.

4.2.3 Quantization aware training

Quantization-aware training (QAT) is critical in optimizing ML models for efficient deployment on hardware accelerators. This is particularly true for field-programmable logic, which is increasingly used in edge computing due to its reconfigurability, energy efficiency, and ability to perform parallel computations. However, constrained resources, such as limited memory and computational elements, necessitate deploying carefully tailored models. Thus, the primary motivation behind QAT

Algorithm 7 Simulation-based training

```
Load the best pre-trained model based on an MSE loss over pulse coefficients.
Initialize the optimizer and loss metric.
Set up ModelCheckpoint callback with the specified coefficients.
for epoch in training epochs do
    Reset loss metric.
    for batch in training set do
        Perform train_step with x_batch and y_batch.
    end for
    Update model checkpoint.
end for
```

is to reduce the precision of the weights and activations in neural networks from floating-point to fixed-point representations, thereby decreasing the model size and computational complexity. By training neural networks to be aware of quantization effects, it is possible to significantly mitigate the degradation in performance typically associated with more traditional techniques like post-training quantization.

In our work, we have adopted `QKeras` [72], an extension of the popular `Keras` library that mimics the behavior of fixed-point arithmetic as part of the training process. `QKeras` provides quantized versions of standard `Keras` layers (e.g., `QDense`, `QConv2D`) where the designer can specify the bit width for weights, biases, and activations directly in the layer definitions. During training, `QKeras` simulates the quantization process: the forward pass computes the layer outputs using quantized weights and activations, simulating the effects of fixed-point quantization. However, for the backward pass and weight updates, floating-point precision is typically used to maintain training stability and performance. Among the `QKeras` customizable parameters, we experimented with:

- **bits** to select the total fixed-point bit width (W) allocated for a layer. Our experiments indicate that diminishing this value to as low as 16 bits preserves our model fidelity.
- **integer** to select the number of bits (I) allocated to the integer portion of the fixed-point representation. Maintaining a minimum of 5 bits for the integer part and 11 bits ($W - I$) for the decimal part did not significantly diminish our model fidelity.

- **alpha** facilitates the simulation of Leaky ReLU functions. A Leaky ReLU introduces a nonzero gradient *alpha* for negative inputs. Introducing this slight slope for negative values helps keep neurons "alive" by ensuring they can still learn during the backpropagation process even when their inputs are negative. In our initial experiments, we adopted a value of 1.

- **qnoise_factor** determines the extent to which quantization noise is added to the weights and activations during the forward and backward passes of model training. The network learns to cope with the noise, leading to potentially better generalization and accuracy in the quantized model. As the value of noise increases, more quantization noise is added, simulating a higher degree of quantization effect. In our current study, we set this parameter to 1.

Finally, it is worth noting that we did not retrain the model from scratch in `QKeras`. Instead, we transferred the weights from the previously trained `Keras` model to the quantized `QKeras` model and then additionally ran quantization-aware training. This method ensured the model's fidelity while transitioning to a quantized representation.

4.2.4 Hardware translation: from quantized model to FPGA deployment

We adopted `hls4ml`, a Python open-source framework [73, 74], to co-design and translate our ML models into a hardware implementation while studying model accuracy, resource utilization, and inference latency. The `hls4ml` workflow begins with a floating-point model from a conventional ML framework, such as `TensorFlow` or `PyTorch`, or a quantized model from `QKeras`. Then, it translates the model into a C++ specification for the high-level synthesis (HLS) flow. HLS generates a hardware description at the register-transfer level for a more traditional synthesis and implementation flow targeting programmable logic as deployment hardware. Designers can leverage `hls4ml` to make quick design explorations by configuring the hardware implementation parallelism [75] and, thanks to the integration with `QKeras`, by also evaluating the impact of low-bit precision on model performance before finalizing the hardware implementation.

We iteratively translated the `QKeras` model into an HLS/C++ specification and hardware implementation to evaluate the resource utilization and model fidelity. In particular, we tuned the reuse factor parameter in `hls4ml`, which impacts parallelism, resource utilization, and performance. In our experience, a reuse factor of 20 has shown a good compromise in resource utilization. Moreover, we adjusted the accumulators' bit accuracy for each layer in fixed-point arithmetic. It is worth noting that `QKeras` does not provide the fine-tuning necessary for optimal hardware implementation. We observed a minimum bit width of around 22 to avoid performance degradation. At this point, `hls4ml` automatically translates the ML model into an HLS/C++ specification that can be simulated for fidelity assessment. This verification step is crucial for confirming that the translation has been successful and that the model specification is ready for the hardware implementation.

We leveraged graphical tools provided by the `hls4ml` framework to ensure translation correctness. For example, we use identity-line plots to compare the layer outputs. In such a plot, the diagonal line (the line of identity) represents perfect agreement between `QKeras` and HLS/C++ implementations. Points sitting precisely on the diagonal indicate that for every input of the layers, both implementations produce the same outputs. The goal is for all points to lie as close to the diagonal as possible, indicating that the two-layer implementations produce nearly identical results. Deviations from the diagonal suggest discrepancies between the outputs of the two systems and should be carefully analyzed.

We ran HLS and implementation targeting FlexLogix eFPGA, a reconfigurable fabric that offers efficient and flexible hardware acceleration solutions [76]. The results of our implementation showed a resource utilization of 693 LUTs, 709 FFs, and 2 DSPs, with a latency of 420ns. The resource utilization is minimal even for the smallest configuration of FlexLogix eFPGAs.

4.2.5 Adapted Randomized Benchmarking (ARB)

The standard approach for estimating quantum gate fidelity in the literature is Randomized Benchmarking (RB), which applies to Clifford gates. Here we provide an adapted algorithm for non-Clifford gates, updating the ideas behind RB, and provide a method for computing confidence intervals for the fidelity.

Assuming that only $R_x(\theta)$ gates are to be tested, let $\mathscr{G}$ be the set of angles that we want to test and $|\mathscr{G}|$ its cardinality. For each angle θ_i, the corresponding imperfect gate is $\hat{R}_x(\theta_i)$. Before starting the algorithm, there are two required steps:

- Preliminary step 1: Choose a set of sequence lengths $M = \{m_1, \ldots, m_M\}$. Each sequence length defines the number of gates that are consecutively applied to the initial state, where the last gate will be the inverse of the sum of the previous gates. The denser M is the better the estimation will be. We empirically observed 2 to 100 is a reasonable range.
- Preliminary step 2: Choose K random sequences. Each k^{th} sequence will be m gates long, depending on the current sequence length m. The first $m-1$ gates are uniformly sampled (possibly with repetition) from the set to be tested. For each new sequence, a new sampling is performed, resulting in (typically) unique sequences. We empirically observed a K on the order of hundreds to be performant.

With the preliminaries complete the Adapted Randomized Benchmarking algorithm for non-Clifford gates is Algorithm 8. Although a formal relation between gate fidelity and the metric estimated by ARB is not established, we will show numerically that the estimation successfully characterizes the error in rotation angles and can be used as the cost function to train a neural network to correct the error.

Testing ARB with direct unitaries

As an initial demonstration, consider a set of perturbed gates without an underlying physical simulation. Here, the interval $[-\pi, \pi]$ is divided in 1,000 angles and each angle θ_i is perturbed by adding noise term drawn from a Gaussian distribution, $\hat{\theta}_i = \theta_i + \mathscr{N}(0, \sigma)$. We perform ARB by doing a sequence of rotations using these perturbed angles, but we modify the procedure and set the final rotation as the inverse formed from the sum of the exact angles instead the noisy angles. While ARB shouldn't use exact gates, here it is necessary because if we use the sum of the perturbed angles for the inverse, we would obtain exactly the initial state. An alternative would have been to use the sum of the same angles and add another Gaussian noise perturbation.

Algorithm 8 Adapted Randomized Benchmarking for non-Clifford gates

for $m \in M$ **do**
  for $k \in 1 \dots K$ **do**
    Uniformly sample a random sequence of numbers $s_k = s_{1_k} s_{2_k} \dots s_{m-1_k}$, where $s_{i_k} \in \{1, \dots, |\mathscr{G}|\}$. Each s_{i_k} corresponds to a gate $\hat{R}_x(\theta_{s_{i_k}})$.
    Prepare the initial state $|0\rangle$.
    Apply the sequence of gates $\hat{R}_x(\theta_{s_{1_k}}) \dots \hat{R}_x(\theta_{s_{m-1_k}}) \hat{R}_x(-\sum_{i=1}^{m-1} \theta_{s_{i_k}})$.
    Perform N repetitions and measurements in the Pauli-Z basis. The probability of outcome 0 is estimated as

$$\hat{p}_{k,m} = \frac{\#zeros}{N},$$

    with standard error from the Binomial distribution,

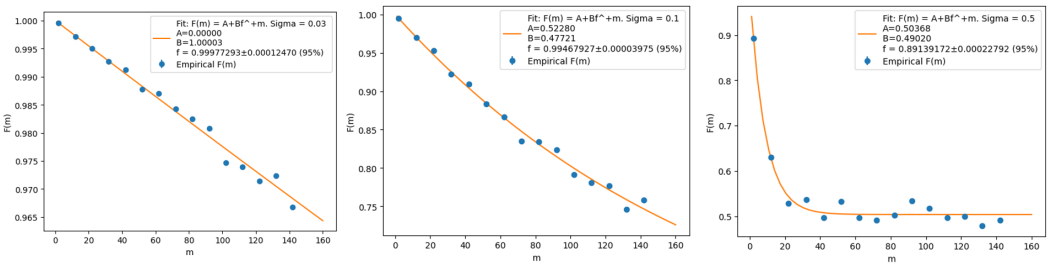
$$SE_{\hat{p}_{k,m}} = \sqrt{\frac{\hat{p}_{k,m}(1-\hat{p}_{k,m})}{N}}$$

  end for
  Calculate the average of $\hat{p}_m$, $\text{avg}_{\hat{p},m}$, over the K sequences. Its standard error will be

$$SE_{\text{avg}_{\hat{p},m}} = \frac{\sqrt{\sum SE^2_{\hat{p}_{k,m}}}}{K}.$$

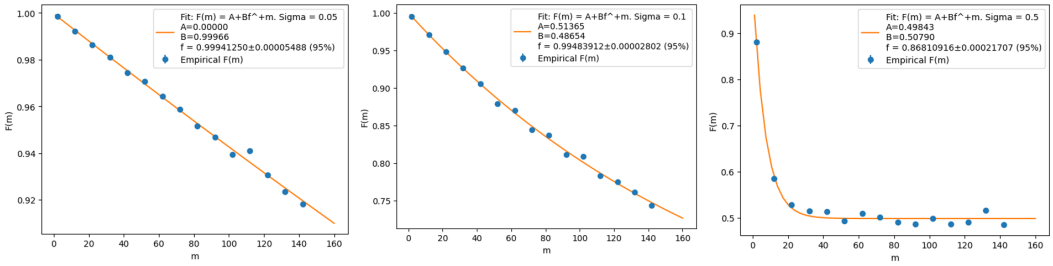
  The current estimated fidelity with m gates is therefore $\mathscr{F}_m = \text{avg}_{\hat{p},m}$, with standard error $err_m = SE_{\text{avg}_{\hat{p},m}}$.
end for
After calculating the quantities for each m, fit $\bar{\mathscr{F}}_m = A + Bf^m$. Here, A, B, f are the coefficients to be found, m is the independent variable and $\bar{\mathscr{F}}_m$ the dependent variable. The err_m's are used as uncertainties to obtain the resulting uncertainties on A, B and f. The bounds should be $0 \leq A \leq 1$, $0 \leq B \leq 1$, $0 \leq f \leq 1$.
We now have an estimate $\hat{f}$ for the single-gate fidelity from the fit, along with a covariance matrix (for example, this is provided automatically when using `scipy.optimize.curve_fit` for the three parameters, $\Sigma \in \mathbb{R}^{3,3}$). The third diagonal entry (if f was used as third parameter for the fitting) will contain $SE^2_{\hat{f}}$.
Finally, perform a 2-tailed Student's t-test, where the number of degrees of freedom is equal to $|M| - 3$ (3 is the number of parameters), and setting $\alpha = 0.05$ to obtain a 95% confidence interval for f, corresponding to

$$\left[\hat{f} - t_\alpha \sqrt{\Sigma(3,3)}, \hat{f} + t_\alpha \sqrt{\Sigma(3,3)}\right]$$



$\sigma = 0.05\ rad$ $\sigma = 0.1\ rad$ $\sigma = 0.5\ rad$

Fig. 4.5 The first three experiments with artificially perturbed gates (K=500).



$\sigma = 0.05\ rad$ $\sigma = 0.1\ rad$ $\sigma = 0.5\ rad$

Fig. 4.6 The last three experiments with artificially perturbed gates (K=1,000).

We perform six experiments, varying K and σ. In all of them we set $N = 1,000$ and $M = \texttt{range}(2, 150, 10)$. The first three have $K = 500$, and σ equal to 0.05 rad, then 0.1 rad, and then finally 0.5 rad. The results are shown in Fig. 4.5. The last three have $K = 1,000$, and the same σ values. These results are shown in Fig. 4.6.

Using ARB for testing a pre-trained neural network on different physical conditions

We initially trained a neural network using a quantum simulation of a qubit with 0 guard levels instead of 1 guard level. The network was trained using the Mean-Squared Error (MSE) between the pulse coefficients generated by Juqbox for the angles in the training set and the corresponding pulse coefficients inferred by the NN as a loss function. We used ARB to measure the fidelity using a simulation with zero guard levels and identical anharmonicity, finding a fidelity that remained at the level of four nines.

The actual physical system has an infinite number of guard levels, and the occupancy probability falls as the level increases. Additionally, temperature cycles

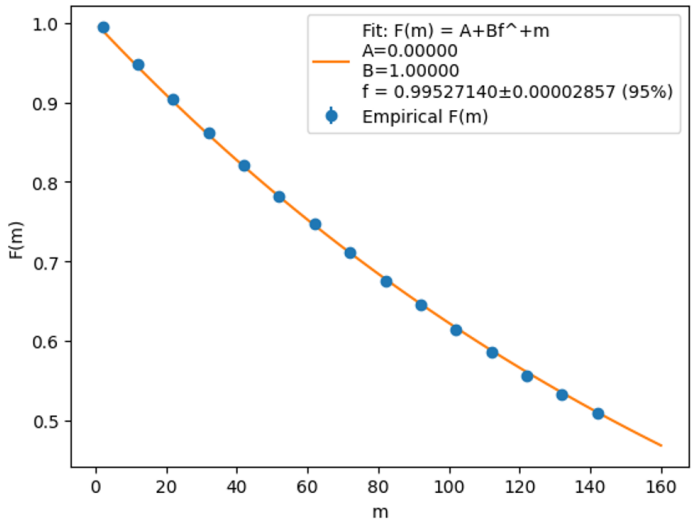


Fig. 4.7 Fidelity estimated with ARB when the anharmonicity is unchanged.

such as heating up the system and then cooling it down again impact some physical parameters, such as the qubit anharmonicity. Therefore, we expect the performance of the NN to decrease over time, because it was trained on pulses that `Juqbox` generated utilizing different physical parameters.

We examined this network using ARB to measure the fidelity of the pulses generated by the pre-trained neural network, first setting the anharmonicity to 200 MHz, then multiplying it by 10. In both cases, we added one guard level to the simulation. The primary purpose of this change is to simulate a mis-modeling of the physics parameters that describe the device. This allows for some estimation of the impact of imperfect device simulation on NN training, for example, in the case of using simulation to bootstrap a model for fine-tuning on hardware. It also provides a proxy for understanding the impact of drifts in the device noise characteristics, which is an important issue for keeping a NN model well-tuned.

As shown in Fig. 4.7, introducing one guard level causes the fidelity decrease by two orders of magnitude. In Fig. 4.8, however, it is evident that multiplying the anharmonicity by 10 has the effect of compensating for this by moving the guard level further from the two essential levels.

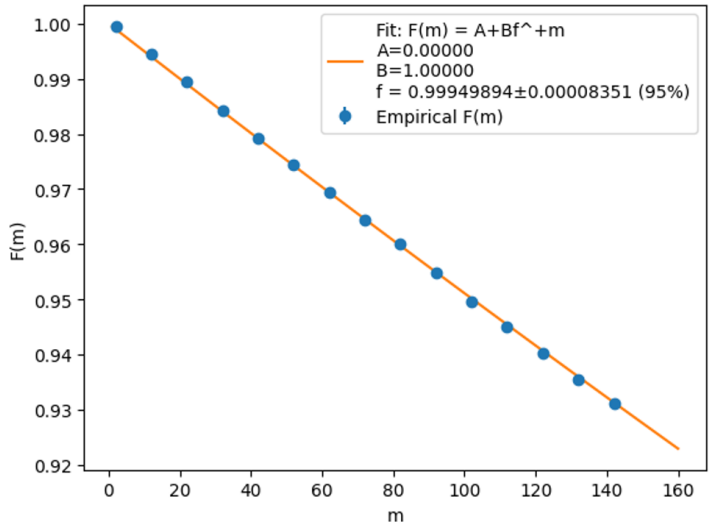


Fig. 4.8 Fidelity estimated with ARB when the anharmonicity is x10 the one used during training.

