TU WIEN Informatics

A Hybrid Quantum-Classical Framework for Reinforcement Learning of Atari Games

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Computational Science and Engineering

eingereicht von

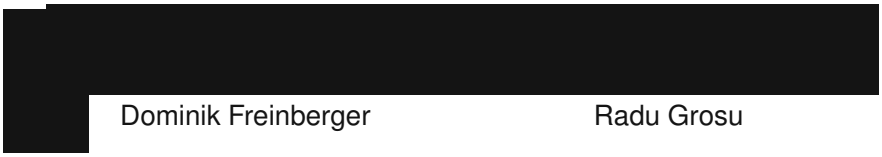
Dominik Freinberger, BSc
Matrikelnummer 11708140

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Dr.h.c. Radu Grosu
Mitwirkung: Dr. Sofiène Jerbi
Univ.Ass. Dipl.-Ing. Julian Lemmel

Wien, 7. Juni 2024



Dominik Freinberger Radu Grosu

TU WIEN Informatics

A Hybrid Quantum-Classical Framework for Reinforcement Learning of Atari Games

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computational Science and Engineering

by

Dominik Freinberger, BSc
Registration Number 11708140

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Dr.h.c. Radu Grosu
Assistance: Dr. Sofiène Jerbi
Univ.Ass. Dipl.-Ing. Julian Lemmel

Vienna, June 7, 2024

Dominik Freinberger Radu Grosu

Erklärung zur Verfassung der Arbeit

Dominik Freinberger, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 7. Juni 2024

Dominik Freinberger

Acknowledgements

I would like to thank Sofiène Jerbi and Julian Lemmel for their professional guidance and continuous support during the research and composition of this thesis. I would also like to thank Radu Grosu for giving me the opportunity to carry out this research within the Cyber-Physical Systems research unit at the Institute of Computer Engineering.

The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

Kurzfassung

Quantum Machine Learning (QML) ist ein vielversprechendes Anwendungsgebiet für aktuelle Quantencomputer, wobei hybride quantenmechanisch-klassische Modelle, basierend auf parametrisierten Quantenschaltkreisen (PQCs), als erfolgversprechende Algorithmen gelten. Jüngste Fortschritte im Bereich des Quantum Reinforcement Learning (QRL) haben das Potenzial von PQCs als Alternative zu klassischen Deep-Learning Modellen demonstriert. Während diese Studien das Potential von PQCs in standardisierten Benchmarking-Aufgaben aus dem OpenAI Gym gezeigt haben, ist noch ungeklärt, inwieweit QRL-Techniken basierend auf PQCs erfolgreich bei komplexeren Problemen mit hochdimensionalen Zustandsräumen angewendet werden können.

Diese Arbeit präsentiert einen hybriden quantenmechanisch-klassischen Ansatz, der einen PQC mit klassischen neuronalen Netzwerken zur Merkmalskodierung und zum Postprocessing kombiniert. Das resultierende Modell wird im Rahmen von approximativem Q-Learning trainiert und evaluiert, wobei eine umfassende Hyperparametersuche durchgeführt wird. Zwei repräsentative Spielumgebungen, Atari Pong und Breakout, dienen dazu, die Leistung des hybriden Modells zu bewerten. Ein klassisches Modell, das ähnlichen architektonischen Einschränkungen wie das hybride Modell unterliegt, wird als Referenz konstruiert. Die Ergebnisse dieser Studie zeigen, dass das entwickelte hybride Modell die Pong-Umgebung erfolgreich lösen kann und bei Breakout vergleichbare Ergebnisse wie die klassische Referenz erzielt. Darüber hinaus zeigen die Ergebnisse wichtige Hyperparameter-Einstellungen und Designentscheidungen auf, die die Interaktion zwischen quantenmechanischen- und klassischen Komponenten beeinflussen. Diese Arbeit trägt zum Verständnis hybrider quantenmechanisch-klassischer Modelle bei und stellt einen ersten Schritt in Richtung ihrer Anwendung in realen RL-Szenarien dar.

Abstract

Quantum machine learning (QML) is a promising area of application for near-term quantum computing devices, with hybrid quantum-classical models based on parameterized quantum circuits (PQCs) as prominent candidate algorithms. Recent advances in quantum reinforcement learning (QRL) have demonstrated the potential of PQCs as an alternative to classical deep learning models. While these studies have shown the performance of PQCs on standard benchmarking tasks from the OpenAI Gym, it remains an interesting open question to what extent QRL techniques based on PQCs can be successfully applied to more complex problems exhibiting higher-dimensional state spaces.

This work presents a hybrid quantum-classical framework that combines a PQC with classical feature encoding and post-processing layers. The resulting model is trained and evaluated in an approximate Q-learning setting and a comprehensive hyperparameter search is performed. Two representative game environments, Atari Pong and Breakout, are selected to assess the performance of the hybrid model. A classical model, subjected to architectural restrictions similar to those present in the hybrid model, is constructed to serve as a reference. The results of this study demonstrate that the proposed hybrid model is capable of solving the Pong environment and achieves scores comparable to the classical reference in Breakout. Furthermore, the findings shed light on important hyperparameter settings and design choices that impact the interplay of quantum and classical components. This work contributes to the understanding of hybrid quantum-classical models and provides a first step towards their deployment in real-world RL scenarios.

Contents

CHAPTER 1

Introduction

1.1 Motivation

There is no doubt that machine learning and artificial intelligence (AI) have established their position as some of the most impactful and transformative technologies of the 21st century, particularly in the course of the last decade. According to the International Data Corporation (IDC), global spending on AI, including software, hardware, and services for AI-centric systems, was projected to reach $154 billion in 2023, representing an increase of 26.9% over the amount spent in 2022 [IDC23]. AI investment is forecasted to approach $200 billion globally by 2025, driven by substantial investments in physical, digital, and human capital to integrate and leverage new AI technologies [Sac23]. This growth is expected to continue, with AI software spending expected to reach almost $300 billion by the year 2027, indicating the rapid incorporation of AI into a wide range of products and services [Gar23].

As the significance of machine learning and AI in industry and society grows, it is imperative to consider the operational challenges associated with these technologies: Ever-increasing model complexities [LVL22] lead to tremendous computational resource demands that in turn fuel enormous energy requirements [SGM19, NDM+21, PGL+21], comparable to those of whole nations, as recent studies suggest [Vri23]. For this reason, it comes as no surprise that there is an increasing interest in exploring alternative computing architectures for machine learning purposes that exceed the limitations of the classical silicon-based *von Neumann architecture* [vN93].

In this pursuit, *quantum computing* stands out as a promising candidate. The field of quantum computing involves information processing with devices that operate based on the principles of quantum mechanics [NC10]. By harnessing quantum mechanical phenomena such as superposition and entanglement and using them as computational resources, quantum computing tries to gain computational advantages in certain problem

classes. Among the most significant theoretical contributions is *Shor's algorithm* [Sho95] providing an exponential speedup for the prime factorization of large integers, a foundational breakthrough with profound implications for cryptography. Similarly, *Grover's algorithm* [Gro96] demonstrates a quadratic speedup for unstructured search problems, demonstrating a clear quantum advantage over classical counterparts for specific tasks. The *Harrow-Hassidim-Lloyd* (HHL) algorithm [HHL09] represents a substantial theoretical contribution to quantum computing by offering an exponential speed-up for solving systems of linear equations with a logarithmic runtime in the number of variables. The algorithm highlights the potential for quantum computing in areas such as optimization, quantum chemistry, machine learning and beyond, where solving large systems of linear equations is paramount. These algorithms serve as cornerstones, showcasing the potential for quantum computing to address problems that are beyond the reach of classical approaches in certain contexts.

However, it is crucial to approach these successes with a balanced perspective: Many proposed quantum algorithms rely on fault-tolerant quantum computers, which will not be available in the near future. The current stage of quantum computing, often referred to as the *noisy intermediate-scale quantum* (NISQ) era [Pre18], is characterized by devices that, while increasingly capable, still face significant challenges such as high error rates, short qubit coherence times and poor scalability in the number of qubits. Current NISQ devices exhibit qubit numbers ranging from tens to hundreds, depending on the architecture. However, to mitigate errors, these devices must combine multiple physical qubits into fewer logical qubits using error-correction codes, effectively reducing the number of usable qubits.

Nevertheless, despite these early-stage hardware limitations, ongoing research has successfully identified algorithms that demonstrate significant potential on today's quantum devices [BCLK+22]. Among these, *variational quantum algorithms* (VQAs) [CAB+21] are particularly promising and versatile candidates, applicable to various problems. VQAs represent a class of quantum algorithms that optimize *parametrized quantum circuits* (PQCs), quantum circuits that contain tuneable parameters, in a hybrid quantum-classical manner. By design, these algorithms are more compatible with near-term devices as they utilize low-depth circuits, require relatively few qubits, and their iterative optimization process is compatible with error mitigation techniques [CBB+23]. Compared to error correction, these techniques are less resource-heavy as they only require multiple iterations of the same noisy circuit.

Quantum machine learning (QML) has established itself as a promising field of application for VQAs, suitable for current and near-term NISQ devices. PQCs have been shown to be a viable alternative to classical neural networks (NNs) in various supervised [MNK+18, FN18, SBS+20, SK19, PSCLGF+20, SSM21] and unsupervised [ZLW19, CMD+20, ZLW21] learning tasks. However, few studies have explored the application of PQCs in reinforcement learning (RL) [CYQ+20, LS20, JGM+21, SJD22], a subfield of machine learning. RL is known to be particularly challenging, as there exists no labeled data and the learning is inherently unstable, which often results in prolonged

training times and significant computational costs. This underscores the importance of further exploring the capabilities and boundaries of quantum reinforcement learning (QRL) within this complex learning paradigm.

1.2 Problem Statement

Previous work [JGM+21, SJD22] has demonstrated the potential of QRL methods to compete with classical deep RL methods in standard benchmarking tasks such as OpenAI Gym environments [BCP+16]. While these results are very promising for a wider applicability of QRL methods, it remains an interesting open question to what degree QRL techniques based on PQCs can successfully be applied to more complex environments exhibiting high-dimensional observation spaces. NISQ devices only possess a limited number of computational units (qubits) and the computational resources required to simulate a quantum computer scale exponentially with the number of qubits, rendering the theoretical study of larger quantum systems increasingly challenging. The feature space of these non-trivial problems such as the raw pixels of Atari game interfaces is therefore too large to be encoded into a PQC directly. One approach to circumvent this issue is to use established dimensionality reduction techniques in a hybrid quantum-classical computing framework that enables testing quantum agents on higher dimensional problem settings. This was attempted in previous work [LS21] but the learning capabilities of the proposed hybrid models in the Atari games Pong and Breakout were deemed insufficient.

The aim of this work is to develop and study a hybrid quantum-classical framework for reinforcement learning in high-dimensional state spaces. The thesis assesses the performance of PQC-based hybrid models in two well-studied Atari games, which serve as a midway point between benchmark environments and real-world scenarios. Therefore the present study addresses the following research questions:

- Can hybrid quantum-classical machine learning methods perform well in high-dimensional reinforcement learning tasks?
- What design choices have a non-trivial impact on the hybrid models performance?
- How does their performance compare to classical models?

1.3 Contributions

This research demonstrates the learning capabilities of hybrid quantum-classical machine learning models compatible with current NISQ requirements in high-dimensional reinforcement learning settings. In particular, the models presented can effectively solve the Atari Pong environment with a performance similar to a comparable classical reference. In the more complex environment of Atari Breakout, the model achieves significant results comparable to the classical counterpart. For the first time, a hybrid quantum-classical model is successfully employed in RL settings with large observation spaces. Furthermore,

this work sheds light on which hyperparameters and design choices significantly influence learning performance, offering valuable insights for future research.

1.4 Outline

This thesis is structured as follows: Chapter 2 provides an introduction to the basic concepts of quantum computing and reinforcement learning, setting the stage for further discussions on how quantum circuits can be used as machine learning models and fit into the overarching reinforcement learning framework. Chapter 3 builds on these concepts and introduces the reader to the field of quantum machine learning, focusing on parameterized quantum circuits, which lie at the heart of the hybrid model proposed in this work. Throughout this chapter, the intricacies of training quantum models are explained and the current state of the art in quantum reinforcement learning is presented, identifying existing research gaps. The methodological setup is outlined in Chapter 4, starting with a detailed explanation of the hybrid model's architecture and parameters as well as the reinforcement learning framework. This chapter also highlights the specifics of the Atari game environments, including details on pre-processing. In Chapter 5 the main results of the study are presented and discussed. Chapter 6 concludes with a summary of the findings, addressing the initial research questions and highlighting the constraints of this study.