4.3 Results

4.3.1 Using ARB for fine-tuning

Our strategy for adapting ARB to a realistic scenario is to take the pre-trained network (as described in Section 4.2.2) and to fine-tune it to a different configuration, using ARB instead of the MSE with "ideal" pulse coefficients for the NN loss.

In particular, a simulated scenario with G=1 guarded levels is considered, with a new anharmonicity equal to 2 GHz, which is 10 times the one that the network was trained on. Notice that this higher anharmonicity compensates for the new guarded level, as it provides a higher frequency separation of the two essential levels $|0\rangle, |1\rangle$ from the guarded level $|2\rangle$.

4.3.2 Code structure

The NN model was trained in `Python` using `TensorFlow`. We further use `Python` to do the fine-tuning and the inference. At each training epoch, we consider a set of angles from $-\pi$ to π and infer a list of 20 pulse coefficients for each of these angles. The 20 coefficients are the B-spline coefficients of the control pulse

described in section 4.2.1. Then, for each angle, its corresponding list of coefficients is used to make the system evolve according to the corresponding pulse and a unitary corresponding to that pulse is obtained. Doing this for all the angles, a set of unitaries is obtained corresponding to the gates that the network inferred for all the angles. At this point, ARB is run to measure the fidelity of these gates to the theoretical ones, and the weights of the network are updated accordingly via gradient descent.

A technical complication arises from the fact that Juqbox is implemented in `Julia`, so we transfer data between applications using pipes, which are a type of Inter-Process Communication (IPC), exchanging data in JSON format.

Another technical consideration is that the unitaries obtained with `Juqbox` result from a numerical integration of differential equations, and due to limited precision may not be unitary. Therefore, as an additional step we renormalize them using their 2-norm.

Optimization algorithm for training

Our ultimate goal is to deploy this network on an embedded device, so we invested substantial effort in minimizing the footprint of the NN. The "small" version of the network has 8 dense layers (fully connected layers with ReLu activation function), resulting in 760 parameters (weights and offsets). Because our loss is calculated via IPC program calls, `TensorFlowautograd` cannot be used, and gradients must be computed by hand. Exact gradient computation would require calculation of the loss varying each parameter. By doing this for all the parameters we would estimate the gradient. However, this results in the calculation of $760 * 2$ losses for each epoch, where each loss calculation involves a number of simulations equal to the number of angles used in training, which can be in the order of thousands. While accurate this procedure is unacceptably slow.

Another method, proven to statistically converge to the same solution, is Simulatenous Perturbation Stochastic Approximation (SPSA) [77] (Fig. 4.9). Here, all network parameters are perturbed at once, adding the same ε to all of them but with random sign. Specifically, considering a function $f(\vec{\theta})$ of which we want to estimate

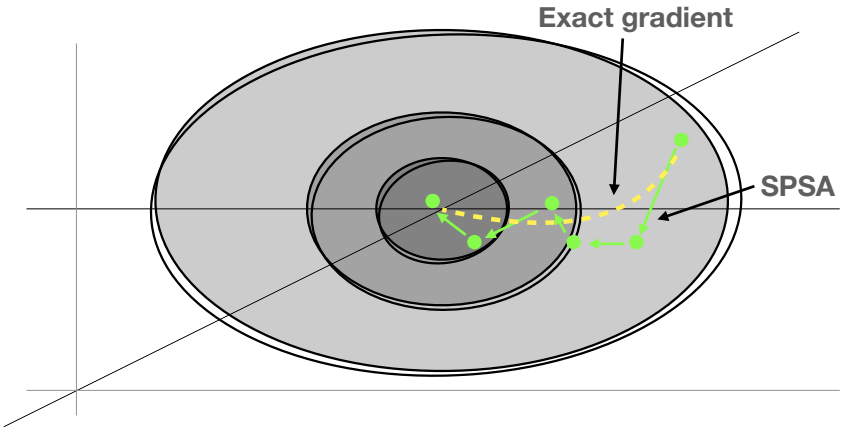


Fig. 4.9 An illustration showing how SPSA works compared with standard gradient descent (figure inspired by [3]).

the gradient, we perform

$$\nabla f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \dots \\ \frac{\partial f}{\partial \theta_n} \end{bmatrix} \approx \frac{f(\vec{\theta} + \varepsilon\vec{\Delta}) - f(\vec{\theta})}{\varepsilon} \vec{\Delta}^{-1} \tag{4.2}$$

where $\vec{\Delta} \in \{+1, -1\}^n$ and, in this case, $\vec{\Delta}^{-1} = \vec{\Delta}$ being its element-wise reciprocal. The advantage of SPSA is that it only requires 2 evaluations per epoch, one with the perturbed parameters and one with the unvaried ones, independent of the network size.

During training, there are actually two hyperparameters that determine the "aggressiveness" of gradient descent. One is the ε by which the gradient is estimated, and the other is the learning rate α, where $\vec{\theta}_{k+1} = \vec{\theta}_k - \alpha\hat{\nabla} f(\vec{\theta}_k)$.

Training set construction

There are multiple ways to construct the training dataset. Network generalization is generally enhanced by a larger number of angles used for training. We empirically observed 500 angles to be a good starting point. Our first experiment used a training set with 500 uniformly sampled angles, with a similar construction for the validation dataset with 100 angles. Note that the sampling procedure generates spacing that is generally non-uniform. As a form of regularization, then, the training set was resampled at each epoch.

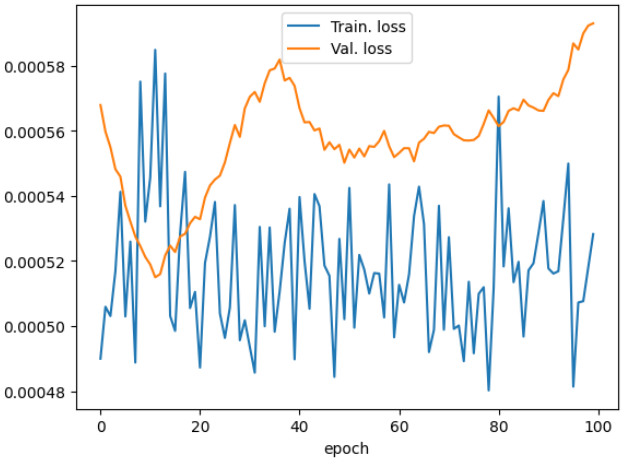


Fig. 4.10 Loss function during training for the first experiment. Blue = training loss, orange = validation loss.

Then, for a second experiment, we trained a version of the network with fixed intervals between angles without re-sampling. The training set was further divided into 10 batches. The validation set was again uniformly sampled. Finally, $\alpha = \varepsilon = 10^{-6}$ were used for both experiments.

Results

The results of the first experiment are shown in Fig. 4.10. We found re-sampling to construct the training set at each epoch made the learning unstable for this problem. In Fig. 4.11, we show the results of the second approach and find better results.

4.4 Discussion

From the results, we see that many epochs are needed to achieve a consistent improvement in accuracy. It is likely that, if the training is uncontrolled, the loss will at some point stabilize or start increasing again. For this reason, an early stopping mechanism is necessary, and some kind of regularization could be useful to avoid overfitting. The loss function is highly non-convex, so it is very hard to achieve its global minimum, and a hyperparameter tuning process can help reach the best possible local minima. A larger neural network may offer better performance, but at the cost of exceeding resources on an embedded platform.

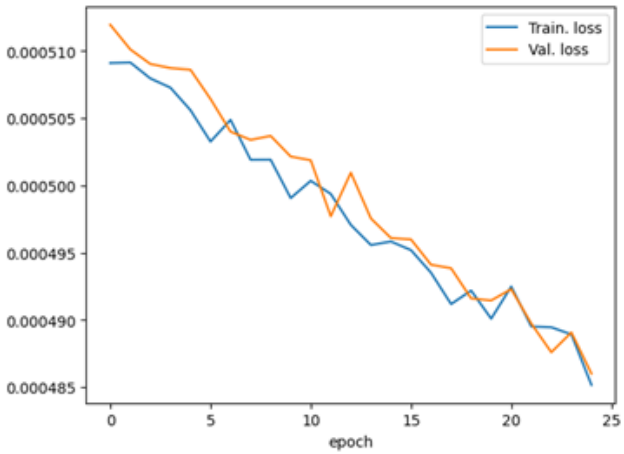


Fig. 4.11 Loss function during training for the second experiment.

Another potential improvement would be to modify the architecture of the network to include the qubit anharmonicity among the inputs, together with the rotation angle. This may reduce the need to retrain the network each time the anharmonicity changes, if it could reliably be trained once considering a set of possible anharmonicities.

4.4.1 Scaling of the control overhead

We analyze how the embedded-ML control strategy scales as the number of qubits Q on the processor increases.

Inference cost

The neural network developed in this chapter (Section 4.2.2, Fig. 4.4) has a fixed architecture: 1 input neuron, a single hidden layer of width 4, and 5 output neurons, totaling 33 trainable parameters. Its forward pass requires a fixed number of multiply-accumulate (MAC) operations:

$$\text{MACs} = (1 \times 4 + 4) + (4 \times 5 + 5) = 33, \tag{4.3}$$

where each term accounts for the weights and biases of one layer. Crucially, this architecture maps one rotation angle θ to the B-spline coefficients for one qubit; it

does not take the qubit index or any multi-qubit coupling as input. Therefore, the per-gate inference cost is a constant independent of Q.

When the processor contains Q qubits, each requiring its own R_x inference, the total computational work for a full inference pass is

$$C_{\text{inf}}(Q) = Q \times 33 \text{ MACs} = O(Q). \tag{4.4}$$

On the deployed eFPGA (Section 4.2.4), each inference completes in 420 ns. Due to the compactness of the model, the total work for Q qubits is $O(Q)$ MACs; however, the execution time depends on the deployment topology, which this chapter does not prescribe. If a dedicated eFPGA instance (or core) is provisioned per qubit, all Q inferences run in parallel and the latency remains 420 ns regardless of Q, i.e. $O(1)$. If a single shared eFPGA serves all qubits sequentially, the latency becomes $420Q$ ns, i.e. $O(Q)$; even in this worst case it stays well below the microsecond timescale of a single physical gate for processors with $Q \sim 10^3$. The minimal resource utilization suggests that replication is feasible, but the actual multi-qubit control architecture is beyond the scope of this chapter.

ARB retraining cost

The Adapted Randomized Benchmarking procedure (Algorithm 8) estimates fidelity by executing $|M|$ sequence lengths $\times$ K random sequences $\times$ N_{shots} repetitions per qubit. The shot budget per qubit is therefore

$$S_{\text{ARB}} = |M| \cdot K \cdot N_{\text{shots}}, \tag{4.5}$$

which is a constant that depends only on the desired estimation precision, not on Q, obtaining a computational cost of $O(1)$ with respect to the number of qubits. Over E training epochs and Q qubits, each qubit requires a total of $E \cdot S_{\text{ARB}}$ shots, giving

$$C_{\text{retrain}}(Q) = Q \cdot E \cdot S_{\text{ARB}} = O(Q) \tag{4.6}$$

if all qubits are served sequentially, otherwise it is still $O(1)$.

This linear scaling holds as long as each qubit is calibrated independently, i.e., when the single-qubit gate error on qubit i does not depend on the pulse applied to qubit $j \neq i$. On current superconducting processors, this assumption is valid for

well-isolated transmons but may break down at high qubit densities due to microwave crosstalk or two-level coupling. In such cases, a joint n-qubit characterization would be needed, and the cost would depend on the crosstalk graph structure.

The shot budget $S_{\mathrm{ARB}} = |M| \cdot K \cdot N_{\mathrm{shots}}$ is a constant with respect to Q, but its absolute magnitude determines the statistical precision of the fidelity estimate $\hat{f}$. To relate the parameters to a target precision, let ε be the desired radius of the 95% confidence interval on $\hat{f}$ (e.g., $\varepsilon = 10^{-4}$ means we can resolve fidelity differences of $0.0001 = 0.01\%$). It can be proven that the number of shots S_{ARB} scales as $O\left(\frac{1}{\varepsilon^2}\right)$, coming from the Cramér-Rao bound for Bernoulli sampling.

4.5 Summary

In conclusion, Adapted Randomized Benchmarking is a good strategy for optimizing neural networks to work with varying physical conditions, making it possible to potentially achieve a high fidelity arbitrary rotation gate on hardware platforms. Because this chapter is based on a simulation of a quantum device, we cannot guarantee the observed results will translate to hardware. However, we provide a complete workflow for real quantum hardware.

While we have not demonstrated the technique on a real hardware platform, by utilizing varying simulation configurations without utilizing the underlying knowledge of those configurations for tuning, our results suggest these techniques will translate to hardware. However, these techniques are not likely to be shot-efficient for superconducting transmons if noise drifts on a platform are significant, as re-training the NN will be time-consuming relative to the strategy of calibrating fixed angle gates and composing arbitrary rotations by utilizing virtual Z's. This technique will be more useful for platforms that do not have the benefit of a "free" and very high-fidelity rotation gate.

In the broader dissertation flow, this chapter complements Chapter 3: there the main bottleneck is entangling-connectivity overhead, whereas here the bottleneck is low-latency control under cryogenic constraints. The next chapter (Chapter 5) completes this progression by focusing on measurement robustness and state-preservation dynamics, providing the final technical component before the overall synthesis in the Conclusions.

Chapter 5

Simulating quantum dynamics of single-photon experiments

5.1 Introduction

While the preceding chapters of this dissertation have primarily focused on the development and optimization of QML algorithms, ranging from the theoretical landscape in Chapter 2 to the hardware-specific implementations on neutral atoms and superconducting qubits in Chapters 3 and 4, the broader utility of quantum platforms extends significantly into the realm of quantum dynamics simulation. Indeed, the ability to simulate complex quantum systems is often cited as the inaugural motivation for the field, providing a foundational benchmark for assessing quantum advantage. Although the present study deviates from the explicit application of machine learning, it addresses a critical prerequisite for any robust QML framework: the precise characterization of measurement paradigms and the preservation of quantum states under experimental conditions.

In this chapter, we investigate the practicalities of quantum theoretical advantages by performing a high-fidelity simulation of a single-photon laboratory experiment [78]. This research is particularly relevant as it explores the boundaries of wavefunction interaction, a core component of the state preparation and readout phases in any NISQ algorithm. The study focuses on the distinction between projective and protective measurements. While traditional projective measurements result in the immediate collapse of the wavefunction, the protective approach utilizes a sequence

of weak interactions, modeled here through a pair of birefringent crystals, coupled with a specific protective mechanism. This mechanism involves repeated projections onto the initial polarization state, in a regime often associated with the **Quantum Zeno Effect (QZE)**. The simulation shows that this strategy can increase the survival probability of photons over a finite range of steps, making it a practically useful protective mechanism for state preservation.

By implementing this experiment via a gate-based quantum circuit model on classical hardware, we aim to achieve a dual objective that complements the hardware-centric analyses of Chapters 3 and 4. First, we provide a rigorous comparison between numerical results and theoretical expectations, validating the robustness of the circuit-model approach for photonic systems. Second, this simulation serves as a benchmark to assess the computational efficiency of classical systems in emulating quantum dynamics as system complexity increases. Ultimately, this chapter offers a detailed perspective on how a noise-free quantum device would behave under these experimental conditions, bridging the gap between abstract quantum theory and physical laboratory implementation while providing insights into state preservation and protective-measurement-based stabilization that are essential for future high-fidelity QML architectures. In this sense, the simulation should be interpreted as evidence-generating for future quantum-advantage studies, not as a direct proof of quantum advantage. The content of this chapter has been previously published by the author in [79].

5.2 Experimental setup

Figure 5.1 shows the experimental setup. Heralded single photons are produced by type I parametric down-conversion in a $LiIO_3$ nonlinear crystal a, producing idler and signal photons. The pump beam is obtained by second-harmonic generation (SHG) of a Ti:Sapphire mode-locked laser output. The idler photons are fiber-coupled and detected by means of a silicon single-photon avalanche diode (SPAD I), sending a trigger pulse to the signal-photon detection system. The signal photons are fiber coupled and the photon beam is collimated as a Gaussian beam that passes through a half-wave plate (HWP) b, a collimating lens c, a polarizing beam splitter (PBS) d, and a HWP e, before entering a sequence of repeating units comprising of a pair of birefringent crystals, i and j (see the appendix for their details) and a polarizer f.

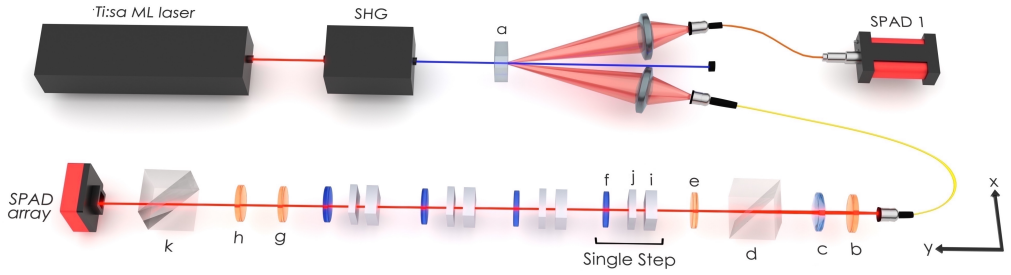


Fig. 5.1 HWP b is used to adjust the laser beam power. A PBS is placed before HWP b (not shown) to achieve this. The operators for HWP e, g and h are $W_{\pi/8}$, $W_{3\pi/8}$ and $W_{\pi/4}$, respectively (see the appendix). For the *protective case* the polarizer f in each step is used and for the $non-protective\ case$, it is removed. The purpose of HWP h is just to swap the reflected and transmitted component of the photon wave function.

The photons then pass through two more HWP, g and h, thereafter passing through a PBS k before being detected by a SPAD array detector. The horizontally polarized component is transmitted while the vertical one is reflected by both PBS d and k. The polarizer f axis is oriented at 45° with the $x-$axis. In the next section, we deduce the state of a photon before being detected.

5.3 Theoretical results

Let the wave function of the photon before HWP e, be $|i\rangle = |H\rangle\,|\psi\rangle\,|\tilde{y}\rangle$, where $|H\rangle$ is the linear polarization state of the photon, $|\tilde{y}\rangle$ implies that the photon is propagating in the $y-$direction and $|\psi\rangle$ represents the Gaussian mode of the photon beam, given by

$$|\psi\rangle = \int_{-\infty}^{\infty} dx\,|x\rangle\,\langle x|\psi\rangle \tag{5.1}$$

where

$$\langle x|\psi\rangle = \psi(x) = \frac{1}{(2\pi\sigma^2)^{1/4}} e^{-x^2/4\sigma^2} \tag{5.2}$$

We first describe the (weak) interaction of the photon with the birefringent crystals. This is modeled using the von Neumann indirect measurement protocol

[80–82]. Assuming that the photon with wave function $|H\rangle|\psi\rangle$ interacted with the crystal i (see the appendix for details) from time t_0 to t, its wave function after it exits the crystal will be $U(t,t_0)|H\rangle|\psi\rangle$, where $U(t,t_0) = e^{-\frac{i}{\hbar}\int_{t_0}^{t} H_i(t')dt'}$ with the condition that the interaction Hamiltonian H_i commutes with itself during the interaction, i.e., $[H_i(t), H_i(t')] = 0$. H_i for our case can be taken as [83] $H_i = \gamma\hat{A}\otimes\hat{p}$, γ being the coupling constant, $\hat{A} = |H\rangle\langle H|$ and $\hat{p} = -i\hbar\frac{\partial}{\partial x}$ being the momentum operator. As H_i does not depend on time, therefore $U = e^{-\frac{i}{\hbar}g\hat{A}\otimes\hat{p}}$, where $g = \gamma(t-t_0)$ is the coupling strength. The wave function of the photon after it exits the birefringent crystal will be

$$\begin{aligned}
U|H\rangle\otimes|\psi\rangle &= e^{-\frac{i}{\hbar}g|H\rangle\langle H|\otimes\hat{p}}|H\rangle\otimes|\psi\rangle \\
&= \left[\mathbb{1} - \frac{i}{\hbar}g|H\rangle\langle H|\otimes\hat{p} + \frac{i^2}{2\hbar^2}g^2|H\rangle\langle H|\otimes\hat{p}^2 - ...\right]|H\rangle\otimes|\psi\rangle \\
&= \left[|H\rangle - \frac{i}{\hbar}g|H\rangle\otimes\hat{p} + \frac{i^2}{2\hbar^2}g^2|H\rangle\otimes\hat{p}^2 - ...\right]|\psi\rangle \\
&= |H\rangle\otimes e^{-\frac{i}{\hbar}g\hat{p}}|\psi\rangle \\
&= |H\rangle\otimes|\psi\rangle_{x-g}
\end{aligned} \tag{5.3}$$

In the last step, $e^{-\frac{i}{\hbar}g\hat{p}}$ acts as a translation operator [84] on the wave function $|\psi\rangle$. Its effect is to transform the $\psi(x)$ amplitudes into $\psi(x-g)$, so that

$$|\psi\rangle_{x-g} = \int_{-\infty}^{\infty} dx\ \psi(x-g)|x\rangle$$

Crystal j is used only for phase compensation, as the $o-$ray and $e-$ray in crystal i follow different paths. The vertical polarization state $|V\rangle$ would pass through the birefringent crystals unchanged.When the polarizer f is used in each step, we call it *protective case* and if it is not used in each step, we call it $non-protective\ case$. If the fast axis of HWPs e, g and h are inclined at α, $3\pi/8$ and $\pi/4$, respectively, the wave function of the exiting photon (after PSB k) for the protective case can be obtained as

$$|f\rangle \propto \frac{1}{2^{n-1/2}}\left[\sum_{k=0}^{n}|\psi\rangle_{x-kg}\left\{\binom{n-1}{k}\sin 2\alpha + \binom{n-1}{k-1}\cos 2\alpha\right\}\right]|H\rangle|\tilde{y}\rangle \tag{5.4}$$

where n is the number of steps and $\binom{n}{k} = 0$ if $k < 0$ or $k > n$. The survival probability of the exiting photon (detected by the SPAD array) will be

$$P(n) = \frac{1}{2^{2n-1}} \int_{-\infty}^{\infty} \left[\sum_{k=0}^{n} \psi(x-kg) \left\{ \binom{n-1}{k} \sin 2\alpha + \binom{n-1}{k-1} \cos 2\alpha \right\} \right]^2 dx \quad (5.5)$$

For $non-protective\ case$, the wave function of the exiting photon will be

$$|f'\rangle = \frac{\sin 2\alpha\, |\psi\rangle_x + \cos 2\alpha\, |\psi\rangle_{x-ng}}{\sqrt{2}} |H\rangle\, |\tilde{y}\rangle + \frac{\sin 2\alpha\, |\psi\rangle_{y-\ell} - \cos 2\alpha\, |\psi\rangle_{y-\ell+ng}}{\sqrt{2}} |V\rangle\, |-\tilde{x}\rangle \quad (5.6)$$

where ℓ is the linear distance traveled by the photon beam, depending on where the origin is chosen along the input beam. For example, if the origin is chosen where the maximum of the beam's cross-sectional intensity distribution hits the coated surface of PBS d (i.e., the interface of the two right-angle prisms which are cemented together), and if the $x-$axis is taken along the line parallel to the incident beam, then ℓ would be the distance between the origin and the point where the $x-$axis intersects the coated surface of PBS k. The probabilities of detecting a photon transmitted and reflected by PBS k will, respectively, be

$$P'_T(n) = \frac{1}{2} \int_{-\infty}^{\infty} \left[\sin 2\alpha \psi(x) + \cos 2\alpha \psi(x-ng) \right]^2 dx \quad (5.7)$$

$$P'_R(n) = \frac{1}{2} \int_{-\infty}^{\infty} \left[\sin 2\alpha \psi(x) - \cos 2\alpha \psi(x+ng) \right]^2 dx \quad (5.8)$$

The plot of survival probabilities vs. n is shown in figure 5.2. $P'_T(n) + P'_R(n) = 1$ as the photon should be found in one of the ports.

Simulating the experiment

Let most of the power of the laser beam be confined in a radius of 1.5 mm. We consider $|\psi\rangle$ to be confined in a finite region $-d \leq x \leq d$, where $d = 3$ mm. This is a valid assumption as a Gaussian distribution is mostly confined in 6σ and we take $\sigma = 0.4$ for the simulation. If we represent the total wave function of the photon

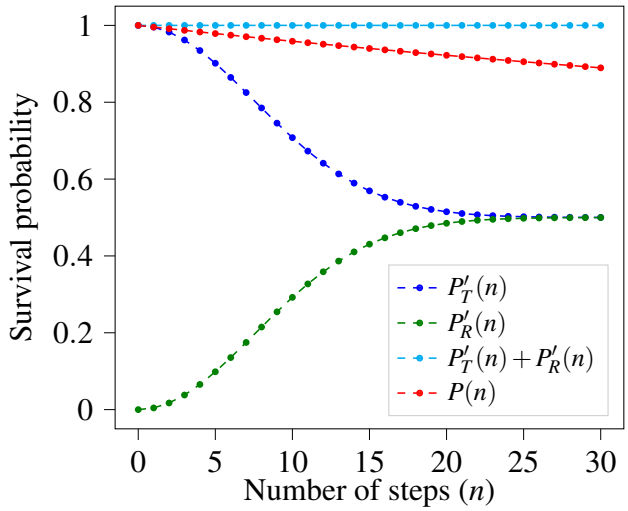


Fig. 5.2 The standard deviation of the Gaussian distribution is $\sigma = 0.4$ and the coupling strength is taken as $g = 106\ \mu m$. These plots are for $\alpha = \pi/8$, the value which we take for the simulation. It can be shown that $\lim_{n\to\infty} P(n) = 0$ and $\lim_{n\to\infty} P'_T(n) = \lim_{n\to\infty} P'_R(n) = 1/2$. The *protective mechanism* works only for a limited number of steps, as for some n, $P(n)$ would definitely become less than $P'_T(n)$.

using seven qubits (one for the photon polarization state and $n = 6$ for $|\psi\rangle$), then the differential step size $\Delta x = \frac{2d}{2^n - 1} = 95.24\ \mu m$ is less than $g = 106\ \mu m$, which is the shift introduced in single step. Seven qubits are therefore sufficient to represent the photon wave function. $|\psi\rangle$ can be written as

$$|\psi\rangle = \sum_{j=0}^{N-1} a_j |j\rangle \tag{5.9}$$

where we consider the discretized version of the normal distribution on $N = 2^n$ equidistant points a_j and $|0\rangle$, $|1\rangle$, ..., $|j\rangle$, ..., $|N-1\rangle$ are the computational basis states of the n qubits. We next discuss how to simulate the effect of the birefringent crystals and the polarizer.

Simulating the effect of birefringent crystals

We use the relation $\hat{p}\psi(x) = \mathscr{F}^{-1} p \mathscr{F} \psi(x) = \mathscr{F}^{-1} p \phi(p)$ where $\phi(p)$ is the Fourier transform ($\mathscr{F}$) of $\psi(x)$, to switch to the momentum basis where $\hat{p}$ is diagonal and then back to the position basis. In order to simulate the effect of birefringent crystals,

we start with 5.3 and use the inverse quantum Fourier transform U_{QFT}^{-1} instead of the continuous $\mathscr{F}$ and vice versa (see the appendix), with the property that U_{QFT} is unitary. We have

$$\begin{aligned}
|H\rangle \otimes e^{-\frac{i}{\hbar}g\hat{p}}|\psi\rangle &= |H\rangle \otimes e^{-\frac{i}{\hbar}gU_{\text{QFT}}\hat{p}U_{\text{QFT}}^{-1}}|\psi\rangle \\
&= |H\rangle \otimes \left[\mathbb{1} - \frac{i}{\hbar}gU_{\text{QFT}}\hat{p}U_{\text{QFT}}^{-1} + \frac{i^2}{2\hbar^2}g^2U_{\text{QFT}}\hat{p}^2U_{\text{QFT}}^{-1} - ...\right]|\psi\rangle \\
&= |H\rangle \otimes U_{\text{QFT}}\left[\mathbb{1} - \frac{i}{\hbar}g\hat{p} + \frac{i^2}{2\hbar^2}g^2\hat{p}^2 - ...\right]U_{\text{QFT}}^{-1}|\psi\rangle \\
&= |H\rangle \otimes U_{\text{QFT}}e^{-\frac{i}{\hbar}g\hat{p}}U_{\text{QFT}}^{-1}|\psi\rangle \\
&= |H\rangle \otimes U_{\text{QFT}}e^{-\frac{i}{\hbar}gp}|\phi\rangle
\end{aligned} \tag{5.10}$$

We can apply U_{QFT}^{-1} on $|\psi\rangle$ to obtain $|\phi\rangle$, the wave function in the momentum space, then apply $e^{-\frac{i}{\hbar}gp}$ and finally apply U_{QFT} to obtain the wave function again in position space. The transformation $|p\rangle \rightarrow e^{if(p)}|p\rangle$, where $f(p) = -\frac{g}{\hbar}p$ and $p =$ 0, 1, 2, ..., $N-1$ or equivalently $|p\rangle \rightarrow e^{i\lambda p}|p\rangle$, where λ is a real number, can be achieved by applying controlled phase shifts [85] as shown in figure 5.3. The value of λ is not necessarily $g/\hbar$ but will have a specific value for the circuit of figure 5.3. We are transitioning from translation of the wave function in theory to the one to be done by the circuit of figure 5.3. In a sense, λ acts as a calibration parameter for this transition and the way it is determined is discussed in the result section. The gate $F_j = \begin{bmatrix} e^{if(2j)} & 0 \\ 0 & e^{if(2j+1)} \end{bmatrix}$, where $j \in \{0,1,2,...,2^{n'}-1\}$ with $n' = n-1$, is applied to the n^{th} qubit and the first n' qubits act as control qubits. For example, if the first five qubits are in the state $|00000\rangle$, then phases $e^{if(0)}$ and $e^{if(1)}$ are set for the basis vectors $|000000\rangle$ and $|000001\rangle$, respectively.

The transformation $|p\rangle \rightarrow e^{if(p)}|p\rangle$ can also be achieved by evaluating the function $f(p)$ [11]. Assuming that the function $f : \{0,...,2^m-1\} \rightarrow \{0,...,2^n-1\}$, where m and n are positive integers, can be computed reversibly using the ancillary register $|y\rangle$, i.e., the operation $|p,y\rangle \rightarrow |p,y \oplus f(p)\rangle$ can be performed reversibly, followed by the transformation $|p,y \oplus f(p)\rangle \rightarrow e^{if(p)}|p,y \oplus f(p)\rangle$ and finally the uncomputation step $e^{if(p)}|p,y \oplus f(p)\rangle \rightarrow e^{if(p)}|p,y\rangle$, thereby implementing the desired transformation. Here, $\oplus$ represents the addition modulo 2 operation. For $f(p) = \lambda p$, we can skip the first and third steps and invoke just the second step by taking only the qubits that represent the pointer wave function, i.e., without using any

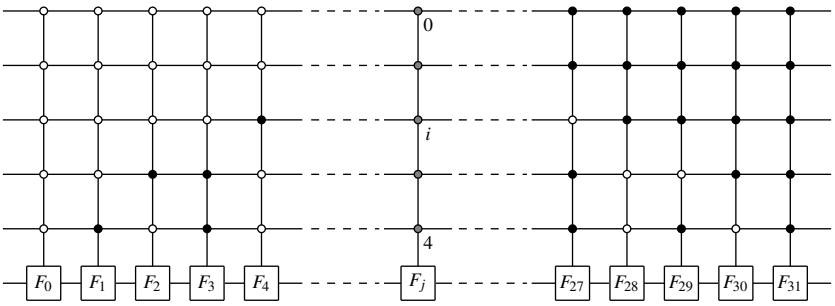


Fig. 5.3 Quantum circuit to perform the transformation $|p\rangle \rightarrow e^{if(p)}|p\rangle$ where $f(p) = \lambda p$ and $p \in \{0,1,2,...,2^n-1\}$ where $n = 6$ is the number of qubits representing the Gaussian state $|\psi\rangle$. The gate $F_j = \begin{bmatrix} e^{if(2j)} & 0 \\ 0 & e^{if(2j+1)} \end{bmatrix}$, where $j \in \{0,1,2,...,2^{n'}-1\}$ with $n' = n-1$, is applied to the n^{th} qubit and the first n' qubits act as control qubits. If $j = \sum_{i=0}^{n'-1} j_i 2^i$ where $j_i \in \{0,1\}$, then for F_j, the conditional on i^{th} qubit where $i \in \{0,1,2,...,n'-1\}$, is set to zero or one depending on whether j_i is 0 or 1, respectively. The circuit implements the transformation that can be represented by a $2^n \times 2^n$ unitary matrix $U_p = \text{diag}(F_0,...,F_j,...,F_{31})$.

ancillary qubit. We have for $p \in [0, 2^n-1]$ and $p = \sum_{j=0}^{n-1} p_j 2^j$ where $p_j \in \{0,1\}$, $e^{i\lambda p} = e^{\sum_{j=0}^{n-1} i\lambda p_j 2^j} = \prod_{j=0}^{n-1} e^{i\lambda p_j 2^j}$, which can be implemented using n single qubit gates. The matrix representation of such a gate that acts on the j^{th} qubit is given by $U_j = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda 2^j} \end{bmatrix}$. We would call the transformation $|p\rangle \rightarrow e^{if(p)}|p\rangle$ as *phase shift*. The operation $U_{\text{QFT}}^{-1} - \textit{phase shift} - U_{\text{QFT}}$ implements the translation of the pointer state $|\psi\rangle$.

Simulating the effect of polarizer

The operator for the polarizer f is the projector $P_{\pi/4} = |+\rangle\langle+|$, which is non-unitary. To simulate this [86], we consider $P_{\pi/4}$ as a linear combination of unitary operators,

$$P_{\pi/4} = \frac{1}{2}(2P_{\pi/4} - I) + \frac{I}{2} = \frac{X}{2} + \frac{I}{2} \tag{5.11}$$

Let $|\phi\rangle$ be the wave function of the system on which $P_{\pi/4}$ is to be applied. Consider the following circuit where the first qubit is an ancilla qubit.