CHAPTER 2

Preliminaries

To fully appreciate the application of quantum computing in the field of machine learning, particularly in reinforcement learning, it is important to first introduce some basic concepts of both disciplines. This chapter revolves around two main topics, quantum computing and reinforcement learning. The first part explains how information processing in the world of quantum mechanics works and sets the stage for the application of quantum computing in machine learning. The second half of the chapter focuses on reinforcement learning, a subset of machine learning that involves self-learning agents that learn through trial and error by interacting with an environment. Specifically, the Q-learning algorithm is explained, as it serves as the overarching RL framework in this thesis.

2.1 Quantum Computing

Quantum computing encompasses information processing with devices that operate based on the principles of quantum mechanics. Research in this field aims to exploit quantum mechanical phenomena such as superposition and entanglement and use them as computational resources to gain computational advantages in specific problem domains or to explore new approaches for solving classically hard or intractable problems. The following sections introduce the key concepts of quantum computing and their interplay in pursuit of the aforementioned objectives. A comprehensive introduction to the field of quantum information and quantum computation is provided by [NC10].

2.1.1 Quantum bits - encoding information into quantum states

The basic unit of quantum information and quantum computation is the *quantum bit*, or *qubit*, representing a quantum analogue to the classical binary bit. In practice, a qubit may be realized by any two-level quantum system. Mathematically, the state of a qubit is described by a normalized vector in a two-dimensional complex vector space, called a

Hilbert space. The two orthogonal states $|0\rangle = (1,0)^T$ and $|1\rangle = (0,1)^T$ form a basis of this space and are referred to as the *computational basis* states. A fundamental principle of quantum mechanics states that a qubit is not limited to its basis states but can exist in any linear combination or *superposition* of the form

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle\,, \tag{2.1}$$

where $a, b \in \mathbb{C}$ are complex numbers known as *probability amplitudes* satisfying the normalization condition $|a|^2 + |b|^2 = 1$. However, there is a redundancy in this formulation as only the relative phase between these coefficients carries any physical meaning. It is thus sufficient to set $a = \cos\left(\frac{\theta}{2}\right)$ and $b = e^{i\phi}\sin\left(\frac{\theta}{2}\right)$ to fully describe the qubit's state.

This choice leads to a geometric representation of a qubit given by the *Bloch sphere*, where every possible state of the qubit corresponds to a point on the sphere's surface. The latter condition imposes an additional constraint on the state of the qubit that allows a general qubit state $|\psi\rangle$ to be parameterized by only two real values θ and ϕ. These are the polar and azimuthal angles in the Bloch sphere representation of a qubit state as depicted in Figure 2.1.

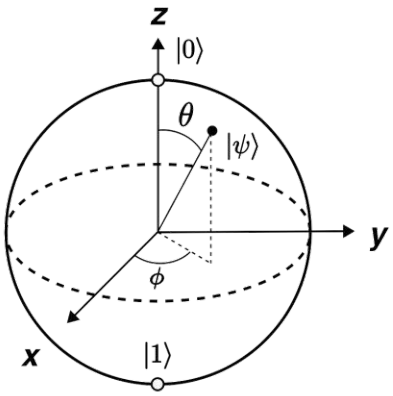


Figure 2.1: The Bloch sphere, a representation of the quantum state of a qubit. The computational basis states are indicated as circles at the north and south poles of the sphere. An arbitrary state $|\psi\rangle$ defined by the angles θ and ϕ is depicted as a black dot on the surface of the sphere.

Just as classical computers do not operate solely on a single bit, a quantum computer typically performs operations on multiple qubits. The state of a general n-qubit system is described by a superposition of all 2^n possible combinations of its qubits' basis states

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} c_i\,|i\rangle\,, \tag{2.2}$$

where $|i\rangle$ denotes the quantum state corresponding to the n-bit binary representation of the integer i. Each $|i\rangle$ is a tensor product of n qubit basis states ($|0\rangle$ or $|1\rangle$), encoding one of the 2^n possible combinations of basis states $\{|0...0\rangle, ..., |1...1\rangle\}$, where $|jk...n\rangle$ abbreviates $|j\rangle \otimes |k\rangle \otimes ... \otimes |n\rangle$. The coefficients c_i are again complex numbers that satisfy the normalization condition $\sum_{i=0}^{2^n-1} |c_i|^2 = 1$, which ensures that the total probability of finding the system in any of its possible states is one. This representation indicates the exponential growth of the state space dimension with an increasing number of qubits n. The combined Hilbert space for an n-qubit system is:

$$\mathcal{H}^{\otimes n} = \bigotimes_{i=1}^{n} \mathcal{H}_i. \tag{2.3}$$

If the state of an n-qubit system can be written as $|\Psi\rangle = |\psi_1\rangle \otimes ... \otimes |\psi_n\rangle$ for $|\psi_i\rangle \in \mathcal{H}_i$, it is called *separable*. Otherwise, the state is said to be *entangled*.

2.1.2 Quantum gates - manipulating quantum information

To perform calculations, it is essential to manipulate the information encoded in the qubits, which is achieved through the application of quantum gates. Analogous to classical logic gates, quantum gates alter the states of one or multiple qubits. Depending on the specific physical realization of the quantum computer, there are different methods to implement quantum gates. Mathematically, quantum gates operating on n qubits are represented by $2^n \times 2^n$-dimensional unitary matrices (that is, $\mathbf{U}^\dagger\mathbf{U} = \mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$), as they are norm-preserving and reversible. The first property ensures that the total probability of measuring the quantum system in any of its possible states remains equal to one, even after the application of a quantum gate. The reversible nature of quantum gates reflects the principle of information conservation in closed quantum systems as postulated by quantum mechanics.

An important set of single-qubit gates are the *Pauli matrices* $\boldsymbol{\sigma}_x$, $\boldsymbol{\sigma}_y$ and $\boldsymbol{\sigma}_z$. As quantum gates and in circuit diagrams, they are denoted as $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$:

$$\mathbf{X} = \boldsymbol{\sigma}_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{Y} = \boldsymbol{\sigma}_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \mathbf{Z} = \boldsymbol{\sigma}_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.4}$$

Each of these gates rotates or *flips* the qubit state vector by π radians around the x, y or z axes of the Bloch sphere. The $\mathbf{X}$ gate is often referred to as **NOT** gate, as it flips the state of a qubit between the two computational basis states ($\mathbf{X}|0\rangle = |1\rangle$ and vice versa). Using the Pauli matrices, it is possible to further define three useful gates, the *Pauli rotations*. While the Pauli matrices and many other quantum gates perform a fixed operation on the qubit(s) they act on, the Pauli rotations are *parameterized gates*, as the corresponding transformations depend on a parameter θ:

$$\mathbf{R}_x(\theta) = e^{-i\frac{\theta}{2}\boldsymbol{\sigma}_x} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix} \tag{2.5}$$

$$\mathbf{R}_y(\theta) = e^{-i\frac{\theta}{2}\boldsymbol{\sigma}_y} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix} \tag{2.6}$$

$$\mathbf{R}_z(\theta) = e^{-i\frac{\theta}{2}\boldsymbol{\sigma}_z} = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \tag{2.7}$$

So far, the gates introduced all act on individual qubits. An important 2-qubit gate is the **CZ** or **Controlled-Z** gate. It is an example of a *controlled gate* that only affects the second qubit, or *target qubit*, if the first qubit, the *control qubit*, is in the $|1\rangle$ state. For example, the **CZ** gate applied to $|01\rangle$ leaves the state in $|01\rangle$ but **CZ** applied to $|11\rangle$ changes the state of the system to $-|11\rangle$. This behaviour is instrumental for introducing entanglement between multiple qubits.

2.1.3 Quantum measurements - retrieving classical information

Unlike a classical bit whose state can be determined directly, the state of a qubit is only accessible through a *quantum measurement* conducted using a suitable *observable*. An observable is a physical quantity and it is represented by a Hermitian or self-adjoint (i.e., $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{M}}^\dagger$) operator $\boldsymbol{\mathcal{M}}$ acting on the quantum state of the system. This property ensures that all eigenvalues of the operator are real numbers and thus physically meaningful, which is critical for interpreting measurement outcomes. The measurement postulate of quantum mechanics states that when a quantum system is measured, the outcome of the measurement corresponds to one of the eigenvalues of the measured observable $\boldsymbol{\mathcal{M}}$, and the state of the system after the measurement is the eigenstate $|m\rangle$ corresponding to the observed eigenvalue m.

An important example of a quantum measurement is the *computational basis measurement* or **Z**-measurement, which uses the Pauli **Z**-matrix $\boldsymbol{\sigma}_z$ as an observable. The computational basis corresponds to the eigenstates of the Pauli **Z**-operator, where $|0\rangle$ is the eigenstate with eigenvalue $+1$ and $|1\rangle$ is the eigenstate with eigenvalue -1. Computational basis measurements are a specific case of *projective measurements* with the projection operators $\mathbf{P}_0 = |0\rangle\langle 0|$ and $\mathbf{P}_1 = |1\rangle\langle 1|$. An important postulate of quantum mechanics is the *Born rule*: Measuring a qubit in the general state 2.1 using a computational basis measurement, the probability of observing the qubit in state $|0\rangle$ is given by $p(0) = \langle\psi|\mathbf{P}_0^\dagger\mathbf{P}_0|\psi\rangle = |a|^2$, and the state of the qubit after measurement becomes $|0\rangle$. Similarly, the probability of observing the qubit in state $|1\rangle$ is $|b|^2$ and the post-measurement state becomes $|1\rangle$. It is said that the measurement has *collapsed* the state.

From this discussion, it becomes apparent that a quantum measurement is inherently a probabilistic process, where a single measurement provides limited insight into the pre-measurement quantum state. Given that the quantum state collapses to an eigenstate

of the measurement observable upon observation, repeated measurements of a single system instance do not yield additional information about the original superposition state. To accurately characterize the quantum state, one must perform repeated measurements across many identically prepared systems. For projective measurements, an explicit formula for the expectation value of the measurement exists:

$$E(\boldsymbol{\mathcal{M}}) = \sum_m mp(m) = \sum_m m \langle\psi|\mathbf{P}_m|\psi\rangle = \langle\psi|(\sum_m m\mathbf{P}_m)|\psi\rangle = \langle\psi|\boldsymbol{\mathcal{M}}|\psi\rangle =: \langle\boldsymbol{\mathcal{M}}\rangle\,. \quad (2.8)$$

Given the spectrum of the Pauli **Z**-matrix, the expectation value of a computational basis measurement falls within the interval $[-1, 1]$. This naturally generalizes to n-qubit systems, where $n > 1$: Separate (*local*) computational basis measurements performed on individual qubits each return a value between -1 and 1 as the expectation value. In this work, occurrences of terms like measurement result or outcome always refer to the expectation value rather than a single measurement.

2.1.4 Quantum circuits - a model for quantum computation

There exists a simple but expressive representation for quantum computations applied to one or multiple qubits: The *quantum circuit* diagram. It consists of several elements illustrating the flow of operations. Figure 2.1 depicts an arbitrary quantum circuit consisting of three qubits with instances of the gates discussed in Section 2.1.2. On the left side of the circuit, the initial state of the involved qubits is highlighted. Starting at the initial state a *quantum wire* indicates the flow of operations (loosely connected to the passage of time). If there appear no gates on a wire, the qubit is assumed to stay in its initial state until a gate is applied or until measurement, indicated by the meter-like symbol at the right end of the wire. Quantum gates that appear on a wire indicate their application to the respective qubit, where the order of application is from left to right. Controlled gates are depicted by a vertical line connecting the control qubit (black dot) with the target qubit (operation). Parametrized gates are represented with their according parameters.

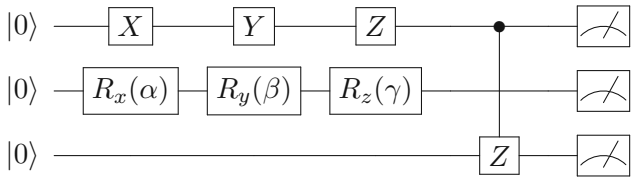


Table 2.1: Quantum circuit diagram containing various gates.

The concepts covered in the previous sections represent a theoretical minimum required to explain variational quantum algorithms based on parametrized quantum circuits. This is the subject of Chapter 3, where the topic of quantum machine learning is introduced.

The following sections discuss the field of classical reinforcement learning, one of the three main areas of modern machine learning.

2.2 Reinforcement Learning

At the heart of Reinforcement Learning lies the concept of a learning agent that interacts with an environment by taking actions based on observed states in order to accomplish a specific goal. No prior information about which actions to take is given to the agent, leaving it to learn through trial and error. Feedback is provided to the agent in the form of numerical rewards, but given that an action might have a long-term effect on the environment's trajectory, this complicates the task of assigning rewards to specific actions. Unlike supervised learning, RL does not rely on pre-labelled data; instead, the agent generates training data through its interactions with the environment. A unique challenge in reinforcement learning tasks lies in finding the right balance between exploring the environment for data collection and exploiting learned strategies. The formalization of RL concepts and the introduction of specific algorithms are the subject of the following sections. An in-depth introduction to the field of reinforcement learning can be found in the seminal work [SB18].

2.2.1 Markov Decision Process - the agent-environment interaction

An abstract framing for a RL problem is given by a Markov decision process (MDP). An MDP is comprised of a set of *states* $\mathcal{S}$, called the state space, which represents the possible observations of the environment and a set of *actions* $\mathcal{A}$, the action space, the agent can choose from. For a discrete sequence of time steps, $t = 0, 1, 2, 3, ...$, the agent chooses a specific action $a_t \in \mathcal{A}$ when in state $s_t \in \mathcal{S}$, which brings it to a new state s_{t+1} and it receives a numerical *reward* $r_{t+1} \in \mathcal{R}$ upon this transition. The probability of going from a state s to its successor state s' and receiving a reward r after choosing action a is given by the probability distribution $p(s', r|s, a)$, which characterizes the dynamics of the MDP. Notably, the probability of transitioning to a new state s' solely depends on the predecessor state s and the chosen action a, but not on earlier states and actions. This limited memory of the environment is known as the *Markov property*. The components of the MDP, i.e., the states, actions, transition probabilities and rewards enable it to describe a wide variety of RL tasks and give rise to further concepts necessary for defining RL algorithms.

2.2.2 The Q-value function - quality of states and actions

A crucial ingredient for solving RL tasks is to define the goal the agent is intended to achieve. With the notion of rewards,it is possible to define the *return* G_t at time step t as the sum of discounted future rewards:

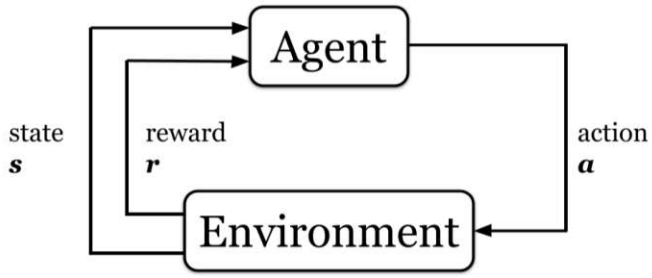


Figure 2.2: A simplistic illustration of the reinforcement learning problem: An agent observes states s and takes actions a in an environment for which it obtains rewards r from the environment.

$$G_t := \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{2.9}$$

where $0 < \gamma < 1$ is the *discount rate* which determines the importance of long-term over short-term rewards and keeps the - possibly infinite (if $T = \infty$) - sum from diverging. Since MDPs are inherently probabilistic, the goal of the agent is to maximize the expected value of this return by selecting proper actions. The agent selects actions based on a *policy function* $\pi(a|s)$ that defines the probability of taking action a when in a state s. With these ingredients, it is possible to define an *action-value function* $Q_\pi(s, a)$ that indicates the value or *quality* of a state-action pair (s, a). In other words: The *Q-function* is the expected return when taking action a while being in state s and following the policy π thereafter,

$$Q_\pi(s, a) := \mathbb{E}_\pi[G_t | s_t = s, a_t = a]. \tag{2.10}$$

The maximum expected return achievable when taking action a while in state s and following an *optimal policy* thereafter (indicated by $\max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$ in the recursive definition below), is given by the *optimal Q-value function*:

$$Q^*(s, a) := \max_\pi Q_\pi(s, a) = \mathbb{E}[r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]. \tag{2.11}$$

Equation 2.11 is known as the *Bellman optimality equation*. If the optimal Q-function $Q_*(s, a)$ is known, deriving an optimal policy π_* is straight-forward: In state s the agent should choose an action that maximizes the optimal Q-function, i.e

$$\pi^*(s) = \arg\max_a Q^*(s, a). \tag{2.12}$$

This observation leads to the main objective of *Q-learning*, which is finding the optimal Q-function in order to deduce the optimal policy for the agent.

It is worth mentioning that there exists another class of methods known as policy-based methods. Unlike Q-learning, which is a value-based approach, policy-based methods directly parameterize the policy function, optimizing it with respect to the expected return by gradient ascent. These methods work by adjusting the policy parameters in a way that increases the probability of actions that lead to higher rewards.

2.2.3 Q-learning - finding the optimal Q-function

Q-learning is an iterative RL algorithm that learns the optimal policy by estimating the optimal Q-values through interactions with the environment. It does not require a model of the environment (i.e., transition probabilities and rewards of the MDP are unknown) and is thus considered a *model-free* method. Q-learning updates the Q-values according to the following update rule,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big], \tag{2.13}$$

where α is the learning rate and the initial Q-values are set to zero. Note that both $Q(s_{t+1}, a_{t+1})$ and $Q(s_t, a_t)$ are approximations and thus Q-learning relies on a concept known as *bootstrapping*, where the estimates are updated using other estimates. The term within brackets in Equation (2.13) is the *temporal difference* (TD) error, reflecting the discrepancy between the current Q-value estimate and a new, more accurate estimate. This new estimate sums the observed reward r_{t+1} with the estimated maximum discounted future value, $\gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$, for a better approximation of the state-action value $Q(s_t, a_t)$. This process allows Q-learning to iteratively converge towards the optimal Q-values [SB18]. In theory, the agent could follow a purely random exploration policy while still being able to eventually estimate the optimal Q-values, thus Q-learning is an example of an *off-policy* method. However, convergence to an optimal policy may take a long time if following a purely random exploration policy. The ε-greedy policy is a simple yet effective strategy for balancing exploration and exploitation during the training process. With an adjustable probability ε, the agent selects a random action, facilitating exploration of the state space. Conversely, with a probability $1 - \varepsilon$, the agent selects the action with the highest estimated Q-value for the current state, exploiting its current policy. This strategy ensures that the agent does not prematurely converge to a suboptimal policy by getting stuck in local maxima and continues to explore the environment to find the optimal policy.

2.2.4 Approximate Q-Learning

In its simplest form, Q-learning is designed as a tabular learning approach where the Q-value for each state and action is stored in a *Q-table*. This approach ultimately limits the applicability of Q-learning to rather low-dimensional state and action spaces. A remedy for this restriction is *approximate Q-learning*. Here, instead of a Q-table, a Q-function approximator is used. This is a function $Q_{\boldsymbol{\theta}}(s, a)$ dependent on a set of

parameters $\boldsymbol{\theta}$ that approximates the Q-values of any state-action pair. Instead of working on all state-action pairs, this method only requires adjusting the parameters $\boldsymbol{\theta}$ in the parameter space, which usually is of far lower dimension. The introduction of deep neural networks as Q-function approximators led to a series of advancements in the field of reinforcement learning and coined the term *deep Q-learning* (DQL), a form of approximate Q-learning using so-called *deep Q-networks* (DQN) [MKS+13, MKS+15]. Here, instead of directly updating the Q-value approximation for a given state-action pair, a loss function based on the TD error is constructed as follows

$$\mathcal{L}(\boldsymbol{\theta}) = \left(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{\theta}^-) - Q(s_t, a_t; \boldsymbol{\theta})\right)^2, \tag{2.14}$$

where the dependence of the approximate Q-function on the parameters $\boldsymbol{\theta}$ is highlighted. This loss function is then minimized through gradient descent on the DQN weights to refine the approximation of the Q-values:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{2.15}$$

The $\boldsymbol{\theta}^-$ in the estimated Q-function of timestep $t+1$ indicates the use of a *target network*: Instead of using the same Q-network for making predictions and setting the target, a second Q-network is used as *target model* $\hat{Q}$, which is a periodically updated copy of the *online model* Q, that learns at each step. This technique avoids undesirable feedback loops and correlations between the target and the estimated Q-values, stabilizing the training process. Another critical concept to stabilize learning is *experience replay*: By storing the agent's experiences at each time step, denoted as transitions $e_t = (s_t, a_t, r_t, s_{t+1})$, in a replay memory $D_t = \{e_1, ..., e_n\}$, the algorithm can randomly sample a mini-batch of transitions to train the network. This approach breaks the temporal correlations between consecutive learning samples and smoothens the learning process by using a diverse set of experiences. Furthermore, experience replay allows for more efficient use of previous experiences by learning from them multiple times, which is especially beneficial in environments where obtaining new samples is costly or limited.

Algorithm 2.1 shows a pseudo-code implementation of deep Q-learning with experience replay and a target network: A replay memory D is initialized to store N sequences (s_t, a_t, r_t, s_{t+1}). The online network Q is initialized with random weights $\boldsymbol{\theta}$ and the target network $\hat{Q}$ is assigned the same weights $\boldsymbol{\theta}^- = \boldsymbol{\theta}$. The algorithm proceeds through a series of episodes. Each episode starts with an initial state s_1. Within each episode, the agent interacts with the environment for a number of time steps T. At each step t, the agent chooses an action a_t: Either it randomly selects an action with probability ε (exploration) or chooses the action that maximizes the current Q-value estimation for the current state s_t (exploitation). The chosen action is executed in the environment, and the agent observes the resulting reward r_t and the next state s_{t+1}. This transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay memory D. The agent then samples a random minibatch of size $|B|$ of transitions from the replay memory. For each transition in the minibatch: If the

next state s_{j+1} is a terminal state, the target value y_j is set to the observed reward r_j. Otherwise, y_j is set to r_j plus the discounted maximum future reward as estimated by the target network $\hat{Q}$ for the next state s_{j+1}. The loss is computed as the mean squared error between the predicted Q-values from the network $Q(s_j, a_j; \boldsymbol{\theta})$ and the target values y_j. A gradient descent step is performed to update the network weights $\boldsymbol{\theta}$ to minimize this loss. Every C steps, the weights of the target network $\hat{Q}$ are updated to match the weights of the online network Q.

Algorithm 2.1: Deep Q-learning with experience replay and target network

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights $\boldsymbol{\theta}$
Initialize target action-value function $\hat{Q}$ with weights $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
for *episode* $= 1$ **to** M **do**
  Initialize sequence s_1
  for $t = 1$ **to** T **do**
    With probability ε select random action a_t otherwise select $a_t = \text{argmax}_a Q(s_t, a; \boldsymbol{\theta})$
    Execute action a_t in emulator, observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random minibatch of size $|B|$ of transitions (s_j, a_j, r_j, s_{j+1}) from D where j is from the index set B
    for $j \in B$ **do**
      Set $y_j = \begin{cases} r_j, & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \boldsymbol{\theta}^-), & \text{for non-terminal } s_{j+1} \end{cases}$
    end
    Compute the loss for the minibatch: $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|B|} \sum_{j \in B} (y_j - Q(s_j, a_j; \boldsymbol{\theta}))^2$
    Perform a gradient descent step: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
    Every C steps, reset $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
  end
end

State of the art in deep Q-learning

Deep Q-learning has seen remarkable successes, for example by achieving superhuman performance in Atari games, as demonstrated in the seminal works [MKS+13] and the follow-up [MKS+15]. In [vHGS15] the DQN algorithm was further refined by introducing Double Deep Q-learning (DDQN), which reduces overestimation by decoupling action selection from value evaluation, improving the accuracy of value estimation. Improvements in the experience sampling method, as proposed in [SQA+16], helped accelerate the learning process by replaying important experiences more frequently. Another improvement on the original DQN algorithm was proposed in [WSH+16] with Dueling Network

Architectures, which separately estimate state values and action advantages, leading to more precise value function estimations.

CHAPTER 3

Quantum Reinforcement Learning

Similar to classical machine learning and reinforcement learning, quantum reinforcement learning (QRL) can be considered a subset of quantum machine learning (QML), an active area of research focused on leveraging quantum computing capabilities to improve existing machine learning methods or develop novel ones. QML is one of the most promising fields of application of quantum computing. Particularly on today's error-prone and limited-qubit hardware, variational quantum algorithms (VQAs) emerged as a promising candidate for near-term quantum devices. VQAs utilize parameterized quantum circuits (PQCs) as main components, optimizing them in a hybrid quantum-classical approach. PQCs are quantum circuits that incorporate parameterized gates adjustable by classical computing resources. This section initially provides an overview of the field of QML, including a concise historical background. The main focus, however, is on PQCs and how they are used in a machine learning context. The encoding of classical data within PQCs is explained and how it affects the function classes representable by PQCs. Further, the conjunction of PQCs with classical layers to form flexible hybrid quantum-classical models is discussed. Lastly, this section shows how hybrid quantum-classical models based on PQCs can be applied in an approximate Q-learning setting, highlights recent advancements in QRL research to illustrate the current state-of-the-art and sets the stage for the research conducted in this study.