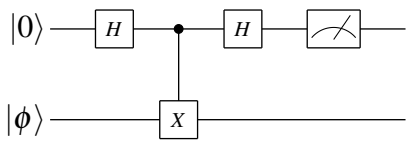


Fig. 5.4

The state of the system will evolve as

$$\begin{aligned}
|0\rangle|\phi\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}}\big[|0\rangle+|1\rangle\big]|\phi\rangle \\
&\xrightarrow{CX} \frac{1}{\sqrt{2}}\big[|0\rangle|\phi\rangle+|1\rangle X|\phi\rangle\big] \\
&\xrightarrow{H} |0\rangle\left[\frac{I}{2}+\frac{X}{2}\right]|\phi\rangle+|1\rangle\left[\frac{I}{2}-\frac{X}{2}\right]|\phi\rangle
\end{aligned}$$

If the ancilla qubit is measured and found to be 0, then only $\frac{I}{2}+\frac{X}{2}$ acts on $|\phi\rangle$. The readings for which the ancilla measures 1 are discarded. We now describe the steps to simulate the experiment.

Simulation steps

For the protective case, n steps can be simulated using the circuit shown in figure 5.5. Here, n ancilla qubits a_1, a_2, ..., a_n are used and p represents the polarization state of the photon. $|0\rangle$ and $|1\rangle$ represent horizontal and vertical polarization states, respectively.

Starting with the initial state $|\psi\rangle|0\rangle_p \otimes_{i=1}^{n}|0\rangle_{a_i}$, the state of the quantum computer $|\alpha\rangle$ after the n ($n>0$) steps, assuming that the ancilla qubits a_1, a_2, ..., a_{n-1} were measured and found to be 0, will be

$$|\alpha\rangle \propto \frac{1}{2^n}\left[\sum_{k=0}^{n}\binom{n}{k}|\psi\rangle_{x-kg}\right]|0\rangle_p|0\rangle_{a_n} + \text{terms of } |1\rangle_{a_n}$$

Again, we consider only those readouts for which a_n measures 0. For the non-protective case, n steps can be simulated using the circuit shown in figure 5.6.

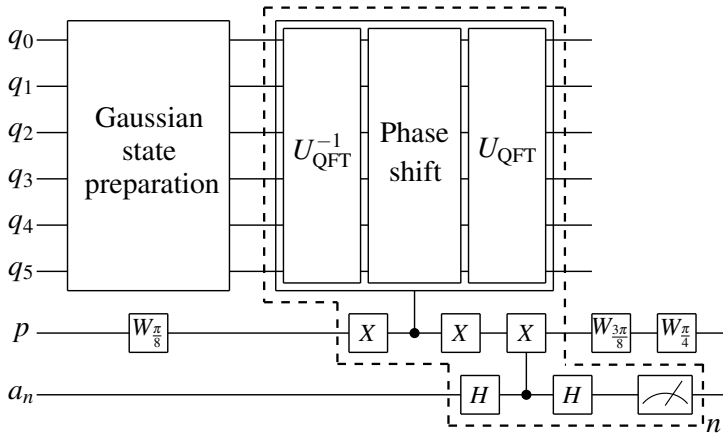


Fig. 5.5 Quantum circuit to simulate the protective case: Qubit p represents the polarization state of the photon and H, Z and X are the Hadamard gate and the Pauli Z and X gate, respectively. The operators for HWP e, g and h are $W_{\pi/8} = H$, $W_{3\pi/8} = HZX$ and $W_{\pi/4} = X$, respectively. The phase shift operation is performed by the circuit shown in figure 5.3, acting on pointer state $|\psi\rangle$. U_{QFT} and U_{QFT}^{-1} represent the quantum Fourier transform and its inverse, respectively.

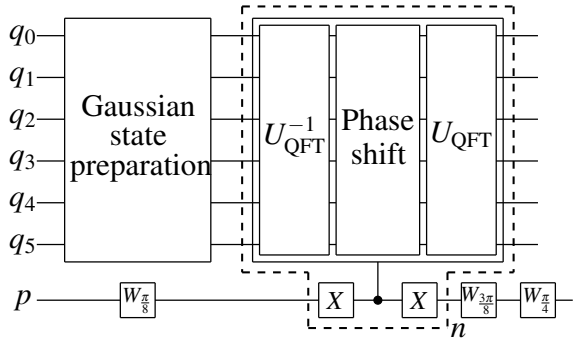


Fig. 5.6 Quantum circuit to simulate the non-protective case

In this case, the state of the quantum computer $|\alpha'\rangle$ after n steps will be as follows.

$$|\alpha'\rangle = \frac{|\psi\rangle + |\psi\rangle_{x-ng}}{2}|0\rangle_p + \frac{|\psi\rangle - |\psi\rangle_{x-ng}}{2}|1\rangle_p \tag{5.12}$$

If we consider the readouts for which the qubit p measures 0 or 1, we obtain $P'_T(n)$ or $P'_R(n)$, respectively.

5.4 Results

We first determine the constant λ that appears in the phase shift implemented using the circuit shown in figure 5.3. We do this by initializing the qubits representing the pointer state with amplitudes of $|\psi\rangle_{x-g}$ using 5.9 and comparing it with those of $|\psi\rangle$ after transformation $U_{\text{QFT}}^{-1} - \textit{phase shift} - U_{\text{QFT}}$. We compare the state vectors in the two cases by varying λ in the latter case to look for the maxima to coincide in both the cases. We perform this simulation of translation of the pointer state using noise-free *statevector* simulator provided in Qiskit's *Aer* simulator backend. For $n = 6$, $\lambda \in [0.913, 0.926]$ and for $n = 7$, $\lambda \in [0.921, 0.926]$. For $n > 7$, we do not obtain any better accuracy due to finite sampling of the output state vector [87]. Figures 5.7 and 5.8 show the results for the survival probability for the protective and non-protective cases, respectively, versus the number of steps when executed using *statevector* simulator with varying shots and the value of λ taken as 0.921.

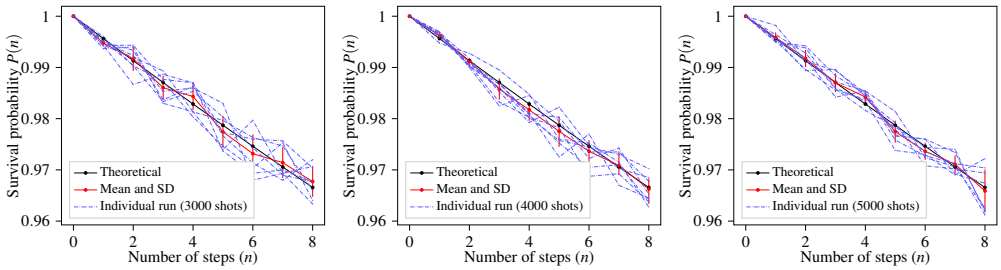


Fig. 5.7 Plot of survival probability of photon vs. number of steps for *protective* case, obtained by executing the circuit in figure 5.5 with varying shots and λ taken as 0.921. Each of the plots show the mean and standard deviation for eight runs.

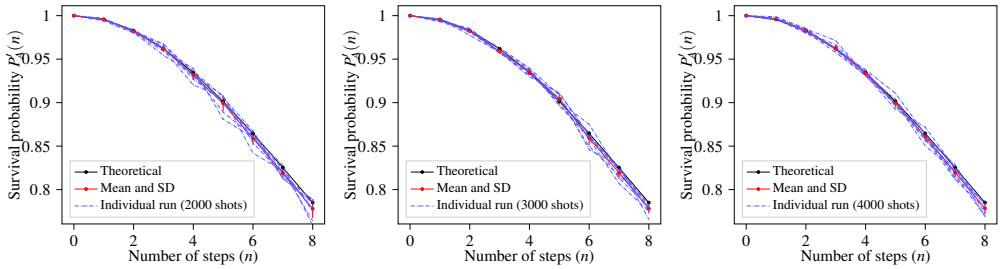


Fig. 5.8 Plot of survival probability of photon vs. number of steps for $non - protective$ case, obtained by executing the circuit in figure 5.6 with varying shots and λ taken as 0.921. Each of the plots show the mean and standard deviation for eight runs.

The data points of figures 5.7 and 5.8 are given in tables (A.1, A.2, A.3) and (A.4, A.5, A.6), respectively. Tables 5.1 and 5.2 show the survival probabilities for different shots for protective and non-protective cases, respectively. $P(n)$ is the theoretical value, X_i^n denote the mean of eight runs for i shots and E_i^n is the percentage relative error (PRE) given by $E_i^n = \frac{X_i^n - P(n)}{P(n)} \times 100\%$. For $non-protective$ case $P(n)$ is replaced by $P'_A(n)$.

n	1	2	3	4	5	6	7	8
$P(n)$	0.99563015	0.99131724	0.98706007	0.98285742	0.97870815	0.97461112	0.97056526	0.9665695
X^n_{3000}	0.99475	0.99166667	0.98604167	0.98433333	0.97741667	0.97316667	0.97141667	0.96770833
$E^n_{3000}\%$	-0.08840081	0.03524831	-0.1031751	0.1501655	-0.13195771	-0.14820864	0.08772262	0.11782227
X^n_{4000}	0.99625	0.990875	0.98578125	0.98146875	0.97746875	0.9733125	0.97121875	0.96584375
$E^n_{4000}\%$	0.06225754	-0.04461176	-0.12955817	-0.14128911	-0.12663607	-0.13324541	0.06733072	-0.07508505
X^n_{5000}	0.99585	0.99175	0.987025	0.984225	0.977475	0.973625	0.9711	0.9659
$E^n_{5000}\%$	0.02208198	0.04365463	-0.00355266	0.13914322	-0.12599747	-0.10118134	0.05509559	-0.0692655

Table 5.1 Mean and percentage relative error for varying shots for protective case. The value of λ was taken as 0.921.

n	1	2	3	4	5	6	7	8
$P'_A(n)$	0.99563015	0.9827484	0.96201852	0.93448356	0.9014789	0.86452517,	0.82521288	0.78509091
X^n_{2000}	0.9955625	0.9820625	0.9621875	0.9316875	0.899125	0.858875	0.819	0.777875
$E^n_{2000}\%$	-0.0067942	-0.06979364	0.01756532	-0.2992087	-0.26111512	-0.65355743	-0.75288171	-0.91911728
X^n_{3000}	0.99483333	0.98229167	0.95895833	0.93541667	0.90433333	0.85925	0.81945833	0.77816667
$E^n_{3000}\%$	-0.0800309	-0.04647468	-0.31810043	0.09985307	0.3166392	-0.61018102	-0.69734048	-0.88196659
X^n_{4000}	0.99646875	0.98278125	0.96275	0.93353125	0.8996875	0.8608125	0.8198125	0.7784375
$E^n_{4000}\%$	0.08422855	0.00334309	0.07603612	-0.10190721	-0.19871766	-0.42944597	-0.65442226	-0.84746952

Table 5.2 Mean and percentage relative error for varying shots for non-protective case. The value of λ was taken as 0.921.

$|E^n_{5000}| < |E^n_{4000}|$ for all steps (n) for protective case. Moreover, the PRE is less than 0.14 for n up to 8 for 5000 shots. For non-protective case $|E^n_{4000}| < |E^n_{3000}|$ for all steps except n =1, 4. These values suggest that the number of shots for protective and non-protective cases can be reliably taken as 5000 and 4000, respectively.

Figure 5.9 (5.10) shows the same plots as in 5.7 (5.8), except that here λ is varied and the number of shots is fixed, 5000 and 4000 for the protective and non-protective cases, respectively. From these figures, it can be concluded that $\lambda = 0.921$ gives the optimal result for both cases. We use this value of λ for all subsequent simulations.

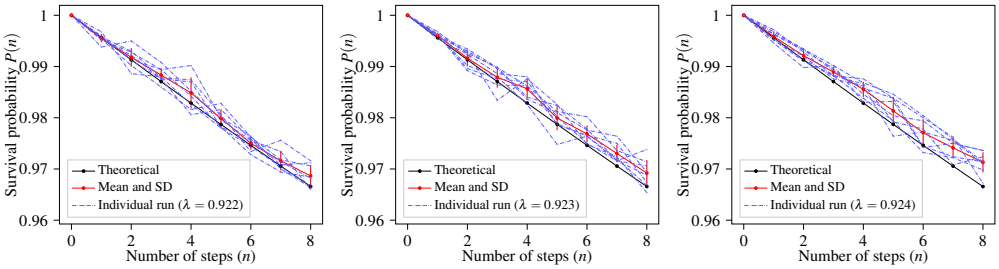


Fig. 5.9 Plot of survival probability of photon vs. number of steps for *protective* case, with varying λ and the number of shots as 5000 in each case. Each of the plots show the mean and standard deviation for eight runs.

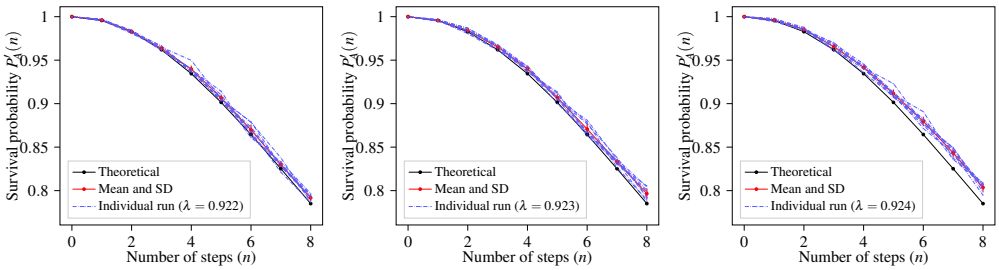


Fig. 5.10 Plot of survival probability of photon vs. number of steps for *non-protective* case, with varying λ and the number of shots as 4000 in each case. Each of the plots show the mean and standard deviation for eight runs.

Figure 5.11 shows the simulation results, depicting the mean and standard deviation for eight runs executed with 5000 and 4000 shots for the protective and non-protective cases, respectively.

5.5 Summary

It is evident from figure 5.9 that the quantum simulation performed on a classical computer has significant deviations from theoretical results. Firstly, the different graphs for different runs with the same number of shots can be attributed to the finite-sampling noise due to repeated circuit evaluations, even on a noise-free machine. Secondly, the standard deviations for individual runs are greater in the protective case than in the non-protective case because of greater number of gates and qubits involved as each step adds an extra ancillary qubit in protective case. From tables 5.1 and 5.2, we observe that for non-protective case, the PRE goes till 0.8475 for n

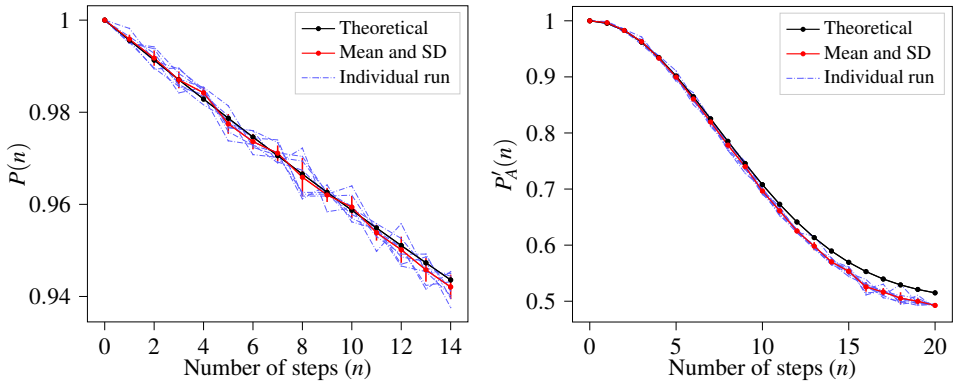


Fig. 5.11 Plot of survival probability of photon vs. number of steps for *protective* and *non − protective* cases, obtained by executing the circuits (using *statevector* simulator) in figures 5.5 and 5.6 with 5000 and 4000 shots, respectively. Each of the plots show the mean and standard deviation for eight runs.

up to 8 for 4000 shots. In contrast, for protective case, the PRE goes only till 0.1413 for n up to 8 for 4000 shots despite more number of gates and qubits being involved.

A plausible reason for this could be that unlike non-protective measurement, protective case involves repeated partial measurements after each step and therefore decreases the overall quantum error rate. The observed behavior is consistent with a QZE-inspired protective mechanism, but it should not be interpreted as a strict demonstration of QZE, since the protective trend is advantageous only over a limited range of steps and not uniformly for all n. The fact that the PRE is reduced by approximately 83% in the protective case, despite the increased circuit depth, suggests that algorithmic protection can outweigh the noise overhead typically associated with larger gate counts.

Scalability remark. The digital emulation cost is dominated by two factors: the quantum Fourier transform, which requires $O(m^2)$ gates per step (where m denotes the number of qubits used to discretize the spatial pointer), and the amplitude encoding of the Gaussian pointer state, which in general requires $O(2^m)$ gates. In the physical experiment, by contrast, both operations are realized natively: the laser beam itself provides the Gaussian pointer, and the birefringent crystal implements the spatial translation, so that each step costs $O(1)$. This gap highlights the overhead of digital emulation and the intrinsic advantage of continuous-variable photonic platforms for this class of protocols.

Role of the simulation. The purpose of this work is not to replace the physical experiment, which remains more efficient, but rather to validate the gate-model decomposition (QFT + phase shift + QFT) against the known quantum dynamics. Having a validated circuit decomposition enables: (i) access to the full intermediate state vector at each step, which is inaccessible in the lab; (ii) the design and optimization of protocol variants (e.g., different pointer widths, coupling strengths, or protection strategies) without rebuilding the optical bench; and (iii) the exploration of how the protective-measurement scheme adapts to different hardware platforms, where the weak interaction is obtained accordingly.