3.1 Quantum Machine Learning

Given the drastic advances made in classical machine learning over the last decade, it is no surprise that quantum computing research is increasingly focusing on this domain with the prospect of further enhancing existing ideas, exploring potential synergies between the fields, or uncovering novel concepts altogether. Before delving into the theoretical aspects of QML, it is crucial to give a high-level description of what the domain of QML encompasses. A categorization found in [SP21] identifies four flavors of quantum

machine learning depending on what the quantum element is - the data, the algorithm or both: QML can involve classical data processed by classical algorithms (CC) that draw inspiration from quantum mechanics, for example *tensor networks* [SS16, GPC19]. The quantum data - classical algorithm (QC) domain seeks to leverage classical machine learning techniques to support progress in quantum information processing itself. The classical data - quantum algorithm (CQ) paradigm includes the framework proposed in this thesis. In this scenario, problems defined by classical data are tackled using a quantum or hybrid quantum-classical algorithm. A particular challenge in this setting is effectively encoding data into a quantum state, an issue that is addressed later in this chapter in Section 3.1.3. In the quantum data - quantum algorithm (QQ) realm, the data itself is inherently quantum, potentially generated by another quantum subroutine.

3.1.1 The history of quantum machine learning

Pinpointing the exact start of QML research is challenging, as self-learning machines have fascinated people since the dawn of computing, and quantum computing is no exception. With the introduction of the famous quantum algorithms of Shor [Sho95] and Grover [Gro96] the first tangible ideas of how quantum computing might be used for self-learning algorithms arose. In the 1990s and early 2000s, QML research - if it can be considered as such - was primarily concerned with artificial neural networks, albeit with limited practical outcomes. Early works investigated possible connections to neural computing, drawing inspiration from biological information processing [Kak95] and conceptualized Quantum Neural Networks (QNNs), with the aim of harnessing the unique computational capabilities of quantum information processing [BNS+00, Alt01, GZ02, AA02]. The introduction of the Harrow-Hassidim-Lloyd (HHL) algorithm for matrix inversion [HHL09] marked a shift in the trajectory of quantum machine learning. Promising an exponential speed-up with a logarithmic runtime in the problem dimension, at least under specific assumptions, the HHL algorithm directed research into enhancing linear algebra subroutines common in many machine learning algorithms. The authors of [WBL12] built upon the HHL algorithm for solving linear regression problems and evaluating least-squares fits. In [RML14], the authors express a support vector machine as an approximate least-squares problem to allow the use of the HHL algorithm. Results reported in [ZFF19] promise an exponential speed-up in compute time in certain cases by applying the HHL algorithm to Gaussian process regression. Noticeably, the HHL algorithm led to a significant expansion of approaches to incorporating quantum computing in the field of machine learning. Nevertheless, it faces considerable constraints. Strict conditions on the occurring matrices to be sparse and have a low condition number limit the applicability to special cases. More critically, the effective implementation of HHL relies on fault-tolerant quantum computers, which remain beyond current technological capabilities. In contrast, the study of *parameterized quantum circuits* (PQCs) as machine learning models [BLS+19] in the framework of variational quantum algorithms (VQAs) [CAB+21] constitutes a different point of view: Instead of focusing on speeding up existing (sub)routines, the research aims to explore new families of models and their potential advantages over classical approaches in solving specific problems. While PQCs on current

NISQ hardware do not guarantee a *quantum advantage*, that is, the ability to outperform classical algorithms for the same tasks, they offer significant advantages over traditional quantum algorithms: PQCs are compatible with today's NISQ devices and even shallow circuits consisting of only a single qubit can successfully solve non-linear learning tasks [PSCLGF+20, JFPN+23]. VQAs using PQCs are an example of hybrid quantum-classical algorithms as part of the computation is performed on classical hardware, while the quantum part serves as the hypothesis or model. The following sections introduce the reader to the concept of PQCs and their application as machine learning models.

3.1.2 Parametrized quantum circuits as machine learning models

In essence, a parametrized quantum circuit (PQC), also *variational quantum circuit*, is a quantum circuit that applies parameterized gates to its qubits. The values of these parameters are controlled by classical hardware, usually to minimize a suitable cost function with respect to an expected value of a measurement observable. An example of such parameterized gates are the Pauli rotations discussed in Section 2.1.2, where the parameters correspond to the angles by which the qubit states are rotated around a specific axis. More generally, given an initial state of a n-qubit quantum system, for example the trivial state $|0\rangle^{\otimes n}$, a PQC applies the parameter-dependent unitary transformation $U(\boldsymbol{x}, \boldsymbol{\theta})$ to its qubits, which is defined as:

$$U(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{l=1}^{L} V_l(\boldsymbol{\theta}) U_l(\boldsymbol{x}). \tag{3.1}$$

In the context of machine learning, the $U_l(\boldsymbol{x})$ are the *encoding unitaries*, responsible for encoding or *embedding* the features x_i of a feature vector $\boldsymbol{x} \in \mathbb{R}^d$ into the quantum state. The $V_l(\boldsymbol{\theta})$ are the *variational* or *parameterized unitaries*, composed of gates of adjustable parameters $\boldsymbol{\theta} \in \mathbb{R}^k$ that are optimized during the training process. The different $U_l(\boldsymbol{x})$ and $V_l(\boldsymbol{\theta})$ can all be distinct unitaries and may each encode only parts of the feature vector $\boldsymbol{x}$ and the parameter vector $\boldsymbol{\theta}$ respectively. The alternating application of encoding and variational gates is repeated L times in *layers*. This *data re-uploading* [PSCLGF+20] architecture is known to enhance the expressivity of PQC-based machine learning models, as will be discussed in Section 3.1.4.

The state of the system resulting from the application of the circuit is $U(\boldsymbol{x}, \boldsymbol{\theta}) |0\rangle^{\otimes n} = |\psi(\boldsymbol{x}, \boldsymbol{\theta})\rangle$. According to the measurement postulate and the Born rule (see Section 2.1.3) the expectation value of some measurement observable $\boldsymbol{\mathcal{M}}$ on the quantum system prepared by the PQC is

$$\langle \boldsymbol{\mathcal{M}} \rangle_{\boldsymbol{x}, \boldsymbol{\theta}} = \langle \psi(\boldsymbol{x}, \boldsymbol{\theta}) | \boldsymbol{\mathcal{M}} | \psi(\boldsymbol{x}, \boldsymbol{\theta}) \rangle =: f_{\boldsymbol{\theta}}(\boldsymbol{x}). \tag{3.2}$$

This defines a deterministic quantum machine learning model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, that, given input $\boldsymbol{x}$ and setting parameters $\boldsymbol{\theta}$, outputs a deterministic value $\langle \boldsymbol{\mathcal{M}} \rangle_{\boldsymbol{x}, \boldsymbol{\theta}}$.

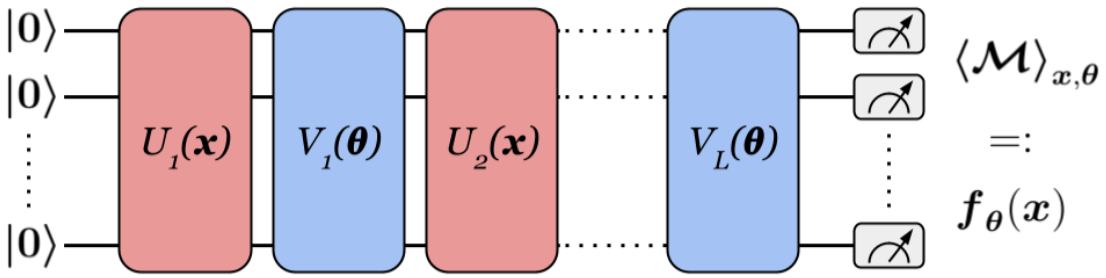


Figure 3.1: A parameterized quantum circuit as defined in Equation 3.1 as machine learning model. A feature vector $\boldsymbol{x}$ is encoded into the quantum system in its trivial state $|0\rangle^{\otimes n}$ via the repeated encoding unitaries $U_l(\boldsymbol{x})$ (red). Intermediate variational unitaries $V_l(\boldsymbol{\theta})$ (blue) enable the training of the circuit. The output of the model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the expectation value $\langle \boldsymbol{\mathcal{M}} \rangle_{\boldsymbol{x},\boldsymbol{\theta}}$ of a (or multiple) observables (e.g., a Pauli-**Z** observable on each qubit) measured at the end of the circuit.

Training the quantum model is conducted in a hybrid quantum-classical manner: On a classical computer, the output produced by the quantum circuit (i.e., Equation 3.2) is compared to a target value through a suitable loss (or cost) function $\mathcal{L}(\boldsymbol{\theta})$ that quantifies the error as a function of the model's trainable parameters $\boldsymbol{\theta}$. The classical computer then optimizes the model parameters towards a lower cost. A common choice for the optimization algorithm are gradient-based methods such as *gradient descent*. These methods minimize the cost by updating the model parameters in the direction of the negative gradient of the cost function with respect to the parameters. This is repeated iteratively until a desired cost is achieved. More details on obtaining gradients of quantum computations are discussed in Section 3.1.5.

3.1.3 Encoding classical data in PQCs

In order to operate on a classical dataset, it is essential to represent the data on the quantum computer. There exist multiple methods to realize the encoding of classical data into a n-qubit quantum system, the most common being *basis encoding*, *amplitude encoding* and *rotation encoding*. Basis and amplitude encoding are well suited for single-layer PQCs, where the data is encoded only once (i.e., $L = 1$ or equivalently $U_l(\boldsymbol{x}) = \boldsymbol{I}$ for $l > 1$ in Equation 3.1). Rotation encoding enables repeated uploading of data throughout the quantum circuit, giving rise to data re-uploading models.

The following notation is adopted throughout the remainder of this chapter: Let $\mathcal{X}$ be the space of (raw) data and $\tilde{\boldsymbol{x}} \in \mathcal{X}$ a *data sample*. Depending on the problem, this could be vectors in $\mathbb{R}^d$, images represented as tensors in $\mathbb{R}^{h \times w \times c}$, or more complex data structures. Further, let $\mathcal{F}$ be the feature space and $\boldsymbol{x} \in \mathcal{F}$ a *feature vector*. Each data sample $\tilde{\boldsymbol{x}}$ is transformed into a feature vector $\boldsymbol{x} = \phi(\tilde{\boldsymbol{x}})$ via a *pre-processing function* $\phi : \mathcal{X} \to \mathcal{F}$. This function ϕ could range from simple normalization over feature extraction methods to sophisticated feature engineering processes, such as convolutional neural networks.

Basis encoding

In basis encoding, a classical data vector $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$ transformed to binary representation $\boldsymbol{x}_{bin} = \phi(\tilde{\boldsymbol{x}})$ is directly encoded into the basis state of the quantum system by flipping the according qubits, resulting in $|\psi(\boldsymbol{x}_{bin})\rangle$, for example: $\tilde{\boldsymbol{x}} = (0.5, -0.1, 0.3) \rightarrow \boldsymbol{x}_{bin} = (01000, 10001, 00100) \rightarrow |\psi(\boldsymbol{x}_{bin})\rangle = |010001000100100\rangle$, where the first bit is used to indicate the sign. As can be observed, the number of qubits required for this encoding technique is rather high, as it scales linearly with the number of features in the input vector as well as logarithmically with the inverse of the desired precision.

Amplitude encoding

Another option is to encode the features onto the amplitudes of the quantum state. Amplitude encoding does not require a binary representation, but normalization of the data vector and padding $\boldsymbol{x}_{pad} = \phi(\tilde{\boldsymbol{x}})$ to expand it to length 2^n, where n is the number of qubits. Using again the example from before: $\tilde{\boldsymbol{x}} = (0.5, -0.1, 0.3) \rightarrow \mathbf{x}_{pad} = (0.845, -0.169, 0.507, 0.000) \rightarrow |\psi(\hat{\mathbf{x}}_{pad})\rangle = 0.845\,|00\rangle - 0.169\,|01\rangle + 0.507\,|10\rangle + 0\,|11\rangle$. Each value is associated with the amplitude of a different basis state, and therefore the number of qubits required is $\lceil \log_2(d) \rceil$ for $\boldsymbol{x} \in \mathbb{R}^d$.

Rotation encoding

The third encoding method is rotation encoding, also called *angle encoding* or in more general cases, *time-evolution encoding* [SP21]. To illustrate this encoding scheme, a single scalar value $x \in \mathbb{R}$ shall be encoded onto a qubit in its trivial state $|0\rangle$. The idea is to use x as the parameter of a Pauli rotation gate, i.e., the angle of rotation of the qubit state, which justifies the name of this encoding technique. This operation yields the state $|\psi(x)\rangle = R_j(x)\,|0\rangle$, where R_j is a Pauli rotation. A crucial observation when encoding features via qubit rotations is the ambiguity arising from the scale of the input features x_i and the periodicity of the effect of a rotation gate: Encoding a feature x_i via some Pauli rotation is equivalent to encoding $x_i + 2n\pi$ with $n \in \mathbb{Z}$. Therefore, pre-scaling the data to the interval of periodicity is required before encoding. The periodic nature of the rotation encoding has a direct relation to the functions a quantum machine learning model based on PQCs can represent, as will be discussed in the following section.

3.1.4 Expressivity of PQCs

Encoding data in a parameterized quantum circuit via rotation gates leads to a concise representation of quantum machine learning models like 3.2 that highlights the function classes these models can represent. In their work [SSM21], the authors offer a rigorous analysis of the expressive power of QML models based on rotation encoding and deduce a correspondence to trigonometric functions, specifically partial Fourier series. Theorem 1, adapted from [SP21] and restricted to Pauli rotation encoding, summarizes one of their key findings:

Theorem 1 (Function class of quantum models with Pauli rotation encoding) *Let $\tilde{\boldsymbol{x}}$ be a data sample in the data space $\mathcal{X}$, $\boldsymbol{x} = \phi(\tilde{\boldsymbol{x}})$ its corresponding feature vector in the feature space $\mathcal{F} = \mathbb{R}^N$ obtained by the feature encoding function $\phi : \mathcal{X} \to \mathcal{F}$. Further, let $\mathcal{Y} = \mathbb{R}$ be the output domain and $f_{\boldsymbol{\theta}}$: $\mathcal{F} \to \mathcal{Y}$ a deterministic quantum model as defined in 3.2 with circuit $U(\boldsymbol{x}, \boldsymbol{\theta})$ as in Equation 3.1. Accordingly, the j'th feature is encoded by a gate $e^{-ix_j\boldsymbol{\sigma}_j/2}$, where $\boldsymbol{\sigma}_j \in \{\boldsymbol{\sigma}_x, \boldsymbol{\sigma}_y, \boldsymbol{\sigma}_z\}$ is a Pauli matrix. Then, $f_{\boldsymbol{\theta}}$ can be written as*

$$f_{\boldsymbol{\theta}}(\phi(\tilde{\boldsymbol{x}})) = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{\omega_1 \in \Omega_1} \dots \sum_{\omega_N \in \Omega_N} c_{\omega_1 \dots \omega_N}(\boldsymbol{\theta}) e^{i\omega_1 x_1} \dots e^{i\omega_N x_N}, \tag{3.3}$$

where the frequency spectrum of the j'th feature j=1,...,N, is given by

$$\Omega_j = \{\lambda_s^j - \lambda_t^j | s, t \in \{1, 2\}\} = \{-1, 0, 1\}. \tag{3.4}$$

This frequency spectrum is the set of all values produced by differences between any two eigenvalues λ^j of $\boldsymbol{\sigma}_j/2$. We are guaranteed that $0 \in \Omega$, and that for each $\omega \in \Omega$ there is $-\omega \in \Omega$ too with $c_\omega(\boldsymbol{\theta}) = c^*_{-\omega(\boldsymbol{\theta})}$. This symmetry guarantees that $f_{\boldsymbol{\theta}}$ is real-valued, and that the sum can be rewritten with cosine and sine functions.

For the proof of Theorem 1 in its general formulation see [SSM21]. The frequencies of the trigonometric function accessible by the model defined in Equation 3.2, as a function of the features $\boldsymbol{x}$, are determined by the eigenvalues of the scaled Pauli matrices $\lambda_1^j = 0.5$ and $\lambda_2^j = -0.5$. For Pauli rotations, all differences $\lambda_s^j - \lambda_t^j$ are integer-valued and Eq. 3.3 represents a multi-dimensional partial Fourier series with limited frequency spectra $\Omega_j = \{-1, 0, 1\}$, assuming that each feature x_j is encoded only once in the circuit (or equivalently, each x_j is different). The coefficients $c_{\omega_1 \dots \omega_N}$ are defined by the non-encoding part of the quantum circuit, as shown in [SSM21]. Together, both the frequency spectra and the coefficients define the functions the quantum model can realize.

While the frequency spectrum of $f_{\boldsymbol{\theta}}$ as a function of a single feature x_j is rather limited, note that it is straightforward to choose a feature encoding ϕ that achieves a rich spectrum for the data components $\tilde{x}_i$. Consider, for example, two features x_i and x_j that encode the same data component $\tilde{x}_k$. Then, in Equation XY, the terms $e^{i\omega_i x_i} = e^{i\omega_i \tilde{x}_k}$ and $e^{i\omega_j x_j} = e^{i\omega_j \tilde{x}_k}$ combine to $e^{i(\omega_i + \omega_j)\tilde{x}_k}$ and the sums over $\omega_i \in \Omega_i$ and $\omega_j \in \Omega_j$ together form a sum over the richer spectrum $\tilde{\Omega} = \{(\lambda_s^i - \lambda_t^i) + (\lambda_s^j - \lambda_t^j) | s, t \in \{1, 2\}\} = \{-2, -1, 0, 1, 2\}$. Therefore, operationally, encoding a single data component $\tilde{x}_i$ in multiple rotations (either sequentially on the same qubit or in parallel over many qubits) has the effect of enriching the spectrum of f_θ as a function of that data component. In general, encoding a data component in d Pauli encodings allows to access the spectrum $\Omega = \{-d, ..., d\}$. The expressivity of the model with respect to the input data therefore is highly dependent on the effect of the pre-processing mapping ϕ.

The repeated uploading of classical information into the quantum circuit was first introduced in [PSCLGF+20], who coined the term *data re-uploading*, and was extensively

studied in [SSM21]: From their analysis, the authors derive that if a quantum model can generate a sufficiently diverse set of frequencies (e.g., through repeated encodings) and can manipulate the Fourier coefficients associated with these frequencies effectively, then such models are universal function approximators. The authors of [JGM+21] demonstrate that data re-uploading models ($L > 1$ in Equation 3.1), where each layer encodes all features x_j of the feature vector $\boldsymbol{x}$, can be mapped to single-layer models ($L = 1$ in Equation 3.1), where features are encoded only once. It follows, that data re-uploading is not a requirement for universal function approximation of PQCs. However, single-layer models have substantially higher requirements in the number of qubits compared to data re-uploading models. The authors of [PSCLGF+20] further demonstrate how trainable weights on the features enhance the expressivity of PQCs, which they use to conjecture universal function approximation of re-uploading PQCs later shown in [PSLNGS+21].

As stated above, an important consideration for the expressivity of a PQC is its general structure or *ansatz*. The circuit ansatz defines the circuit template, similar to the architecture of a neural network. Choosing the right ansatz can have a great impact on the expressibility of a quantum model, as numerically demonstrated in [SJAG19]. A particular versatile ansatz is the *hardware efficient ansatz* introduced in [KMT+17], which aims to utilize gates and qubit connectivities native to a given quantum device. In this work, single-qubit rotations and entangling gates are used that follow the topology of current NISQ devices, similar to those used in the quantum circuit sampling experiments conducted in [AAB+19].

3.1.5 Estimating gradients of quantum computations

As previously discussed in Section 3.1.2, training quantum machine learning (QML) models involves minimizing a cost or loss function $\mathcal{L}(\boldsymbol{\theta})$, which measures the discrepancy between computed expectation values $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ and target values. Like in classical machine learning, this optimization task can be tackled with gradient-based methods like gradient descent and its variations in a hybrid quantum-classical manner. Once the gradient of the cost function with respect to the variational gate parameters $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ is known, the classical computer can update the parameters of the PQC in the direction of smaller cost by performing a gradient descent step. In classical machine learning, gradients are obtained via *backpropagation*, a form of automatic differentiation (autodiff), which uses intermediate values to efficiently calculate the gradient of a function with a runtime similar to the execution of the function itself. This dependency on intermediate information makes it difficult to apply backpropagation, or autodiff in general, to quantum computations, as the collapse of the quantum state upon measurement prevents the straight-forward reuse of information [MBK21, AKH+23]. This fundamental difference requires alternative approaches for gradient computation in quantum machine learning, such as the *parameter-shift rule* [MNK+18, SBG+19, MBK21].

The parameter-shift rule exploits the trigonometric nature of PQCs as functions of their parameters. Theorem 1 equally applies when the quantum model is seen as a function of the variational parameters $\boldsymbol{\theta}$ instead of the inputs $\boldsymbol{x}$. As a function of a single parameter

θ_i with all other parameters $\theta_j \in \boldsymbol{\theta}, j \neq i$, and the inputs $\boldsymbol{x}$ fixed, the quantum model as defined in 3.2 is of the form

$$f_{\theta_i} = a \cdot \sin(\theta_i + b) + c, \tag{3.5}$$

where the constants a, b and c are determined by the fixed part of the circuit. Functions of this type adhere to the parameter-shift rule, which allows the exact computation of the derivative with respect to the parameter θ_i using two evaluations of the function with a shift in the parameter. Specifically, for a given parameter θ_i in a quantum circuit, the gradient of the expectation value $f_{\theta_i} := \langle \mathcal{M} \rangle_{\boldsymbol{\theta_i}}$ can be evaluated as:

$$\frac{\partial f_{\theta_i}}{\partial \theta_i} = \frac{f_{\theta_i + \pi/2} - f_{\theta_i - \pi/2}}{2} = a \cdot \cos(\theta_i + b). \tag{3.6}$$

This applies to an expectation value of the form 3.2 - the deterministic quantum machine learning model $f_{\boldsymbol{\theta}}$. Note that Equation 3.6 is not a finite difference approximation, but gives the exact value of the gradient - if the expectation value $\langle \mathcal{M} \rangle$ was known exactly. In that sense, an estimation of the exact gradient can be computed via Equation 3.6 using $2s$ circuit evaluations per parameter θ_i, where s is the number of shots to estimate the expectation.

When *simulating* quantum computations on classical computers, the situation is however different: The gradients of simulated quantum computations can be evaluated without the need for the parameter-shift rule. Since the quantum state of the system is completely known during all times of the computation, classical automatic differentiation can be applied directly.

One phenomenon that critically impacts trainability of PQCs, is a gradient vanishing exponentially in the circuit depth and number of qubits. These *barren plateaus* [MBS+18] in the cost function landscape are especially prominent in deep circuits, as in this setting an exponentially large space has to be searched for the solution. Much effort is put into mitigating the barren plateau problem through educated ansatz strategies and suitable parameter intialization [GWO+19, ZLH+22].

3.1.6 Hybrid quantum-classical models

Training PQCs involves quantum (the PQC) and classical (the optimizer) parts. However, the model itself can also be hybrid. Essentially, these models are composed of classical and quantum processing parts that together act as hypotheses. To some extent, every QML model within the CQ (classical data, quantum algorithm) domain can be viewed as a hybrid model, as usually the input to the model is subject to some pre-processing $\phi(\cdot)$ (e.g., re-scaling) and, similarly, the expectation values are usually post-processed to match some desired criteria. In this work, a stricter view of hybrid quantum-classical models is adopted, where the pre-processing $\phi(\cdot)$ is some non-trivial mapping, like a

classical neural network and similarly for the post-processing. A prototypical example was introduced in [MBI+20], where classical neural network layers are employed as pre- and post-processing units with the quantum circuit performing the main computations. The authors demonstrated the capabilities of this architecture in transfer learning tasks and labelled it as *dressed quantum circuit*:

$$\mathcal{Q}_{hybrid} = L_{n_q \to n_{out}} \circ Q_{bare} \circ L_{n_{in} \to n_q}. \tag{3.7}$$

Here, $\mathcal{Q}_{hybrid}$ represents the hybrid quantum-classical model, consisting of classical pre-processing layers $L_{n_{in} \to n_q}$ followed by the *bare* PQC Q_{bare} as defined in Equation 3.1 and classical post-processing layers $L_{n_q \to n_{out}}$. Two main advantages were identified: First, the classical layers can learn an optimal embedding of the input data and post-processing of the expectation values. Second, the hybrid model has more flexibility in the number of input and output values and is not limited by the number of encoding gates used in the PQC. This is indicated by the subscripts of the classical layers $n_{in} \to n_q$ and $n_q \to n_{out}$. In [LSI+20], this architecture was adapted to perform image classification: A pre-trained ResNet ($L_{n_{in} \to n_q}$) processes an image, with its last layer replaced by a linear layer that outputs a 2-dimensional intermediate feature vector. These features serve as input into a parameterized quantum circuit (Q_{bare}). After training the linear layer in conjunction with the PQC, the classical layer learns to organize the data into patterns that the quantum circuit can distinguish, achieving perfect separation of the classes.

3.2 Quantum Q-Learning

This section first describes the integration of hybrid quantum-classical models into the deep Q-learning framework. Further, it highlights cutting-edge results in the field of quantum reinforcement learning, focusing on PQC-based methods in both value-based and policy-based settings.