Within the dissertation narrative, this chapter closes the technical sequence by adding the measurement-and-preservation perspective to the algorithmic and control layers developed in Chapters 3 and 4. These three perspectives are then integrated in Chapter 6 to define a unified hardware-aware roadmap for future large-scale QML validation.

Chapter 6

Conclusions and Future Work

This dissertation has explored the intricate relationship between quantum algorithms and the physical architectures they inhabit. The central objective was to demonstrate that the challenges of the NISQ era can be mitigated through hardware-aware optimizations that bridge the gap between high-level software and low-level pulse control. Through the investigations presented in Chapters 3, 4, and 5, we have shown that specialized strategies tailored to specific quantum platforms can yield performance improvements that surpass general-purpose gate-based approaches.

From a broader QML perspective, the current state of the field is promising but still pre-deployment: available models are typically small, benchmarks are often synthetic, and experimental pipelines remain sensitive to calibration, noise, and data-interface overhead. In this sense, the key question is not only whether quantum models can outperform classical baselines in isolated settings, but what is still missing for robust real-world utility. The results of this dissertation indicate four enabling conditions: hardware-aware model design, scalable and low-latency control, reliable state-preparation and readout mechanisms, and clearer problem-class characterization for which quantum resources provide a net advantage.

The results from Chapter 3 highlight the potential of neutral atom arrays as a robust platform for kernel-based learning. By leveraging the Rydberg blockading mechanism and geometric optimization, we demonstrated that QSVMs can achieve high classification accuracy with minimal qubit resources. This finding underscores the importance of "quantum expressivity", the ability of a quantum system to represent complex data distributions, which often precedes raw computational speedup.

For future work, a primary direction involves the scaling of hardware-aware QML models to real-world, high-dimensional datasets. While this dissertation demonstrated success on synthetic benchmarks, the next logical step is to evaluate how the pulse-level optimizations and geometric configurations developed in Chapter 3 for neutral atom arrays perform when applied to industrial-scale classification tasks. In particular, the integration of specialized Rydberg-state dynamics with more complex VQC architectures could reveal further advantages in terms of expressivity and resource efficiency. Furthermore, it is essential to understand for what class of datasets this method actually brings a quantum advantage.

Similarly, the work conducted at Fermilab, detailed in Chapter 4, proved that embedded machine learning is a viable path toward high-fidelity quantum control. The ability to optimize control pulses in real-time, without the overhead of host-to-controller communication, represents a significant step toward the scalability of superconducting quantum processors. Future research should prioritize the deployment of these embedded machine learning models on larger-scale processors. Exploring the transferability of pre-trained control models between different cryogenic environments or hardware generations could significantly reduce the overhead of initial system characterization. Furthermore, the convergence of these real-time control strategies with the state-preservation techniques explored in Chapter 5 presents a promising pathway; specifically, leveraging QZE-inspired protective strategies to dynamically stabilize entangled states during the long execution times required by deep QML training procedures.

In Chapter 5, the photonic simulation of protective measurements provided a critical benchmark for measurement stability. The study showed a finite-step survival-probability advantage in the protective regime, which supports a practically useful state-preservation mechanism that is a clear advantage when dealing with current technological limitations of qubits. The immediate next step involves transitioning from classical emulation to real-time implementation on physical NISQ photonic hardware. This would allow for a direct assessment of how protective measurements and QZE-related mechanisms can mitigate environmental decoherence in active laboratory settings. Such investigations are critical for stabilizing the state-preparation and readout phases of the QML architectures discussed in Chapters 3 and 4. Additionally, extending the simulation to include non-Gaussian states and more complex spatial beam profiles could provide a richer encoding landscape for quantum kernels. This would bridge fundamental quantum dynamics with practical quantum network-

ing and distributed learning frameworks, ensuring high-fidelity, noise-resilient state preservation for the next generation of quantum intelligent systems.

6.1 Limitations and Open Directions

The scope of the experiments presented in this dissertation naturally defines a number of open directions for future investigation.

Problem scale. The neutral-atom QKE pipeline (Chapter 3) was validated with $N = 3$ qubits due to the computational cost of full-state emulation via QuTiP; the embedded pulse-synthesis pipeline (Chapter 4) targeted single-qubit $R_x(\theta)$ gates; and the photonic simulation (Chapter 5) encoded the spatial beam profile into $6+1$ qubits. In all three cases, the methodologies are designed to be qubit-count agnostic, but their empirical validation at larger scales remains an open milestone. On the neutral-atom side, the availability of tensor-network or GPU-accelerated simulators, as well as access to real neutral-atom processors, would enable benchmarking with tens of qubits. On the superconducting side, extending the embedded calibration framework to multi-qubit gates and to processors with non-negligible crosstalk is a natural next step.

Dataset generality. The QKE experiments relied on Qiskit's synthetic `ad_hoc_data` generator, which is specifically designed to exhibit a quantum-classical separation gap. Assessing whether the observed accuracy advantage persists on real-world, high-dimensional datasets is a key direction for establishing practical utility of the method, and more broadly of quantum feature kernels.

Noise-free simulation. The photonic protective-measurement study was carried out on a noiseless statevector simulator. Introducing realistic noise models (depolarizing, amplitude-damping, readout errors) would clarify how the finite-step survival probability advantage degrades and whether error-mitigation techniques can preserve it. Similarly, deploying the simulation on real photonic or gate-based NISQ hardware would provide direct experimental validation of the Quantum Zeno Effect–based protection mechanism.

Finally, a significant frontier lies in the synthesis of quantum computing with the Neuro-Symbolic Artificial Intelligence paradigm introduced as a side research in Appendix C. Future efforts should aim to integrate the formal logic frameworks and Logic Tensor Networks with the QML methods discussed in this dissertation. By embedding symbolic constraints directly into the quantum optimization pipeline, it may be possible to develop hybrid models that are not only computationally superior but also logically consistent and explainable. Such a holistic integration, spanning from the physical pulse control to the symbolic reasoning layer, will be essential for realizing the full potential of quantum intelligent systems in critical applications.

Appendix A

Experimental and theoretical details for Chapter 5

A.1 Experimental details

We describe the relevant elements of the experimental setup as well as the operators associated with them. Let $|H\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|V\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. The operator for a HWP whose fast axis is inclined at an angle α is $W_\alpha = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix}$. In figure 5.1, the HWP b is placed before a PBS (not shown) and can be used to change the photon polarization to tune the intensity of the laser beam after PBS d. For HWP e, g and h, the operators are $W_{\pi/8}$, $W_{3\pi/8}$ and $W_{\pi/4}$, respectively. All of the PBS transmit horizontally polarized photons and reflect vertically polarized ones. The direction of optic axis of the two birefringent crystals (calcite) i and j is as follows. For i, it is in the $xy-$plane and inclined 45° with the $x-$axis and for j which is used only for phase compensation, it is along the $z-$axis. The operator of a polarizer oriented at an angle θ with the $x-$axis is $P_\theta = |\theta\rangle\langle\theta|$, where $|\theta\rangle = \cos\theta\,|H\rangle + \sin\theta\,|V\rangle$. The operator for polarizer f is $P_{\pi/4}$.

A.2 Wave function in position and momentum space

The wave function $\psi(x)$ of a free particle in the position space is related to its wave function in the momentum space $\phi(p)$ (p is the momentum along the $x-$axis) as

$$\phi(p) = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} \psi(x) e^{-ipx/\hbar} dx = \mathscr{F}\left[\psi(x)\right] \tag{A.1}$$

$$\psi(x) = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} \phi(p) e^{ipx/\hbar} dp = \mathscr{F}^{-1}\left[\phi(p)\right] \tag{A.2}$$

where $\mathscr{F}$ and $\mathscr{F}^{-1}$ represent the Fourier transform and its inverse, respectively. Differentiating A.2,

$$\begin{aligned}
\frac{\partial \psi(x)}{\partial x} &= \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} \frac{ip}{\hbar} \phi(p) e^{ipx/\hbar} dp \\
\Rightarrow \mathscr{F}\left[-i\hbar \frac{\partial \psi(x)}{\partial x}\right] &= p\phi(p) \\
\Rightarrow \mathscr{F}^{-1}\mathscr{F}\left[-i\hbar \frac{\partial \psi(x)}{\partial x}\right] &= \mathscr{F}^{-1} p \mathscr{F}\left[\psi(x)\right] \\
\Rightarrow -i\hbar \frac{\partial}{\partial x} &= \mathscr{F}^{-1} p \mathscr{F} \\
\Rightarrow \hat{p} &= \mathscr{F}^{-1} p \mathscr{F}
\end{aligned}$$

It is important to note that the right side has to be applied to functions in momentum base in order for the equality to hold. The relation 5.10 can also be obtained by using the shifting property of the Fourier transform. We have

$$\begin{aligned}
\mathscr{F}\left[\psi(x-g)\right] &= e^{-\frac{i}{\hbar}gp}\phi(p) \\
\Rightarrow \psi(x-g) &= \mathscr{F}^{-1} e^{-\frac{i}{\hbar}gp} \mathscr{F}\left[\psi(x)\right] \\
\Rightarrow e^{-\frac{i}{\hbar}g\hat{p}}\psi(x) &= \mathscr{F}^{-1} e^{-\frac{i}{\hbar}gp} \mathscr{F}\left[\psi(x)\right] \\
\Rightarrow e^{-\frac{i}{\hbar}g\hat{p}}|\psi\rangle &= U_{\mathrm{QFT}} e^{-\frac{i}{\hbar}gp} U_{\mathrm{QFT}}^{-1}|\psi\rangle
\end{aligned}$$

Note that the Fourier transform $\mathscr{F}$ corresponds to the inverse quantum Fourier transform and viceversa, due to the fact that there is a $-$ sign in the argument for

the exponential in $\mathscr{F}$ and a $+$ sign in that of U_{QFT}. Indeed, the quantum Fourier transform of the state given by 5.9 is $U_{QFT}\left|\psi\right\rangle$, where

$$U_{\mathrm{QFT}} = \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} e^{2\pi i jk/N} \left|k\right\rangle \left\langle j\right|, \tag{A.3}$$

so the inverse quantum Fourier transform U_{QFT}^{-1} is given by

$$U_{\mathrm{QFT}}^{-1} = U_{QFT}^{\dagger} = \frac{1}{\sqrt{N}} \sum_{k,j=0}^{N-1} e^{-2\pi i jk/N} \left|j\right\rangle \left\langle k\right| \tag{A.4}$$

A.3 Data points for individual runs

n	1	2	3	4	5	6	7	8
X_1	0.99600000	0.99033333	0.98333333	0.98533333	0.98033333	0.97433333	0.96900000	0.96766667
X_2	0.99433333	0.99200000	0.98933333	0.98133333	0.97966667	0.97633333	0.97466667	0.96600000
X_3	0.99433333	0.99433333	0.98766667	0.98700000	0.97366667	0.96933333	0.97000000	0.96766667
X_4	0.99366667	0.99400000	0.98300000	0.98133333	0.97400000	0.97966667	0.96933333	0.97066667
X_5	0.99433333	0.98666667	0.98866667	0.98700000	0.97733333	0.97600000	0.97566667	0.96966667
X_6	0.99500000	0.99233333	0.98533333	0.98600000	0.98300000	0.96800000	0.96933333	0.96333333
X_7	0.99533333	0.99033333	0.98733333	0.98033333	0.97466667	0.97033333	0.97533333	0.96400000
X_8	0.99500000	0.99333333	0.98366667	0.98633333	0.97666667	0.97133333	0.96800000	0.97200000

Table A.1 Probabilities for eight individual runs for 3000 shots and $\lambda = 0.921$ for *protective* case.

n	1	2	3	4	5	6	7	8
X_1	0.99700000	0.99025000	0.98675000	0.98100000	0.97250000	0.97700000	0.96700000	0.96450000
X_2	0.99650000	0.99075000	0.98600000	0.98025000	0.97825000	0.96875000	0.96925000	0.96375000
X_3	0.99600000	0.99075000	0.98700000	0.98075000	0.97525000	0.97050000	0.97100000	0.96400000
X_4	0.99675000	0.99175000	0.98525000	0.97925000	0.98225000	0.97475000	0.97025000	0.96850000
X_5	0.99675000	0.99000000	0.98250000	0.98450000	0.97950000	0.97225000	0.97425000	0.96525000
X_6	0.99575000	0.99050000	0.98325000	0.98025000	0.97575000	0.97450000	0.97300000	0.97025000
X_7	0.99650000	0.99025000	0.98600000	0.98100000	0.98025000	0.97350000	0.97350000	0.96775000
X_8	0.99475000	0.99275000	0.98950000	0.98475000	0.97600000	0.97525000	0.97150000	0.96275000

Table A.2 Probabilities for eight individual runs for 4000 shots and $\lambda = 0.921$ for *protective* case.

n	1	2	3	4	5	6	7	8
X_1	0.99820000	0.99120000	0.98940000	0.98500000	0.97580000	0.97280000	0.97120000	0.97040000
X_2	0.99500000	0.99380000	0.98600000	0.98340000	0.97380000	0.97300000	0.96900000	0.97220000
X_3	0.99500000	0.99420000	0.98840000	0.98480000	0.97680000	0.97600000	0.97340000	0.96220000
X_4	0.99580000	0.99180000	0.98420000	0.98540000	0.98140000	0.97240000	0.97060000	0.96260000
X_5	0.99560000	0.98940000	0.98960000	0.98440000	0.97780000	0.97360000	0.97120000	0.96120000
X_6	0.99520000	0.98960000	0.98560000	0.98380000	0.97920000	0.97080000	0.97020000	0.96940000
X_7	0.99560000	0.99120000	0.98720000	0.98540000	0.97660000	0.97600000	0.96920000	0.96740000
X_8	0.99640000	0.99280000	0.98580000	0.98160000	0.97840000	0.97440000	0.97400000	0.96180000'

Table A.3 Probabilities for eight individual runs for 5000 shots and $\lambda = 0.921$ for *protective* case.

n	1	2	3	4	5	6	7	8
X_1	0.99600000	0.98050000	0.95450000	0.93300000	0.88900000	0.86150000	0.81300000	0.78450000
X_2	0.99650000	0.98300000	0.96350000	0.92700000	0.90100000	0.85900000	0.82050000	0.77800000
X_3	0.99550000	0.98250000	0.96750000	0.92950000	0.90150000	0.86200000	0.82800000	0.75850000
X_4	0.99700000	0.98100000	0.95900000	0.93950000	0.88100000	0.86000000	0.82350000	0.77550000
X_5	0.99550000	0.98300000	0.96050000	0.92050000	0.90550000	0.86750000	0.81300000	0.78600000
X_6	0.99500000	0.98250000	0.96250000	0.93200000	0.90800000	0.85400000	0.81700000	0.78800000
X_7	0.99400000	0.98000000	0.96700000	0.93550000	0.90050000	0.86500000	0.81550000	0.78850000
X_8	0.99500000	0.98400000	0.96300000	0.93650000	0.90650000	0.84200000	0.82150000	0.76400000

Table A.4 Probabilities for eight individual runs for 2000 shots and $\lambda = 0.921$ for $non-$ *protective* case.

n	1	2	3	4	5	6	7	8
X_1	0.99300000	0.98200000	0.95700000	0.93700000	0.90133333	0.87500000	0.82066667	0.77900000
X_2	0.99466667	0.98166667	0.96333333	0.93966667	0.90866667	0.84566667	0.82766667	0.76600000
X_3	0.99533333	0.98066667	0.95900000	0.93500000	0.89866667	0.85833333	0.80700000	0.77500000
X_4	0.99633333	0.97733333	0.95800000	0.93166667	0.91033333	0.86333333	0.81300000	0.78033333
X_5	0.99600000	0.98400000	0.95766667	0.93766667	0.90300000	0.84966667	0.81600000	0.78500000
X_6	0.99500000	0.98300000	0.95700000	0.93433333	0.90500000	0.85933333	0.82366667	0.77600000
X_7	0.99466667	0.98433333	0.96133333	0.93833333	0.89633333	0.86100000	0.82466667	0.78066667
X_8	0.99366667	0.98533333	0.95833333	0.92966667	0.91133333	0.86166667	0.82300000	0.78333333

Table A.5 Probabilities for eight individual runs for 3000 shots and $\lambda = 0.921$ for $non-$ *protective* case.

n	1	2	3	4	5	6	7	8
X_1	0.99675000	0.98075000	0.96650000	0.93325000	0.90350000	0.85900000	0.81675000	0.76950000
X_2	0.99625000	0.97975000	0.96200000	0.93525000	0.89775000	0.86500000	0.81325000	0.78650000
X_3	0.99700000	0.98550000	0.96075000	0.93125000	0.89925000	0.85900000	0.81150000	0.77850000
X_4	0.99875000	0.98475000	0.97125000	0.93175000	0.89225000	0.87175000	0.82475000	0.76925000
X_5	0.99500000	0.98250000	0.96225000	0.93700000	0.89575000	0.86125000	0.82850000	0.77300000
X_6	0.99675000	0.98275000	0.96075000	0.93075000	0.89850000	0.85650000	0.82400000	0.78425000
X_7	0.99600000	0.98175000	0.95750000	0.93125000	0.89975000	0.85025000	0.82150000	0.78300000
X_8	0.99525000	0.98450000	0.96100000	0.93775000	0.91075000	0.86375000	0.81825000	0.78350000

Table A.6 Probabilities for eight individual runs for 4000 shots and $\lambda = 0.921$ for $non-$ *protective* case.

Appendix B

Quantum Computing Technologies

B.1 Introduction

As established in the preceding chapters, the transition from abstract quantum algorithms to practical hardware implementations is contingent upon the specific physical characteristics of the chosen platform. While Chapter 2 provides a systematic analysis of the theoretical QML landscape and its foundational principles, the diversity of algorithms and control strategies explored in this dissertation necessitates a more granular understanding of the specific quantum technologies currently available. The methodologies developed in this dissertation are inherently platform-dependent. For instance, the universal gate-based approach for Quantum Kernel Estimation presented in Chapter 3 exploits the geometric flexibility unique to neutral atom arrays to overcome connectivity barriers. Conversely, the embedded machine learning framework investigated in Chapter 4 is specifically tailored to the cryogenic control requirements and communication bottlenecks of superconducting transmon qubits. Building these foundations is essential for identifying the structural nuances and unique capabilities of quantum models beyond immediate speedups. The purpose of this appendix is thus to provide a technical overview of the quantum computing technologies that support these diverse applications. By examining the fundamental principles and operational limitations of superconducting, photonic, and atomic systems, as well as non-universal paradigms like quantum annealing, this appendix serves as a self-contained reference for the hardware architectures discussed throughout the dissertation.

B.2 Superconducting Qubits

Superconducting qubits are a leading technology in the field of quantum computing, utilizing the principles of superconductivity to create quantum bits, or qubits. These qubits are typically made from superconducting materials such as niobium or aluminum, which, when cooled to near absolute zero, exhibit zero electrical resistance (superconductivity). There are several types of superconducting qubits, including charge qubits, flux qubits, and phase qubits. Transmon qubits, that are the subject of Chapter 4, are a particular type of charge qubits. Each type manipulates different quantum properties to encode information. For instance, charge qubits use the presence or absence of Cooper pairs (pairs of electrons) on a superconducting island to represent the qubit states, whereas flux qubits use the direction of the circulating supercurrent. Phase qubits, on the other hand, leverage the phase difference of the superconducting wave function across a Josephson junction. One of the key advantages of superconducting qubits is their scalability. They can be fabricated using standard lithographic techniques and integrated into complex circuits. Furthermore, they can be coupled to each other using microwave photons, allowing for the creation of entangled states necessary for quantum computation. Superconducting qubits also benefit from relatively fast gate times, which are essential for executing quantum algorithms efficiently. However, superconducting qubits also face challenges, such as maintaining coherence times long enough to perform computations and minimizing errors due to environmental noise. Researchers are actively developing techniques such as error correction codes and improving qubit design to address these issues. Moreover, the need for extremely low temperatures (millikelvin range) requires sophisticated cryogenic systems, which adds complexity and cost to the overall setup.

Advantages of Superconducting Qubits

- **Scalability:** Superconducting qubits can be fabricated using well-established lithographic techniques, allowing for the integration of many qubits into a single chip.
- **Fast Gate Times:** They offer relatively fast operation speeds (order of tens of nanoseconds), which are beneficial for running quantum algorithms efficiently.

- **Strong Coupling:** The ability to couple qubits strongly through microwave photons enables effective entanglement and interaction between qubits, crucial for complex quantum operations.
- **Mature Fabrication Techniques:** Leveraging existing semiconductor fabrication technologies aids in the rapid development and scaling of superconducting qubit systems.

Disadvantages of Superconducting Qubits

- **Short Coherence Times:** The coherence times of superconducting qubits are limited, which can lead to errors during quantum computations.
- **Environmental Sensitivity:** They are susceptible to noise from the environment, necessitating advanced error correction methods.
- **Cryogenic Requirements:** The need for operation at millikelvin temperatures demands complex and expensive cryogenic systems.
- **Material Defects:** Variations and defects in the superconducting materials can impact the performance and reliability of the qubits.

In summary, superconducting qubits represent a promising and rapidly advancing area of quantum computing technology, with significant potential applications ranging from solving complex computational problems to advancing our understanding of quantum mechanics. Continuous research and development are focused on overcoming the existing challenges to fully harness their capabilities. One of the current superconducting Quantum Computers (QC) is IBM Condor, with 1121 qubits (2025).

B.3 Photonic Qubits

Photonic qubits leverage on photons, the fundamental particles of light, as the carriers of quantum information. These qubits are advantageous because photons can travel long distances with minimal loss and are less susceptible to decoherence from their environment. Photonic qubits are typically encoded using properties

such as polarization, time-bin, or spatial modes of the photons. One of the primary benefits of photonic qubits is their compatibility with existing optical communication infrastructure. This allows for potential integration with current fiber optic networks, facilitating long-distance quantum communication. Additionally, photonics-based systems often operate at room temperature, avoiding the need for complex cryogenic cooling. Photonic qubits can be generated using sources like spontaneous parametric down-conversion or quantum dots, and can be manipulated with devices such as beam splitters, phase shifters, and waveguides. Detection of photonic qubits is achieved using highly sensitive photodetectors. However, there are challenges associated with photonic qubits. Efficient single-photon sources and detectors are still an area of active research, and losses in optical components can impact the fidelity of quantum operations. Furthermore, integrating photonic qubits into scalable quantum circuits remains a significant technical hurdle.

Advantages of Photonic Qubits

- **Low Decoherence:** Photons are less susceptible to environmental noise, resulting in longer coherence times.
- **Room Temperature Operation:** Photonic systems can operate without the need for cryogenic cooling.
- **Integration with Optical Networks:** Compatibility with existing fiber optic infrastructure enables long-distance quantum communication.
- **High Speed:** Photons can travel at the speed of light, facilitating fast information transfer. Gates are in the order of picoseconds to nanoseconds.

Disadvantages of Photonic Qubits

- **Photon Loss:** Losses in optical components and fibers can degrade qubit fidelity.
- **Single-Photon Sources and Detectors:** Efficient generation and detection of single photons are challenging and require further development.
- **Scalability:** Integrating a large number of photonic qubits into scalable quantum circuits is technically demanding.

An example of a photonic QC is Xanadu Borealis, with 216 qubits (2022).

B.4 Trapped Ions and Neutral Atoms

Trapped ions and neutral atoms are another promising approach to building QC, leveraging individual atoms or ions as qubits. These systems utilize electromagnetic fields to trap and manipulate the particles, with quantum information encoded in the internal states of the atoms or ions. Trapped ion qubits are typically confined using radiofrequency or optical fields in ion traps. They offer extremely long coherence times and high-fidelity quantum operations. Quantum gates are performed using laser pulses to induce interactions between the ions. Similarly, neutral atoms, which are the subject of Chapter 3, are trapped in optical lattices or tweezers, and manipulated using laser light. One of the main advantages of these systems is the high degree of control over individual qubits, allowing for precise quantum operations. Additionally, trapped ions and neutral atoms can achieve high-fidelity measurements and entanglement, crucial for quantum computation and error correction. However, these systems face challenges such as the complexity of trapping and cooling the particles, and the scalability of the setups. Trapping and controlling a large number of ions or atoms in a stable manner is technically demanding. Furthermore, the requirement for sophisticated laser systems and vacuum chambers adds to the complexity and cost.

Advantages of Trapped Ions and Neutral Atoms

- **Long Coherence Times:** Trapped ions and neutral atoms exhibit long coherence times, reducing decoherence effects.
- **High-Fidelity Operations:** They enable high-fidelity quantum gates and measurements, essential for accurate quantum computation.
- **Precise Control:** Advanced trapping and manipulation techniques allow for precise control of individual qubits.
- **Strong Interaction:** The strong interaction between trapped ions or neutral atoms facilitates efficient entanglement and quantum operations.

Disadvantages of Trapped Ions and Neutral Atoms

- **Complex Setup:** Trapping and cooling systems are complex and require precise engineering.
- **Scalability:** Scaling up the number of qubits while maintaining stability and control is challenging.
- **Expensive Infrastructure:** The need for advanced laser systems, vacuum chambers, and cooling technologies increases the overall cost.
- **Slower Gates:** The gate durations are currently in the order of microseconds.

In summary, both trapped ions and neutral atoms offer unique advantages and face specific challenges. Continuous advancements in these technologies are essential for realizing practical and scalable QC. An example of a state-of-the-art neutral atom QC is Quera Aquila, 256 qubits (2023), whereas a trapped ion one is IONQ Forte, 36 qubits (2022).

B.5 Non-Universal Quantum Computing

Universal QC, such as those based on the gate model, can theoretically perform any computation that a classical computer can, and more, given enough time and resources. In contrast, non-universal quantum computing models are specialized and cannot efficiently simulate any arbitrary quantum algorithm. These models are designed to solve specific types of problems more efficiently than classical algorithms, rather than providing a general-purpose quantum computational framework. For instance, they may lack the full set of operations needed to perform arbitrary quantum computations but can still be extremely powerful for certain tasks. One notable example of a non-universal quantum computing approach is Quantum Annealing (QA), which focuses on solving optimization problems by exploiting quantum fluctuations.

B.5.1 Quantum Annealing

QA is a general method to find the global minimum of a given function in a set of candidates. The class of algorithmic methods for QA is a promising metaheuristic

tool for solving local search problems in multivariable optimization contexts. These problems usually consist in finding the maximum or minimum for a cost function that comprises several independent variables and a large number of instances [88]. The evaluation of cost in this context must necessarily be computed in probabilistic terms. A single configuration is defined as a 'tuple' of values over the whole set of independent variables. The value of the cost function depends on the configurations, being the solution to the problem set as the definite optimal configuration which minimizes, or maximizes, the cost function with some arbitrarily chosen confidence level or probability.

B.5.2 Implementation

To detail this architecture, it is useful to start with the qubits that are the lowest energy states of the superconducting loops that make up the D-Wave QPU. These states have a circulating current and a corresponding magnetic field. At the end of the QA process, each qubit collapses from a superposition state into either 0 or 1 (a classical state). A qubit's state is implemented in a circulating current with a corresponding magnetic field [89]. The physics of this process can be visualized with an energy diagram, as in Figure B.1.

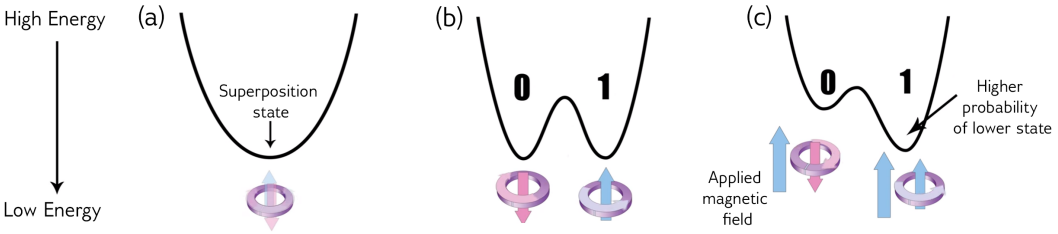


Fig. B.1 Energy diagram changes over time as the quantum annealing process runs, and a bias is applied [4].

This diagram changes over time: to begin, there is just one valley (a), with a single minimum. As the QA process runs, the barrier is raised, and this turns the energy diagram into what is known as a double-well potential (b). Here, the low point of the left valley corresponds to the 0 state, and the low point of the right valley corresponds to the 1 state. The qubit ends up in one of these valleys at the end of the anneal. Everything else being equal, the probability of the qubit ending in the 0 or the 1 state is equal ($\frac{1}{2}$). We can, however, control the probability of it falling

into the 0 or the 1 state by applying an external magnetic field to the qubit (c). This field tilts the double-well potential, increasing the probability of the qubit ending up in the lower well. The programmable quantity that controls the external magnetic field is called a bias, and the qubit minimizes its energy in the presence of the bias. The bias term alone is not useful, however. The real utility of the qubits comes when they are linked together so that they can influence each other. This is done with a device called a coupler. A coupler can make two qubits tend to end up in the same state, both 0 or both 1, or it can make them tend to be in opposite states. Like a qubit bias, the correlation weights between coupled qubits can be programmed by setting a coupling strength. Together, the programmable biases and weights are the means by which an optimization problem is defined in the D-Wave QC. As stated, each qubit has a bias and qubits interact via the couplers. When formulating a problem, users choose values for the biases and couplers. The biases and couplings define an energy landscape, and the D-Wave QC finds the minimum energy of that landscape. Currently, the state of the art is the D-Wave Advantage2 QPU, with approximately 4400 qubits (2025).

Appendix C

Neuro-Symbolic AI for Visual Reasoning

C.1 Introduction

While the core of this doctoral dissertation focuses on the convergence of Quantum Computing and Machine Learning, the research activity conducted during the Ph.D. cycle also explored parallel frontiers in Artificial Intelligence. Specifically, this appendix details a side research project concerning Neuro-Symbolic (NeSy) AI. This field addresses a fundamental challenge that is also relevant to quantum-enhanced models: the integration of low-level representation learning with high-level relational reasoning. In the following sections, we investigate how symbolic constraints can be enforced within a differentiable framework. The content of this chapter has been previously published by the author in [90].

C.1.1 The Neuro-Symbolic Paradigm and Logic Tensor Networks

The past decade has been defined by the success of deep learning (DL) in performing inductive tasks across various domains, including computer vision and natural language processing. However, purely data-driven strategies exhibit significant shortcomings, particularly when training data is scarce or when the model must comply with explicit external requirements, such as physical laws, safety regulations,

or logical axioms [91, 92]. To enhance the reliability and generalization of these systems, Neuro-Symbolic AI seeks to combine the strengths of statistical learning with the rigor of knowledge representation and reasoning [93, 94].

One of the most robust frameworks in this domain is the Logic Tensor Network (LTN) paradigm. LTNs utilize an infinitely-valued fuzzy logical language known as Real Logic, which extends classical first-order logic signatures (such as constant, function, and predicate symbols) into a differentiable environment. In this context, the term *grounding* refers to the mapping of these symbolic entities onto concrete tensors in the real field [95, 96]. By adopting fuzzy semantics, LTNs allow formulas to be partially true, enabling the training of neural networks under a set of logical constraints that act as regularizers or as part of the loss function.

C.1.2 Benchmark Investigation: Visual Sudoku Classification

This research explores the application of LTNs to the Visual Sudoku Puzzle Classification (ViSudo-PC) benchmark [97]. This task provides a sophisticated testbed for NeSy systems as it requires the simultaneous processing of visual perception (recognizing handwritten digits) and complex relational constraints (satisfying Sudoku rules). Unlike standard digit recognition, the classification task here is to determine whether a 9×9 grid of images constitutes a valid Sudoku solution, notably without providing the labels for individual digits during the reasoning phase.

A significant challenge of the ViSudo-PC benchmark is that the performance of a decoupled approach, where a Convolutional Neural Network (CNN) classifies digits followed by a hard-coded logic check, deteriorates rapidly if the perceptual module is not perfectly accurate [97]. Robust performance thus requires a system capable of joint reasoning, where the symbolic constraints can guide the perceptual module in resolving ambiguities. This appendix analyzes various configurations for logical constraint formulation and explores the integration between the neural perceptual layer and the LTN-based reasoning module, providing insights into the efficiency of hybrid learning procedures.

C.2 Methodology

C.2.1 Puzzle construction

A Sudoku "puzzle" or "board" consists of 9×9 grid, in which each cell is populated with digits 1–9. A puzzle is correctly solved if no row, column, or non-overlapping 3×3 subgrid (or "square") contains all the possible numbers without repetitions. The ViSudo-PC benchmark generalizes the concept of Sudoku puzzles assuming that the board is a square of dimension $m = n \times n$ (specifically, 4×4 and 9×9 grids are provided), and that symbols can come from arbitrary domains. It includes four domains of increasing complexity: digits (MNIST), English letters (EMNIST), fashion items (FashionMNIST), and Japanese characters (KMNIST). In the following, we will refer for simplicity and without loss of generality to the symbols as *digits*.

C.2.2 Common definitions

In this section, definitions of all domains, variables, and predicates are provided. The function $\mathbf{D}(\cdot)$ associates each variable with the corresponding domain, and $\mathbf{D_{in}}(\cdot)$ each predicate with the input domain.

Domains:

- **images**, denoting the images that form the individual cells of the Sudoku puzzle, encoded as 28×28 pixel maps,
- **sudoku**, denoting the image representing the Sudoku puzzle,
- **results**, denoting the Boolean value that indicates the validity of the Sudoku board,
- **digits**, denoting the digits from 0 to 9,
- **coordinates**, denoting the coordinates of each cell in a Sudoku puzzle.

Variables:

- **x** ranging over the list of images $[x_{0,0},...,x_{i,j},...,x_{m,m}]$ associated to each Sudoku board, where $x_{i,j}$ with $i \in [0,m-1]$, $j \in [0,m-1]$ are the images at position (i,j), with $\mathbf{D}(x_{i,j}) = \text{images}$;
- **d** ranging over the list of digits $[d_{0,0},...,d_{i,j},...,d_{m,m}]$ associated to each Sudoku board, where $d_{i,j}$, with $i \in [0,m]$, $j \in [0,m]$ represents the digit at position (i,j), with $\mathbf{D}(d_{i,j}) = \text{digits}$;
- **se** ranging over all sub-elements of each Sudoku, that is rows, columns, and squares, each formed by a sequence of coordinates $\{i,j\}$;
- **S** for the $m \times m$ Sudoku puzzles in the data, with $\mathbf{D}(S) = \text{sudoku}$;
- $\mathbf{S}_+$ and $\mathbf{S}_-$ for the correct and incorrect Sudoku puzzles, respectively;
- **l** for the labels, i.e., the validity of the Sudoku, $\mathbf{D}(l) = \text{results}$.