3.2.1 Hybrid quantum-classical models in approximate Q-learning

Just as neural networks serve as Q-function approximators in the DQL setting, PQCs or hybrid models based on PQCs approximate the Q-function in an approximate Q-learning setting that might then be called *quantum Q-learning*. Figure 3.2 illustrates the DQL pipeline, based on Algorithm 2.1 in Section 2.2.4. Here the ϵ-greedy policy and the minibatch sampling from the replay buffer are ignored for simplicity. While the online model Q and target model $\hat{Q}$ in Algorithm 2.1 are purely classical, here they are replaced by hybrid quantum-classical models, $\mathcal{Q}$ and $\hat{\mathcal{Q}}$, respectively. Like in the classical case, the online and target hybrid models are used to break correlations between the target and estimated Q-values and to stabilize training. Both models follow the structure introduced in Equation 3.7, and the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^-$ refer to all trainable parameters in the hybrid models, that is, to the weights of the classical layers $L_{n_{in} \to n_q}$ and $L_{n_q \to n_{out}}$ as well

as to the angles of the parameterized rotations in the PQC Q_{bare}. Assuming that the experience replay memory is at its full capacity, a training iteration proceeds as follows:

The online model processes a state s_t as observed in the environment and predicts Q-values $\mathcal{Q}(s_t, a; \boldsymbol{\theta})$ for each possible action a in that state. The action a that results in the largest Q-value, $a_t = \text{argmax}_a \mathcal{Q}(s_t, a; \boldsymbol{\theta})$, is executed in the environment. The resulting sequence (s_t, a_t, r_t, s_{t+1}) is stored in the experience replay memory. A sequence (s_j, a_j, r_j, s_{j+1}) is randomly selected from the replay buffer. The online model is given the state s_j and action a_j to predict the corresponding Q-value. The state s_{j+1} is given to the target model, to predict the Q-values for all actions of that next state. The output of the online model, $\mathcal{Q}(s_j, a_j; \boldsymbol{\theta})$, and the target model, $\max_{a'} \hat{\mathcal{Q}}(s_{j+1}, a'; \boldsymbol{\theta}^-)$, together with the reward r_j from the replay buffer are used to calculate the loss 2.14. The parameters of the online model $\boldsymbol{\theta}$ are updated according to 2.15 and every C steps, the target model parameters are set to match the online model parameters $\hat{\mathcal{Q}} = \mathcal{Q}$.

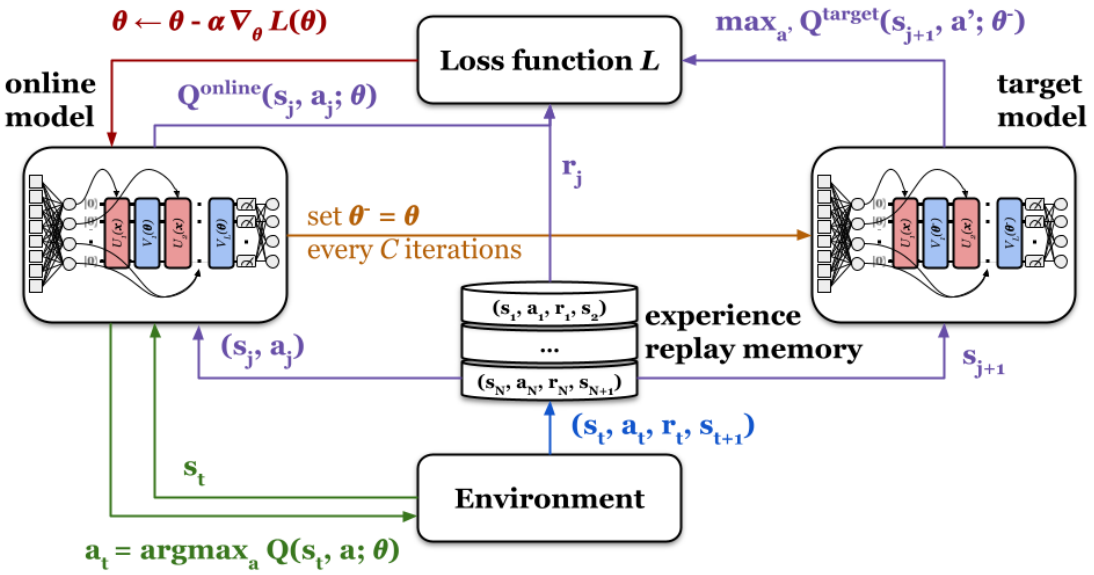


Figure 3.2: Schematic of the Deep Q-Learning framework using hybrid quantum-classical models. The online model $\mathcal{Q}$ (here Q^{online}) interacts with the environment (green arrows), which in turn stores state transition sequences in the experience replay memory (blue arrow). Such (s_j, a_j, r_j, s_{j+1}) sequences are drawn at random from the replay memory and used by the online model and target model $\hat{\mathcal{Q}}$ (here Q^{target}) to make predictions. Together with the observed reward r_j, these predictions are used to compute the loss (purple arrows). The online model parameters $\boldsymbol{\theta}$ are updated by performing a gradient descent step based on the gradient of the loss function with respect to the model parameters (red arrow). Parameter synchronization between the online and target model occurs every C steps (orange arrow).

3.2.2 State of the art in QRL research

Research in QML has predominantly focused on supervised and unsupervised learning tasks, while fewer studies have considered reinforcement learning as a testbed for variational quantum algorithms. In [MUP+24], the authors provide a comprehensive overview of current QRL research, including NISQ-compatible VQA-based approaches as well as methods suited for fault-tolerant hardware. In this section, the focus lies on the former, as it is the path taken in this work. Specifically, the application of quantum and hybrid quantum-classical techniques to classical problems is discussed. This selection is not exhaustive but highlights results that border or intersect with the research conducted in this thesis. The interested reader is referred to the aforementioned publication for a wider view of the field of QRL.

Value-based QRL

As discussed in Section 2.2, the objective in value-based RL (and QRL) is to approximate the Q-function, which in turn is used by the agent to derive optimal actions in each given state.

The first efforts to transfer this approach to the QRL domain were made by [CYQ+20]. Inspired by the seminal work of [MKS+15], the authors employ PQCs in conjunction with classical bias terms on the outputs as Q-function approximators in a Double DQN fashion. As their benchmark, they selected the environments FrozenLake (OpenAI Gym [BCP+16]) and CognitiveRadio, both exhibiting a discrete state space. Using basis encoding (Section 3.1.3) to encode environment states in the quantum system, the authors demonstrate the learning capabilities of the quantum model and argue about advantages in memory consumption and a reduction of model parameters, compared to the classical case.

The authors of [LS20] extend this study by adapting the encoding techniques to also comfort with continuous state spaces. In the OpenAI Gym environments CartPole and Blackjack, the authors propose using fixed rotations to qubits based on the sign of the input data - a form of rotation encoding, see Section 3.1.3. Comparing purely quantum and hybrid models, they demonstrate non-trivial learning outcomes, however, not solve the environments as per their definition [BCP+16].

In [SJD22] a PQC is used as a Q-function approximator in the OpenAI Gym environments FrozenLake (discrete states) and CartPole (continuous states) using basis encoding in the discrete setting and rotation encoding in the continuous environment. The authors highlight the importance of the right choice of observable to match the range of possible Q-values and further introduce trainable weights on the observable to allow for flexible scaling of the observable's expectation values. Performing comprehensive ablation studies with different settings of their design choices, the authors find that a combination of data re-uploading [PSCLGF+20, SSM21] and trainable scaling parameters on the inputs greatly enhances the quantum agent's performance allowing it to successfully solve the continuous CartPole environment, in contrast to earlier works.

Policy-based QRL

Although not the approach taken in this work, notable results have been achieved in using PQCs as policy-function approximators. Importantly, design choices established in [JGM+21] have been successfully adopted to the value-based approach in [SJD22] mentioned in the previous section.

The first studies to assess the performance of PQC-based models in a value-based setting were conducted in [JGM+21]. The authors present a rigorous value-based approach to tackle the continuous state space environments CartPole, MountainCar and Acrobot from the OpenAI Gym. In their work, the authors highlight important design choices in their hybrid models, most notably: trainable scaling parameters on the inputs encoded in rotations as suggested in [PSCLGF+20] and [SSM21], trainable weights at the output of the PQC, together with an adjustable softmax function to better control exploration-exploitation in the environment. With these design choices, their proposed quantum models solve the OpenAI Gym environments with similar performance as classical deep learning approaches and demonstrate an empirical advantage over their classical counterparts in constructed environments based on PQCs. The authors further prove a theoretical exponential advantage in learning performance for quantum agents over classical agents. This is achieved by adapting separation results from [LAT21] based on the classical hardness of the discrete logarithm problem to a policy-based RL setting.

Further extending the work of [JGM+21] on parameterized quantum policies, recent studies have explored variations in the parameter initialization and readout strategy [SSB23], classical post-processing [MSP+23a] and techniques to improve the trainability of the proposed algorithms [MSP+23b]. These contributions offer nuanced views on specific details in approximating the policy function using PQCs both theoretically and experimentally.

QRL in high-dimensional state spaces

The studies highlighted so far cover a wide range of model designs, encoding techniques and environments and demonstrated promising results on standard benchmarking tasks from the OpenAI Gym. However, a common characteristic of all these previous works is a rather low-dimensional state space, described by less than 10 variables that can be encoded directly into the quantum circuit. It is thus natural to ask whether suitable hybrid quantum-classical models are capable of performing well in high-dimensional state spaces.

Insights to this challenge are provided in [CCL+23] where the authors train a hybrid model to solve a 20×20 dimensional GridWorld environment. The proposed architecture extends the model in [CYQ+20] by a classical deep neural network that pre-processes the raw inputs. The authors highlight the benefits of the hybrid architecture to be adaptive to problems of higher dimensions, here $\sim$400, and claim a reduced complexity of their hybrid model compared to full-quantum models.

A step closer to real-world problems is taken in [LS21], which extends their previous work [LS20]. Here, the authors study the performance of a hybrid model in the high-dimensional environments of Atari 2600 Pong and Breakout. The state space of both environments is specified by a 3D array of pixels of dimension $210 \times 160 \times 3$, resulting in approximately 100,000 variables. They propose a hybrid architecture consisting of a classical convolutional network for feature extraction and a quantum circuit consisting of quantum convolution operations [CCL19] and "conventional" parameterized layers. They compare two different approaches: either post-processing the expectation values via a classical fully connected layer or applying a quantum pooling operation to match the environment's action space dimension. Although the architecture achieves notable results in CartPole, the proposed model fails to learn in the high-dimensional Atari game environments of Pong and Breakout.

It remains thus unclear whether PQCs together with classical pre- and post-processing constitute viable models in high-dimensional RL settings that can hold up with their classical counterparts. It is the overall aim of this work to develop and study such a quantum-classical framework and further investigate important design choices and the interplay of the classical and quantum components in order to assess the performance of near-term QRL algorithms on close-to-real-world problems.

CHAPTER 4

Methodology

The previous chapter introduced the fundamental concepts required to describe the hybrid quantum-classical RL framework. This chapter elaborates on the implementation of these principles to make the application of the hybrid model in a deep Q-learning setting tangible. Most importantly, the hybrid quantum-classical model as used in this work is introduced, highlighting the architecture and interplay of classical and quantum components. This model serves as a quantum baseline, against which different design choices in the architecture and hyperparameters are compared. Additionally, a purely classical model featuring an artificial bottleneck is described, which serves as a fair reference for the hybrid models. Besides the models, the game environments used to assess the agents' performance are introduced. Further, the game mechanics, the action space, and reward system, custom to each environment, are detailed. A crucial part is the pre-processing of the environment states issued to the agent, which is explained in detail. Lastly, all relevant numerical experiments conducted are listed together with the exact parameter settings and the rationale behind each one.

4.1 The Hybrid Quantum-Classical Model

The central component of the hybrid quantum-classical RL framework is a hybrid quantum-classical model, which serves as the Q-function approximator in the approximate Q-learning setting. As introduced in Chapter 3, what makes the model *hybrid* is not only the training procedure, which is partially carried out on classical hardware but also the structure of the model itself. Similarly to the approach taken in [MBI+20], the hybrid model consists of classical pre- and post-processing layers effectively encapsulating the parameterized quantum circuit. Figure 4.1 sketches the hybrid model used in this work: The model first consists of three convolutional layers, which, together with a subsequent fully connected (dense) linear layer, act as a dimensionality reduction and feature extraction unit. The terms *linear layer* and *linear activation* refer to the case

where the activation function in each neuron is the identity function. The role of the linear layer is to gather the still high-dimensional output of the convolutional layers and further compress the latent representation down to a manageable number (16 or 36, depending on the setting) of *latent features*, as determined by the number of encoding gates in the quantum circuit. An additional functionality of this pre-processing layer is the scaling of the inputs to the PQC, which adds additional flexibility as theorized in [PSCLGF+20, SSM21] and empirically observed in [JGM+21, SJD22]. The latent output of the linear layer is then used as an input for the encoding gates of the PQC. The PQC acts as the main, *quantum processing unit* of the hybrid model, operating on the latent features generated by the preceding layers. The expectation values of each qubit are treated as the output of the PQC and are passed to another linear layer, which serves two purposes: First, as with the pre-processing linear layer, it allows for rescaling of the fixed-range expectation values to match the optimal Q-values, which is crucial in a Q-learning setting (see Section 2.2). Second, it gives the model more flexibility with regard to the number of actions in a given environment, meaning that the number of qubits does not have to reflect the number of possible actions, which could be a limiting factor for environments possessing large action spaces.

4.1.1 Detailed architecture of the baseline hybrid model

As introduced in the beginning of this chapter, for the hybrid model, a baseline setting is established which serves as a reference when exploring different hyperparameter settings in the model's architecture and training procedure. The parameters of the convolutional layers are the same as in [MKS+15]: The first layer consists of 32 filters of size 8×8 and a stride of 4, the second layer convolves 64 filters of size 4×4 and a stride of 2 and the third layer uses 64 filters of size 3×3 and a stride of 1. All convolutional layers use the non-linear *rectifier* activation function (ReLU). Following the convolutional layers is a fully-connected layer consisting of $n \times l$ neurons with linear activation, where n is the number of qubits and l the number of layers in the PQC, as explained shortly. This layer is referred to as *pre-processing* layer as it prepares (i.e., further downsizes and rescales) the 3136-dimensional output of the convolutional layers for encoding in the PQC. The 3136 values produced by the convolutional layers are reduced to just 16 or 36, depending on the setting (see Table 4.3).

The parameterized quantum circuit follows a hardware-efficient ansatz (see Section 3.1.4): First a set of three variational gates ($\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$) is applied to each qubit. Following this variational block, a set of **CZ**-gates is implemented in a circular arrangement (i.e., the control $\rightarrow$ target setup is $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 4$, ..., $n-1 \rightarrow n$, $1 \rightarrow n$) to introduce entanglement into the quantum state. Subsequently, a $\mathbf{R}_x$-gate is applied to each qubit, where each gate encodes a distinct latent feature from the pre-processing layer on a separate qubit. In summary, the sequence of blocks in the PQC is as follows: Variational block $\rightarrow$ entangling block $\rightarrow$ encoding block. This configuration is referred to as a single *layer* of the PQC and is repeated l times. Finally, another block of variational gates ($\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$) is applied to all qubits. The latent features produced by the fully connected

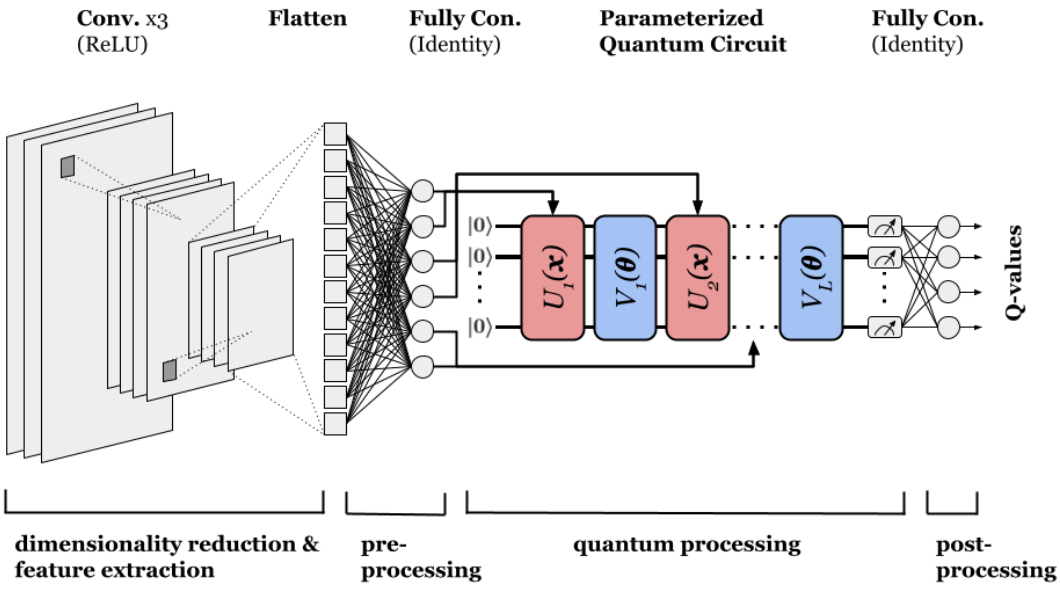


Figure 4.1: The hybrid quantum-classical model. Three convolutional layers act as a dimensionality reduction and feature extraction unit to produce a low-dimensional latent representation of the high-dimensional input. A subsequent fully connected layer with identity activation further pre-processes this latent representation resulting in a limited number of latent features to be encoded in the PQC. The PQC is at the heart of the hybrid model and serves as a quantum processing unit. Local Pauli-**Z** measurements of each qubit are the output of the PQC, which is further post-processed by a fully connected layer with identity activation.

pre-processing layer are encoded layer-wise, where the first n features are encoded in the first encoding block, the second n features in the second encoding block and so forth. Thus, the number of encoding gates in the circuit and, accordingly, the output size of the pre-processing layer is $n \times l$. The total number of trainable parameters in the PQC is $(3 \times n) \times (l+1)$. The expectation value of each qubit is obtained by local (i.e., qubit-wise) Pauli-**Z** measurements. Figure 4.2 depicts the PQC structure with $n = 3$ and $l = 1$. In the default setting used as a baseline, the hybrid model exhibits a PQC of $n = 4$ qubits and $l = 4$ layers and thus contains 60 tuneable parameters and accommodates 16 latent features in the quantum circuit. The final (or output) layer of the hybrid model is again a fully connected linear layer, where the number of output neurons is determined by the number of actions available in the respective environment.

In Section 3.1.5 the concept of barren plateaus was discussed. To mitigate the risk of encountering this undesirable phenomenon and avoid trainability issues, we adopt a Gaussian initialization of the variational parameters θ_i according to the following distribution: $\theta_i^{init} \sim \mathcal{N}(0, \sigma^2)$ where $\sigma = 0.01 \cdot \pi$. This variance was chosen to lie well within the bounds of $\mathcal{O}(1/L)$, where L is the number of single-qubit rotations per qubit.

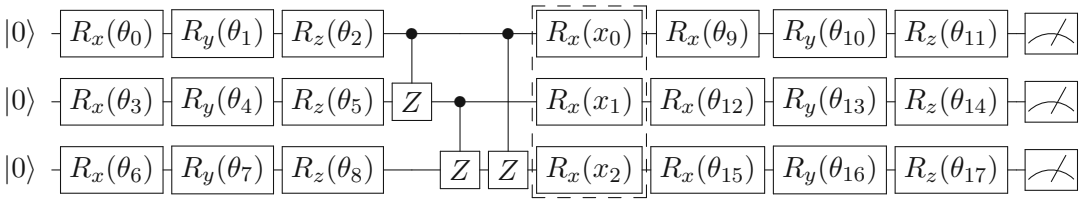


Figure 4.2: Quantum circuit diagram of the PQC with 3 qubits and a single layer. A block of variational gates is followed by a set of entangling gates in a circular arrangement. The feature encoding block is highlighted by a dashed line. A second variational block comes after the encoding block. At the end of the circuit, local Pauli-**Z** measurements are conducted.

As indicated in [ZLH+22], this choice leads to well-behaved bounds of the gradients.

4.1.2 Classical reference model

In order to have a fair reference with which to compare the performance of the hybrid model, here a classical reference model is introduced. The architecture is inspired by the Deep Q-Network introduced in the seminal paper [MKS+15], which consists of three convolutional layers, a fully connected non-linear layer consisting of 512 neurons with ReLU activation and a final linear output layer, i.e., with identity activation. In this work, this architecture is adapted in order to have a fair comparison with the hybrid quantum-classical model (see Figure 4.3): Arguably, one of the biggest limitations of the hybrid model is its small latent space dimension, i.e., the restricted number of features it can accommodate in the PQC. By introducing an artificial *bottleneck layer* after the convolutional layers and before the non-linear layer in the DQN model taken from [MKS+15], the resulting model is subject to similar constraints in the latent space dimension as the hybrid model introduced in Section 4.1.1. The number of neurons in this bottleneck layer is chosen to be the same as in the pre-processing layer of the hybrid model and a linear activation function is used to be as little invasive as possible.

The next section explains the Atari game environments chosen as a benchmark for the hybrid quantum-classical model.

4.2 Atari 2600 Environments

As a testbed for the hybrid framework, two well-known game environments, Pong and Breakout from the Atari 2600 game collection, were selected. Emulators for both games are provided in the OpenAI Gym [BCP+16]. As discussed in Chapter 2 Section 2.2, there exists a wealth of literature on addressing these environments through different classical RL algorithms, offering insights and references for the experiments carried out in this study. Both environments meet the requirements needed to investigate the initial

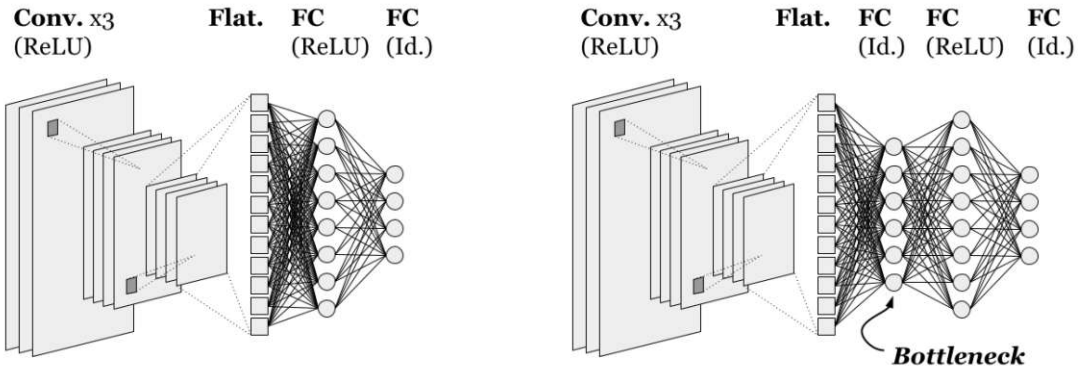


Figure 4.3: An illustration of the model architecture proposed in [MKS+15] (left) and an adapted version obtained by insertion of a linear *bottleneck* layer (right). The original model consists of three convolutional layers, a fully connected (FC) non-linear (ReLU) layer and a final fully connected output layer with identity activation (Id.). The reference model used in this work has an additional layer with relatively few neurons and identity activation right after the convolutional layers and before the fully connected layer. This bottleneck layer drastically limits the size of the latent feature space, creating constraints for the classical reference similar to the hybrid quantum-classical model introduced in Section 4.1.1.

questions posed in this study, particularly, since they have high-dimensional feature spaces represented by RGB images. While both games share similarities in the underlying control problem, their game dynamics exhibit notable differences. Studies indicate that Breakout poses a more complex learning challenge for RL agents [MKS+15], making Pong a suitable proof-of-concept environment to assess the ability of the hybrid model to learn in high-dimensional environments. Therefore, further experimental configurations will be tested in Breakout. This section first outlines the characteristics, goals and game mechanics of both environments and further explains the pre-processing steps applied to the raw environment states before training the agent.

Pong

In the first environment, Pong, the objective is to outscore a computer-controlled opponent in a simulation similar to tennis: the player controls a paddle, moving it vertically to hit a ball back and forth competing against the computer that has the same goal. Both players start with a score of zero and a player scores one point when the opponent hits the ball out of bounds or misses a hit. These points translate one-to-one into the rewards obtained by the agent, i.e., scoring a goal gives 1.0 as a reward. No negative rewards are issued when the opponent scores a point. The game ends once either the computer or the agent scores a total of 21 points. By default, the environment allows

for six possible actions, where only three are relevant for playing the game: since the ball is fired automatically when the game starts, the actions "FIRE", "RIGHTFIRE" and "LEFTFIRE" are removed from the action space. The actions "NOOP" (no operation), "RIGHT" and "LEFT" (corresponding to the up and down movement of the paddle) are kept for the agent to choose from.

Breakout

Breakout, the second game, shares similar dynamics with Pong in terms of paddle and ball movement. However, Breakout presents a more complex challenge: The player must use the paddle to hit a ball towards six layers of bricks at the top of the screen. The goal is to break as many bricks as possible. When the ball hits a brick, it bounces back and the brick is destroyed. The player earns points for destroying bricks, with different point values for bricks in different rows. The highest possible score in the game is 432 points, as there are 18 bricks per row and the first two rows give one point per brick, the middle two rows four and the uppermost rows seven points. As in Pong, these scores serve as numerical rewards for the agent. The agent starts with five lives and loses one life each time the ball misses the paddle and goes out of bounds. No negative rewards are issued when the agent loses a life. The game ends once all lives are lost. In Breakout, a player has to start the game manually by selecting the "FIRE" action. In order to avoid long initial learning periods during which the agent must learn to initiate the game, an auto-firing mechanism is implemented, which issues the "FIRE" command automatically upon the restart of an episode or when a life is lost. The "FIRE" action is removed, which leaves the agent to choose from "NOOP", "RIGHT" and "LEFT", like in Pong. Table 4.1 summarizes the environment specifications of both games. More details on pre-processing are discussed in the following section.