Predicates:

- $\textbf{digit}(x,d)$ is a digit classifier, where d is a term denoting a digit constant or a digit variable. The classifier should return the probability of an image x being of digit d, with $\mathbf{D_{in}}(\text{digit}) = \text{images}, \text{digits}$;
- $\textbf{equal}(x_{i,j},x_{k,l})$ is a predicate indicating whether two images $x_{i,j}$ and $x_{k,l}$ contain the same digit, with $\mathbf{D_{in}}(\text{equal}) = \text{images}, \text{images}$;
- $\textbf{valid}(S)$ denotes whether a Sudoku board S is valid. The predicate should return the probability that the image represents a valid Sudoku solution, with $\mathbf{D_{in}}(\text{valid}) = \text{sudoku}$;
- $\textbf{validElement}(se)$ denotes whether a sub-element of a Sudoku board S is valid.

Fuzzy operators:

- **Diagonal quantification** $\text{Diag}(x,\ldots,l)$ quantifies over tuples that combine the i-th instance of each of the variables in the argument of Diag [96]. For instance, given a data set with samples x and target labels y, $\forall\text{Diag}(x,y)$ quantifies over each label, sample pair.

C.2.3 Knowledge base definition

At its core, an LTN-based NeSy approach to the ViSudo-PC task can be constructed by combining one or more CNNs, that recognize digits and/or classify the whole Sudoku board, with a LTN that enforces the logical constraints given by the rules of Sudoku. Multiple solutions are available depending on which predicates are defined and how they are combined in the knowledge base. The first group of solutions (denoted in the following as *indirect* solutions) verifies whether the detected digits constitute a valid Sudoku solution by means of a non-trainable predicate that enforces the rules of Sudoku. The network learns to detect the digits via a learnable $\text{digit}(x,d)$ predicate. At inference time a fuzzy or crisp validity score can be computed applying the rules of Sudoku using real or standard logic. The second group of solutions (denoted in the following as *direct* solutions) computes instead the validity of the entire Sudoku board via the $\text{valid}(S)$ predicate. An auxiliary $\text{digit}(x,d)$ predicate is used to detect digits and enforce the rules of Sudoku. Both predicates $digit(x,d)$ and $valid(S)$ are grounded by CNNs. In all cases, we assume that labels for each individual digit are not available, and that digit classification must be trained in a semi-supervised fashion exclusively by enforcing the rules of Sudoku, as mandated by the ViSudo-PC task [97].

Indirect solution #1

This approach is based on the observation that every sub-element (row, column, and square) belonging to a correct Sudoku is also correct. Conversely, if all the sub-elements are correct, the Sudoku is correct as well. On the other hand, if the Sudoku is not correct, at least one wrong sub-element must exist. We further observe that a Sudoku sub-element is correct if, and only if, all available digits are present. Indeed, since the length of a sub-element is equal to the number of digits, if one is repeated twice, then another must be missing. Hence, the problem can be reduced to the simpler problem of detecting whether a sub-element is correct or not.

Axioms:

$$\forall \mathbf{se} \left(\left(\forall d \; \exists x_{i,j} : \{i,j\} \in \mathbf{se} \; (\text{digit}(x_{i,j},d)) \right) \iff \text{validElement}(\mathbf{se}) \right) \quad \text{(C.1)}$$

$$\forall \text{Diag}(S,l)\ \Big(l \iff \big(\forall \mathbf{se} : \mathbf{se} \in s\ \text{validElement}(\mathbf{se})\big)\Big) \tag{C.2}$$

$$\forall \text{Diag}(S,l)\ \Big(\neg l \iff \big(\exists \mathbf{se} : \mathbf{se} \in s\ \neg\text{validElement}(\mathbf{se})\big)\Big) \tag{C.3}$$

Grounding:

The digit predicate is grounded by a CNN taking as input each digit image. The validElement predicate is a non-trainable predicate that can be computed directly using the axiom in Eq. C.1.

Indirect solution #2

This solution is based on the observation that a given sub-element (row, column, or square) cannot contain the same symbol twice. More generally, given all possible image pairs within a Sudoku board, it is possible to define an additional predicate sameSubelement($[p_1, p_2]$) that determines whether the image pair at positions $p_1 = (i, j)$ and $p_2 = (l,k)]$ belong to the same sub-element and hence cannot contain the same digit (or, more generally, the same symbol). Based on this principle, two axioms can be formulated, one for correct Sudoku boards and one for incorrect Sudoku boards. In the experiments, only correct examples were used.

Axioms:

$$\forall S_+\ \forall([p_1,p_2])\ \Big(\text{sameSubelement}(p_1,p_2) \implies \neg\text{equal}(x_{p_1},x_{p_2})\Big) \tag{C.4}$$

$$\forall S_-\ \exists([p_1,p_2])\ \Big(\text{sameSubelement}(p_1,p_2) \implies \text{equal}(x_{p_1},x_{p_2})\Big) \tag{C.5}$$

Grounding:

The equal(x_1,x_2) predicate is grounded by a function of the probabilities that x_1 and x_2 belong to each digit, computed by the digit(x) predicate::

$$\text{equal}(x_1,x_2) = exp\Big(-ReLU(\|\text{digit}(x_1) - \text{digit}(x_2)\| - c)\Big) \tag{C.6}$$

where c is a constant added for numerical stability. Alternative grounding formulations could compute the distance i) based only on the most probable digits or ii) directly from the image embedding, e.g., before computing the softmax.

Indirect solution #3

This solution stems from the observation that the same number can appear only once for each sub-element (row, column, and square). Hence, if we denote as $x_0, ..., x_k, ..., x_n$ the list of images in sub-element *se*, the following knowledge base can be defined:

Axioms:

$$\forall S_+ \, \forall \mathbf{se} \, \forall x_1, x_2, d : (x1 \in \mathbf{se} \wedge x2 \in \mathbf{se}) \left(x_1 \neq x_2 \wedge \text{digit}(x_1, d) \implies \neg \text{digit}(x_2, d) \right) \tag{C.7}$$

$$\forall S_- \, \forall \mathbf{se} \, \exists x_1, x_2, d : (x1 \in \mathbf{se} \wedge x2 \in \mathbf{se}) \left(x_1 \neq x_2 \wedge \text{digit}(x_1, d) \implies \text{digit}(x_2, d) \right) \tag{C.8}$$

Direct solution

The direct solutions involve two predicates, valid to compute the validity of the Sudoku board and digit to compute the probability that each cell contains a given digit. The latter predicate is used only to enforce Sudoku rules. At inference time, predictions are directly calculated from the valid predicate. While the digit predicate is trained in a semi-supervised fashion, to comply with the training setting specified by the ViSudo-PC benchmark, the valid predicate can be trained in a supervised fashion. In addition, additional axioms are specified to encode prior knowledge about the rules of Sudoku. Any of the axioms that were defined in previous solutions can be used for this purpose: only one possible solution is shown here.

Axioms:

$$\forall S_+ \text{valid}(S), \forall S_- \neg \text{valid}(S) \tag{C.9}$$

$$\forall S \left(\left(\forall([p_1,p_2])(\text{sameSubelement}(p_1,p_2) \implies \neg\text{equal}(x_{p_1},x_{p_2})) \right) \Leftrightarrow \text{valid}(S) \right) \tag{C.10}$$

$$\forall S \left(\left(\exists([p_1,p_2])(\text{sameSubelement}(p_1,p_2) \implies \text{equal}(x_{p_1},x_{p_2})) \right) \Leftrightarrow \neg\text{valid}(S) \right) \tag{C.11}$$

Grounding: The equal(x_1,x_2) predicate is grounded as in Eq. C.6. The two predicates valid(S) and digit(x,d) are grounded by either two separate CNNs or by one CNNs with a common backbone and two different prediction heads.

Potential pitfalls

Alternative solutions could be designed by extending the LTN for the semi-supervised MNIST classification task proposed in [96], in which MNIST classification is learned by solving single- and multi-digit additions:

$$\forall Diag(x_1,x_2,y_1,y_2,n)\ (\exists d_1,d_2,d_3,d_4 : 10d_1+d_2+10d_3+d_4=n. \\ (digit(x_1,d_1) \wedge digit(x_2,d_2) \wedge digit(y_1,d_3) \wedge digit(y_2,d_4))) \tag{C.12}$$

A similar approach, in this case, would yield the following solution:

$$\forall Diag(x_{0,0},...,x_{i,j},...,,x_{m,m},l)\ (\exists d_{0,0},...,d_{i,j},...,,d_{m,m} : \\ \text{validPuzzle}(d_{0,0},...,d_{i,j},...,,d_{m,m}) = \\ = l.(digit(x_{0,0},d_{0,0}) \wedge ... \wedge digit(x_{i,j},d_{i,j}) \wedge ... \wedge digit(x_{m,m},d_{m,m}))) \tag{C.13}$$

where validPuzzle determines if the specific sequence of digits yields a Sudoku board consistent with the label (i.e., valid or invalid). Empirically, we found that solutions involving up to $m \times m$ multiple $\wedge$ operations in a single axiom led to exploding memory issues, possibly due to the LTNtorch implementation [98]. The proposed solutions are based on simpler formulas, each implementing a separate logical constraint for pairs of digits at a time, aggregated using existential and universal quantifiers. However, given a $m \times m$ board, with m rows, m columns, and $\sqrt{m}$ squares, the number of possible digit pairs with each sub-element is $\binom{m}{2} = \frac{N(N-1)}{2}$, hence the number of constraints increases as m^3.

Extension to digit localization

A near-perfect accuracy in digit classification is paramount to determining whether a given Sudoku puzzle is correctly solved. The solutions proposed in the previous section rely heavily on the $digit(x,d)$ and $equal(x_{p_1},x_{p_2})$ predicates, and assume that the image grid is known. This limitation could be overcome by integrating an Object Detection method within the NeSy framework. In this way, the grounding of the sub-images can be extended by including the bounding box coordinates, which would be used to localize the digits within the board. The digit(S,b,d) predicate would take as input the Sudoku Board S and a bounding box b to predict the probability that the bounding box b contains the digit d, and would be grounded by an object detector. In a preliminary implementation, we used the You Only Look Once (YOLO) [99] algorithm, specifically the YOLOv1 architecture. YOLOv1 consists of two main parts: *i)* the feature extractor and the *ii)* detection network. The latter is a fully connected layer that, from the output of the feature extraction, generates a set of bounding boxes that contain the detected objects in the image. The feasibility of training an object detector and a LTN end-to-end has been established in [100].

C.3 Exploratory assessment

C.3.1 Dataset

Preliminary experiments were conducted on the basic dataset provided by the ViSudo-PC benchmark, which includes 10 splits per dataset to be used for scoring [97][1]. We performed experiments on the 4×4 Sudoku with data sources MNIST, EMNIST, FMNIST, and KMNIST. Each split contains 50/100/100 puzzles for training/validation/test, respectively. Performances (Area under the ROC curve) are reported in cross-validation by averaging across the 10 splits.

[1]The dataset was downloaded from https://github.com/linqs/visual-sudoku-puzzle-classification

C.3.2 Experimental settings

A total of three configurations were tested, three variants of the indirect solution #2 introduced in Section C.2.3, denoted in the following as LTN_IND_A, LTN_IND_B and LTN_IND_C. Preliminary experiments were also conducted on the direct solution (LTN_DIR). The indirect solution was tested with and without negative examples (Eq. C.11) in the knowledge base. At inference time, the probability of a correct Sudoku is computed using the same axiom for positive example (Eq. C.10) for the indirect solutions, whereas for the direct solution it is directly computed by the valid predicate.

The digit CNN consists of 2 convolutional layers with a kernel size of 5, each followed by a max pooling layer of size 2 with stride 2. The latter feeds into fully connected (FC) dense layers of sizes 320, 50 with ReLU activation and a final softmax layer. For the direct solution, two output branches of sizes 320, 50 and 5120, 320 computes the Valid and digit predicate, respectively.

We approximate the universal quantifier with the generalized mean w.r.t. the error aggregator $A_{pME} = 1 - \left(\frac{1}{n} \sum_{i=1}^{n} (1 - a_i)^{p_\forall} \right)^{\frac{1}{p_\forall}}$ and the existential quantifier with the generalized mean aggregator $A_{pM} = \left(\frac{1}{n} \sum_{i=1}^{n} a_i^{p_\exists} \right)^{\frac{1}{p_\exists}}$, as suggested by Badreddine et al. [96]. We set $p_\exists = 1.5$, whereas for the universal quantifier both fixed $p_\forall = 2$ (LTN_IND_A and LTN_IND_C) and scheduled values (LTN_IND_B, LTN_DIR) were evaluated. In the latter, $p_\forall$ is gradually increased as follows: 1 (epochs 0–20), 2 (20–120), 6 (120–170), 8 (170–200), and 10 (200–500). Higher p values at the beginning of training prevented the network from converging, as previously reported [96]. For inference $p_\forall$ was set to 1000. All networks were trained with batch size 8 and the Adam optimizer. Learning rate and number of epochs were set experimentally for each configuration (LTN_IND_A: 0.001/200; LTN_IND_B: 0.0005/300; LTN_IND_C: 0.0005/500; LTN_DIR:0.0005/300). The LTN was implemented in PyTorch using the LTNtorch package [98].

C.3.3 Results

Results for the indirect solutions are presented in Table C.1. LTN–based solutions perform best for the simpler dataset (MNIST). As observed in [96], LTNs appear

Data source	NeuPSL [97]	LTN_IND_A	LTN_IND_B	LTN_IND_C
MNIST4	0.88 ± 0.02	0.83 ± 0.18	0.84 ± 0.14	0.94 ± 0.10
EMNIST4	0.79 ± 0.09	0.58 ± 0.04	0.58 ± 0.06	0.65 ± 0.14
FMNIST4	0.74 ± 0.04	0.67 ± 0.11	0.76 ± 0.15	0.83 ± 0.11
KMNIST4	0.65 ± 0.12	0.83 ± 0.09	0.85 ± 0.11	0.87 ± 0.09

Table C.1 Performance of the indirect LTN solution on the 4×4 Sudoku. The mean area under the receiver operating characteristic curve (AuROC) along with the standard deviation over 10 splits is reported. The tested LTN versions differ as follows: LTN_IND_A: positive examples only and fixed $p_\forall$;LTN_IND_B: positive examples only and varying $p_\forall$; LTN_IND_C: all axioms and fixed $p_\forall$.

sensitive to random initialization and may occasionally fail to converge to a non-trivial solution, which explains high standard deviations for most configurations. Compared to the baseline LTN (LTN_IND_A), stability and performance are enhanced by scheduling p in the universal aggregator [96], as well as by including axioms for negative examples (LTN_IND_C). Compared to NeuPSL [97], LTN–based solution appear to perform better on the KMNIST domain, and worse on the EMNIST domain; it is possible that, when samples are more difficult to classify, predicting whether two digits are equal is a more robust alternative than comparing the actual classifications, and thus the different may be also attributed to the choice of knowledge base.

The direct solution does not achieve performance above random guess on MNIST4 (0.52 ± 0.02) and was not tested on other domains. However, When computing predictions using the digit predicate and rules of Sudoku, performance increases (0.85 ± 0.02). Hence, the LTN learns to distinguish digits, but fails to predict the validity directly from the input image, possibly due to the specific choice of grounding.

C.4 Summary

In this chapter, multiple LTN-based solutions to the ViSudo-PC benchmark were discussed. While experimental validation is still preliminary, we observe that several axioms could be used to encode the rule of Sudoku. Although logically equivalent, different solutions could result in different numerical properties, hindering comparison across different implementations and scenarios. Further experiments are needed

to fully characterize all possible LTN-based solutions, as well as to optimize their grounding and increase stability to random initialization [96].

References

[1] Yunseok Kwak, Won Joon Yun, Jae Kim, Hyunhee Cho, Minseok Choi, Soyi Jung, and Joongheon Kim. Quantum heterogeneous distributed deep learning architectures: Models, discussions, and applications. 02 2022.

[2] Bochen Tan and Jason Cong. Optimality study of existing quantum computing layout synthesis tools. IEEE Transactions on Computers, 70(9):1363–1373, sep 2021.

[3] Qiskit. Training Parametrized Quantum Circuits. https://learn.qiskit.org/course/machine-learning/training-quantum-circuits, 2024.

[4] Hadi Salloum, Mohammad Reza Bahrami, Ahmad Haidar, Osama Orabi, Hamza Shafee Aldaghstany, and Manuel Mazzara. Integration of Machine Learning with Quantum Annealing, page 338–348. 04 2024.

[5] Marco Russo and Bartolomeo Montrucchio. From classical to quantum machine learning: A comprehensive review. In Proceedings of the 3rd International Conference on Frontiers of Artificial Intelligence, Ethics, and Multidisciplinary Applications (FAIEMA 2025). Springer, October 2025.

[6] K. Zaman, A. Marchisio, M.A. Hanif, and M. Shafique. A survey on quantum machine learning: Current trends, challenges, opportunities, and the road ahead. arXiv preprint, 2023.

[7] L. Chen, T. Li, Y. Chen, X. Chen, M. Wozniak, N. Xiong, and W. Liang. Design and analysis of quantum machine learning: a survey. Connection Science, 2024.

[8] OVHcloud. What is quantum machine learning? https://us.ovhcloud.com/learn/what-is-quantum-machine-learning/, 2025. Last accessed 2025-06-14.

[9] D. Gopalakrishnan, L. Dellantonio, A. Di Pilato, W. Redjeb, F. Pantaleo, and M. Mosca. qclue: A quantum clustering algorithm for multi-dimensional datasets. Frontiers in Quantum Science and Technology, 3:1462004, 2024.

[10] Quantropi. Quantum versus classical computing and the quantum threat. https://www.quantropi.com/quantum-versus-classical-computing-and-the-quantum-threat/, 2025. Last accessed 2025-06-14.

[11] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, Cambridge, 10th anniversary edition edition, 2010.

[12] M. Schuld and N. Killoran. Quantum machine learning in feature hilbert spaces. Physical Review Letters, 122, 2019.

[13] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. Physical Review A, 103(3), March 2021.

[14] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. Nature, 549(7671):195–202, 2017.

[15] S. Thudumu, J. Fisher, and H. Du. Supervised quantum machine learning: A future outlook from qubits to enterprise applications. arXiv preprint, 2025.

[16] G. Gentinetta, A. Thomsen, D. Sutter, and S. Woerner. The complexity of quantum support vector machines. Quantum, 2024.

[17] M. Russo, E. Giusto, and B. Montrucchio. Quantum kernel estimation with neutral atoms for supervised classification: A gate-based approach. In 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), pages 219–228, 2023.

[18] M. Kashif, A. Marchisio, and M. Shafique. Computational advantage in hybrid quantum neural networks: Myth or reality? arXiv preprint, 2025.

[19] Fiveable. Quantum k-means algorithm. https://library.fiveable.me/quantum-machine-learning/unit-13/quantum-k-means-algorithm/study-guide/BtwdNnQ3uBT3fwRG, 2025. Last accessed 2025-06-14.

[20] S S Kavitha and Narasimha Kaulgud. Quantum k-means clustering method for detecting heart disease using quantum circuit approach. Soft Computing, 27(18):13255–13268, May 2022.

[21] Zhihao Wu, Tingting Song, and Yanbing Zhang. Quantum k-means algorithm based on manhattan distance. Quantum Information Processing, 21(1), December 2021.

[22] D. Herman, C. Googin, X. Liu, et al. Quantum computing for finance. Nature Reviews Physics, 5:450–465, 2023.

[23] Michael Marzec. Portfolio optimization: Applications in quantum computing, April 2016.

[24] Sean J. Weinberg, Fabio Sanches, Takanori Ide, Kazumitzu Kamiya, and Randall Correll. Supply chain logistics with quantum and classical annealing algorithms. Scientific Reports, 13(1), March 2023.

[25] A.M. Smaldone, Y. Shee, G.W. Kyro, C. Xu, M.H. Farag, A. Galda, S. Kumar, E. Kyoseva, and V.S. Batista. Quantum machine learning in drug discovery: Applications in academia and pharmaceutical industries. Chemical Reviews, 2024.

[26] Erico Souza Teixeira, Lucas Barros Fernandes, and Yara Rodrigues Inácio. Quantum neural network applications to protein binding affinity predictions, 2025.

[27] R.S. Gupta, C.E. Wood, T. Engstrom, J.D. Pole, and S. Shrapnel. A systematic review of quantum machine learning for digital health. npj Digital Medicine, 8(1):237, 2025.

[28] S. Suhas and S. Divya. Quantum-improved weather forecasting: Integrating quantum machine learning for precise prediction and disaster mitigation. In 2023 International Conference on Quantum Technologies, Communications, Computing, Hardware and Embedded Systems Security (iQ-CCHESS), pages 1–7, 2023.

[29] QuantumGrad. The nisq era of quantum computing: Challenges. https://www.quantumgrad.com/article/733, 2025. Last accessed 2025-06-14.

[30] Stanford University. 2025 stanford emerging technology review. Technical report, Stanford University, 2025.

[31] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. Quantum Science and Technology, 2(4):045001, August 2017.

[32] J. Cunningham and J. Zhuang. Investigating and mitigating barren plateaus in variational quantum circuits: A survey. Quantum Information Processing, 24:48, 2025.

[33] Maria Schuld and Francesco Petruccione. Machine Learning with Quantum Computers. Springer International Publishing, 2021.

[34] Riccardo Mengoni and Alessandra Di Pierro. Kernel methods in quantum machine learning. Quantum Machine Intelligence, 1(3-4):65–71, November 2019.

[35] Karol Bartkiewicz, Clemens Gneiting, Antonín Černoch, Kateřina Jiráková, Karel Lemr, and Franco Nori. Experimental kernel-based quantum machine learning in finite feature space. Scientific Reports, 10(1), July 2020.

[36] Maria Schuld and Nathan Killoran. Is quantum advantage the right goal for quantum machine learning? PRX Quantum, 3(3), jul 2022.

[37] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. 567(7747):209–212. Number: 7747 Publisher: Nature Publishing Group.

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

[39] Boris Albrecht, Constantin Dalyac, Lucas Leclerc, Luis Ortiz-Gutiérrez, Slimane Thabet, Mauro D'Arcangelo, Vincent E. Elfving, Lucas Lassablière, Henrique Silvério, Bruno Ximenez, Louis-Paul Henry, Adrien Signoles, and Loïc Henriet. Quantum feature maps for graph machine learning on a neutral atom quantum processor, 2022.

[40] Meet quera, Aug 2022.

[41] 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. arXiv:2104.15044 [quant-ph].

[42] X. L. Zhang, L. Isenhower, A. T. Gill, T. G. Walker, and M. Saffman. Deterministic entanglement of two neutral atoms via rydberg blockade. Physical Review A, 82(3), September 2010.

[43] T. Wilk, A. Gaëtan, C. Evellin, J. Wolters, Y. Miroshnychenko, P. Grangier, and A. Browaeys. Entanglement of two individual neutral atoms using rydberg blockade. Physical Review Letters, 104(1), January 2010.

[44] T. M. Graham, M. Kwon, B. Grinkemeyer, Z. Marra, X. Jiang, M. T. Lichtman, Y. Sun, M. Ebert, and M. Saffman. Rydberg-mediated entanglement in a two-dimensional neutral atom qubit array. Physical Review Letters, 123(23), December 2019.

[45] Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: From theory to algorithms. Cambridge university press, 2014.

[46] Core Features; Pulser 0.11.1 documentation — pulser.readthedocs.io. https://pulser.readthedocs.io/en/stable/apidoc/core.html#physical-devices. [Accessed 30-Apr-2023].

[47] J.R. Johansson, P.D. Nation, and Franco Nori. Qutip 2: A python framework for the dynamics of open quantum systems. Computer Physics Communications, 184(4):1234–1240, 2013.

[48] Marco Russo. QKE-Neutral-Atoms: Source code for quantum kernel estimation on neutral atom devices. https://github.com/marcorusso97/QKE-Neutral-Atoms, 2026.

[49] Dmitri Maslov, Yunseong Nam, and Jungsang Kim. An outlook for quantum computing [point of view]. Proceedings of the IEEE, 107:5–10, 1 2019.

[50] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. Phys. Rev. A, 76:042319, Oct 2007.

[51] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. Phys. Rev. A, 86:032324, Sep 2012.

[52] Sergey Bravyi, Oliver Dial, Jay M. Gambetta, Darío Gil, and Zaira Nazario. The future of quantum computing with superconducting qubits. Journal of Applied Physics, 132(16):160902, 10 2022.

[53] James Ang, Gabriella Carini, Yanzhu Chen, Isaac Chuang, Michael Austin DeMarco, Sophia E. Economou, Alec Eickbusch, Andrei Faraon, Kai-Mei Fu, Steven M. Girvin, Michael Hatridge, Andrew Houck, Paul Hilaire, Kevin Krsulich, Ang Li, Chenxu Liu, Yuan Liu, Margaret Martonosi, David C. McKay, James Misewich, Mark Ritter, Robert J. Schoelkopf, Samuel A. Stein, Sara Sussman, Hong X. Tang, Wei Tang, Teague Tomesh, Norm M. Tubman, Chen Wang, Nathan Wiebe, Yong-Xin Yao, Dillon C. Yost, and Yiyu Zhou. Architectures for multinode superconducting quantum computers, 2022.

[54] David C. McKay, Christopher J. Wood, Sarah Sheldon, Jerry M. Chow, and Jay M. Gambetta. Efficient z gates for quantum computing. Phys. Rev. A, 96:022330, Aug 2017.

[55] N. Anders Petersson and Fortino Garcia. Optimal control of closed quantum systems via b-splines with carrier waves, 2022.

[56] Thomas Alexander, Naoki Kanazawa, Daniel J Egger, Lauren Capelluto, Christopher J Wood, Ali Javadi-Abhari, and David C McKay. Qiskit pulse: programming quantum computers through the cloud with pulses. Quantum Science and Technology, 5(4):044006, aug 2020.

[57] 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.

[58] Cirq Developers. Cirq, December 2023.

[59] David Xu, A. Barls Ozguler, Giuseppe Di Guglielmo, Nhan Tran, Gabriel N. Perdue, Luca Carloni, and Farah Fahim. Neural network accelerator for quantum control. In 2022 IEEE/ACM Third International Workshop on Quantum Computing Software (QCS), nov 2022.

[60] Joel J Wallman and Steven T Flammia. Randomized benchmarking with confidence. New Journal of Physics, 16(10):103032, oct 2014.

[61] Marina Kudra, Mikael Kervinen, Ingrid Strandberg, Shahnawaz Ahmed, Marco Scigliuzzo, Amr Osman, Daniel Pérez Lozano, Mats O. Tholén, Riccardo Borgani, David B. Haviland, Giulia Ferrini, Jonas Bylander, Anton Frisk Kockum, Fernando Quijandría, Per Delsing, and Simone Gasparinetti. Robust preparation of wigner-negative states with optimized snap-displacement sequences. PRX Quantum, 3:030301, Jul 2022.

[62] Xinyuan You, Yunwei Lu, Taeyoon Kim, Doga Murat Kurkcuoglu, Shaojiang Zhu, David van Zanten, Tanay Roy, Yao Lu, Srivatsan Chakram, Anna Grassellino, Alexander Romanenko, Jens Koch, and Silvia Zorzetti. Crosstalk-robust quantum control in multimode bosonic systems, 2024.

[63] Zhubing Jia, Shilin Huang, Mingyu Kang, Ke Sun, Robert F. Spivey, Jungsang Kim, and Kenneth R. Brown. Angle-robust two-qubit gates in a linear ion crystal. Phys. Rev. A, 107:032617, Mar 2023.

[64] Andrey V. Rodionov, Andrzej Veitia, R. Barends, J. Kelly, Daniel Sank, J. Wenner, John M. Martinis, Robert L. Kosut, and Alexander N. Korotkov. Compressed sensing quantum process tomography for superconducting quantum gates. Physical Review B, 90(14), October 2014.

[65] Muhammad AbuGhanem. Full quantum process tomography of a universal entangling gate on an ibm's quantum computer, 2024.

[66] Ze Li, Ming-Jie Liang, and Zheng-Yuan Xue. Time-optimal universal quantum gates on superconducting circuits. Physical Review A, 108(4), October 2023.

[67] E. Onorati, A. H. Werner, and J. Eisert. Randomized benchmarking for individual quantum gates. Physical Review Letters, 123(6), August 2019.

[68] M.N. Bhat, M. Russo, L.P. Carloni, G. Di Guglielmo, F. Fahim, A.C.Y. Li, and G.N. Perdue. Machine learning for arbitrary single-qubit rotations on an embedded device. Quantum Machine Intelligence, 7:8, 2025.

[69] David Isaac Schuster. Circuit Quantum Electrodynamics. PhD thesis, Yale University, 2007.

[70] Blake Robert Johnson. Controlling photons in superconducting electrical circuits. PhD thesis, 2011.

[71] Madhav Bhat, Marco Russo, L.P. Carloni, G. Di Guglielmo, A.C.Y. Li, and G.N Perdue. ml-qubit-control. https://github.com/mb4989/ml-qubit-control, 2024.

[72] Claudionor N Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. Nature Machine Intelligence, 3(8):675–686, 2021.

[73] FastML Team. hls4ml. https://github.com/fastmachinelearning/hls4ml, 2024.

[74] J. Duarte et al. Fast inference of deep neural networks in FPGAs for particle physics. Journal of Instrumentation, 13(07), jul 2018.

[75] Farah Fahim et al. hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices. In Proc. tinyML Research Symposium, 2021.

[76] Flex Logix Technologies, Inc. Flex logix efpga technology. https://www.flex-logix.com/, 2024.

[77] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control, 37(3):332–341, 1992.

[78] Enrico Rebufello, Fabrizio Piacentini, Alessio Avella, Rudi Lussana, Federica Villa, Alberto Tosi, Marco Gramegna, Giorgio Brida, Eliahu Cohen, Lev Vaidman, Ivo Pietro Degiovanni, and Marco Genovese. Protective measurement—a new quantum measurement paradigm: Detailed description of the first realization. Applied Sciences, 11, May 2021.

[79] Priyasheel Prasad, Marco Russo, and Bartolomeo Montrucchio. Simulating quantum dynamics of single-photon experiments. Discover Applied Sciences, 8(3):265, feb 2026.

[80] Yakir Aharonov, David Z. Albert, and Lev Vaidman. How the result of a measurement of a component of the spin of a spin-1/2particle can turn out to be 100. Physical Review Letters, 60:1351–1354, April 1988.

[81] Guth. J. v. Neumann, Mathematische Grundlagen der Quantenmechanik: (Grundlehren der mathematischen Wissenschaften, Bd. 38.) J. Springer, Berlin 1932. 262 Seiten. Preis geh. RM18, volume 40. Springer Science and Business Media LLC, December 1933.

[82] K. J. Resch and A. M. Steinberg. Extracting joint weak values with local, single-particle measurements. Physical Review Letters, 92, March 2004.

[83] F. Piacentini, A. Avella, M. P. Levi, R. Lussana, F. Villa, A. Tosi, F. Zappa, M. Gramegna, G. Brida, I. P. Degiovanni, and M. Genovese. Experiment investigating the connection between weak values and contextuality. Physical Review Letters, 116, May 2016.

[84] J.J. Sakurai and Jim J. Napolitano. Modern Quantum Mechanics. Pearson, Harlow, Essex, England, 2nd edition, 2014.

[85] Giuliano Benenti and Giuliano Strini. Quantum simulation of the single-particle schrödinger equation. American Journal of Physics, 76:657–662, July 2008.

[86] Quantum Information and Computation, (11-12), November 2012.

[87] David A. Kreplin and Marco Roth. Reduction of finite sampling noise in quantum neural networks. Quantum, 8:1385, June 2024.

[88] Alfonso de la Fuente Ruiz. Quantum Annealing. https://arxiv.org/pdf/1404.2465, 2014.

[89] D-Wave. How Quantum Annealing Works in D-Wave QPUs. https://docs.dwavesys.com/docs/latest/c_gs_2.html#how-quantum-annealing-works-in-d-wave-qpus.

[90] Lia Morra, Alberto Azzari, Letizia Bergamasco, Marco Braga, Luigi Capogrosso, Federico Delrio, Giuseppe Di Giacomo, Simone Eiraudo, Giorgia Ghione, Rocco Giudice, Alkis Koudounas, Luca Piano, Daniele Rege Cambrin, Matteo Risso, Marco Rondina, Alessandro Sebastian Russo, Marco Russo, Francesco Taioli, Lorenzo Vaiani, and Chiara Vercellino. Designing logic tensor networks for visual sudoku puzzle classification. In Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy), pages 223–232, May 2023.

[91] Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, et al. Toward trustworthy AI development: mechanisms for supporting verifiable claims. arXiv preprint arXiv:2004.07213, 2020.

[92] Eleonora Giunchiglia, Mihaela Catalina Stoian, and Thomas Lukasiewicz. Deep learning with logical constraints. In IJCAI, 2022.

[93] Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neural-symbolic artificial intelligence. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pages 4943–4950, 2021.

[94] Pascal Hitzler. Neuro-symbolic artificial intelligence: The state of the art. IOS Press, 2022.

[95] Luciano Serafini and Artur S. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. ArXiv, abs/1606.04422, 2016.

[96] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. Artificial Intelligence, 303:103649, feb 2022.

[97] Eriq Augustine, Connor Pryor, Charles Dickens, Jay Pujara, William Yang Wang, and Lise Getoor. Visual sudoku puzzle classification: A suite of collective neuro-symbolic tasks. In International Workshop on Neural-Symbolic Learning and Reasoning (NeSy), Windsor, United Kingdom, 2022.

[98] Tommaso Carraro. LTNtorch: PyTorch implementation of Logic Tensor Networks, mar 2022.

[99] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.

[100] Francesco Manigrasso, Filomeno Davide Miro, Lia Morra, and Fabrizio Lamberti. Faster-LTN: a neuro-symbolic, end-to-end object detection architecture. In Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30, pages 40–52. Springer, 2021.