Environment characteristics		
Parameter	Pong	Breakout
possible actions	"NOOP", "RIGHT", "LEFT"	"NOOP", "RIGHT", "LEFT"
possible rewards	0.0 or 1.0	0.0, 1.0, 4.0 or 7.0
max. score	21 points	432 points

Table 4.1: Summary of environment specifications. Each game environment supports three possible actions. The reward system in Pong is simple, either a point is scored or not. In Breakout, the number of points scored for destroying a brick depends on the row the brick belongs to. The total number of points achievable in both games differs as well.

4.2.1 Pre-processing of the raw environment state

Several pre-processing steps are applied to both game environments before being used to train the agent. These are best practices suggested in [MBT+17], which aim to establish a standard in evaluating RL algorithms on Atari 2600 game environments. It is important to state that these pre-processing steps do not refer to the operations applied by the

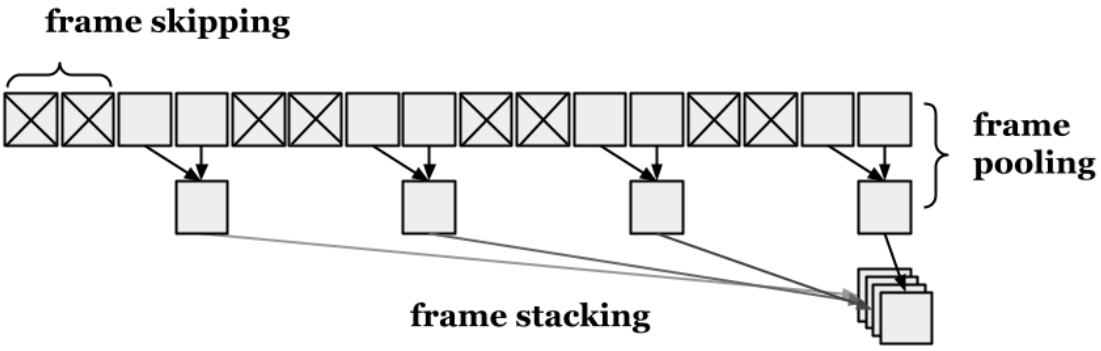


Figure 4.4: The pre-processing pipeline illustrated on a sequence of game frames: A rectangle represents a frame as produced by the game emulator. The crossed-out frames are skipped and not shown to the agent. The last two frames of each sequence of four successive frames are combined using a frame pooling operation. Finally, the pooled frames are stacked to form a single new observation.

pre-processing layers in the hybrid model, but are carried out "manually" before passing the observations to the model.

In its unprocessed form, the observation space of both environments is the RGB image of the game's screen of a given frame, which is specified by a 3D array of pixels of dimension $210 \times 160 \times 3$. This corresponds to an image shape of 210 pixels in height and 160 pixels in width. Each pixel is comprised of three colour channels (red, green and blue), ranging from 0 to 255. As a first step, the observations are grey scaled, which means the number of colour channels is reduced to one, and the image is downsized to 84×84. Instead of using each individual frame of the game, only every fourth frame is considered, a technique called *frame skipping*, which simplifies the reinforcement learning problem and speeds up training. This implies that the agent can only take an action every four frames of the game and all rewards issued between these frames are summed up. Each point in time at which the agent can take an action is called an *environment step* and corresponds to four frames of the game. Since the Atari 2600 hardware is limited in the number of sprites it can render in a given frame, a *frame pooling* operation is applied on two successive frames, combining them into a single frame. In order to encode temporal information into a single observation, four subsequent frames are stacked along the colour channels dimension, an operation referred to as *frame stacking*. The dimension of the final observation is $84 \times 84 \times 4$, where the third dimension does not represent colour information, but the four stacked frames. The frame skipping, frame pooling and frame stacking operations are illustrated in Figure 4.4. A comparison between the unprocessed environment state and the result after pre-processing is depicted in Figure 4.5 on the example of the Breakout environment. The next section highlights the RL framework used to assess the hybrid agent's performance in the two game environments just explained.

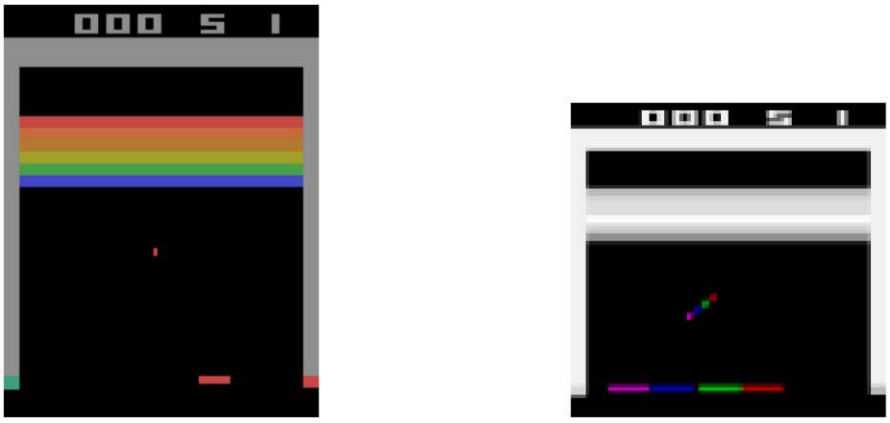


Figure 4.5: Breakout observation space before (left) and after (right) pre-processing applied. The four colours in the right image are for visualization purposes; each colour actually represents one of four distinct frames. The observation dimensions change from $210 \times 160 \times 3$ for the raw RGB image to $84 \times 84 \times 4$ for the cropped, grey-scaled and stacked observations.

A note on stochasticity: The Atari 2600 environments do not provide any source of randomness, which implies that each episode of each game starts in the exact same state and evolves deterministically. This involves the small risk of the agent learning an open-loop control strategy, where the observations made in the environment are irrelevant and the agent learns to issue an optimal sequence of actions by trial and error. As stated in [MBT+17], the ϵ-greedy policy in approximate Q-learning is a suitable source of stochasticity, mitigating this problem. Random experience replay sampling additionally randomizes the agent's behaviour. The strategy employed in [MKS+15], namely randomly enforcing 0 to 30 initial no-operations upon the start of the game, is not used in this work. This leads to higher scores for both hybrid and classical models, which is desirable for the feasibility study and comparisons conducted in this thesis.

4.3 The Hybrid Model in the Q-Learning Framework

For evaluating the hybrid quantum-classical model in a reinforcement learning setting, the deep Q-learning framework with a target model and experience replay is used. The general idea is described in Section 2.2 with the adaptation to hybrid models detailed in Section 3.2.1. Here the exact setting used in this work is presented. Note that most decisions are based on [MKS+15].

Training begins by following a purely random policy for 20,000 environment steps (i.e., 80,000 frames) in order to populate the replay buffer with training samples. Once 100,000 (s_t, a_t, r_t, s_{t+1}) sequences are gathered, the replay buffer operates on a first-in, first-out (FIFO) basis. Experiences are uniformly sampled from the replay buffer. Following this initial warm-up phase, the predictions of the hybrid agent are used to derive a policy

(specified below). The online model is trained every 4 steps and the target model is updated every 8,000 steps in the pre-processed environment or 32,000 frames in the game. For minimizing the loss function explained in Equation 2.14, the Adam optimizer is chosen with a default learning rate of 2.5e-4. Note that in the baseline setting, the learning rate is the same for all parts of the hybrid model (i.e., classical and quantum layers). Throughout the training process, an ϵ-greedy policy is employed, where ϵ is initialized to 1 and gradually decreases to 0.01 following a linear decay schedule over the course of 250,000 environment steps. The discount factor used is 0.99 and remains constant. Training is carried out for a total of 2.5 million environment steps corresponding to 10 million frames.

4.3.1 Numerical experiments conducted

In the previous sections, the architecture of the hybrid quantum-classical model and the game environments used to assess the model's performance were described in detail. Here, the experimentation procedure is explained, starting with the different hyperparameter settings tested in multiple numerical experiments. Experiment runs are repeated multiple times with different initial seeds to compute meaningful statistics and counteract statistical fluctuations. Hyperparameters of interest are the learning rate of the post-processing layer of the hybrid model, the magnitudes of the rewards issued by the environment, the size of the latent space after the linear pre-processing layer as well as the activation function of the pre-processing layer.

4.3.2 Parameter settings

In the following, all hyperparameters that were investigated are listed and motivated. The actual values and combinations tested are summarized at the end of this section in Table 4.3.

Pre-processing activation function: To encode features into a PQC using the rotation encoding technique, as discussed in Section 3.1.3, it is desirable for the variations across all samples of any given feature to be constrained to a maximum of 2π. Values of features that exceed this range introduce ambiguities because of the periodic nature of the controlled rotations. In a hybrid model, employing a particular activation function in the pre-processing layer is an effective method to comply with this constraint. Instances of such functions include a scaled hyperbolic tangent $\pi \cdot \tanh(x)$ or a scaled sigmoid function $2\pi \cdot \sigma(x)$, with images $(-\pi, \pi)$ and $(0, 2\pi)$ respectively. Setting 1 in Table 4.3 refers to this hyperparameter.

Post-processing learning rate: An essential factor to consider is the learning rate of the post-processing layer, which plays a critical role in adjusting the outputs of the PQC and is thus crucial for Q-learning. Results presented in [SJD22] suggest that a higher learning rate for the weights in the post-processing layer benefits the overall performance of the hybrid model. This aligns with the idea that the predicted Q-values

should quickly adapt to the optimal Q-values. Settings 2a to 2f include experiments tuning this hyperparameter.

Reward scaling: Observations of the model's output in the baseline setting (Figure 5.4) indicate that the individual predicted Q-values fall within a close range. This is the case for both the quantum and classical models. This motivates multiplying the rewards given by the environment by a fixed scaling factor $s > 1$ in order to promote a greater absolute separation of predicted Q-values as the scaling effectively increases the target Q-values by the same factor. Different values for reward scaling were studied in settings 2a to 2f.

Latent space dimension: An obvious bottleneck of the hybrid model is the limitation in the number of features encoded in the circuit, as dimensionality reduction always comes at the cost of information loss. This number of latent features (or dimension of the latent feature space), is tied to the number of qubits and encoding layers in the circuit and corresponds to the number of neurons in the pre-processing layer. To address this subject, a model possessing more qubits and layers, and thus encoding gates, is compared to the baseline hybrid model. A larger latent space dimension is investigated in setting 3a and combined with reward scaling in setting 3b and 3c.

The values of the hyperparameters were tested in Breakout, performing an informal search: Each parameter setting was evaluated against the baseline setting (see Table 4.3) running five differently seeded experiments to counteract statistical fluctuations. For the post-processing learning rate and reward scaling (settings 2a to 2f in Table 4.3) a grid search was performed. Due to computational resource constraints, here three runs each were performed. The best-performing hyperparameter settings of the hybrid model were tested as well in the classical setting, listed in Table 4.2.

A post-selection of successful runs was performed in the game of Pong: For the baseline model, 11 runs were performed using the classical reference model and 5 runs using the hybrid model. In 7 out of 11 cases, the classical model did not learn at all in the Pong environment. Only the 4 successful runs were used for evaluation. The hybrid model delivered 4 successful runs out of a total of 5 runs. These were used for evaluation. The unconstrained classical model (depicted in Figure 4.3 on the left) produced 4 successful runs out of a total of 5 runs.

Hyperparameter settings (Classical reference)				
Setting	**Activation$_{\text{pre}}$**	**LR$_{\text{post}}$**	**Reward Sc.**	**Qubits $\times$ Layers**
baseline	identity	2.5e-4	-	4×4
setting 1a	identity	2.5e-2	10x	4×4
setting 1b	identity	2.5e-1	100x	4×4
setting 2	identity	2.5e-4	-	6×6

Table 4.2: The different hyperparameter settings used for the classical reference model. For an explanation of the abbreviations see Table 4.3. For the **Reward Scaling** parameter only those values where chosen, that performed best in the hybrid setting.

Hyperparameter settings (Hybrid model)				
Setting	**Activation$_{\text{pre}}$**	**LR$_{\text{post}}$**	**Reward Scaling**	**Qubits $\times$ Layers**
baseline	identity	2.5e-4	-	4×4
setting 1	$\pi \cdot \tanh$	2.5e-4	-	4×4
setting 2a	identity	2.5e-3	-	4×4
setting 2b	identity	2.5e-3	10x	4×4
setting 2c	identity	2.5e-2	10x	4×4
setting 2d	identity	2.5e-2	100x	4×4
setting 2e	identity	2.5e-1	10x	4×4
setting 2f	identity	2.5e-1	100x	4×4
setting 3a	identity	2.5e-4	-	6×6
setting 3b	identity	2.5e-2	10x	6×6
setting 3c	identity	2.5e-1	100x	6×6

Table 4.3: A summary of the numerical experiment settings. **Activation$_{\text{pre}}$** is the activation function in the pre-processing layer, **LR$_{\text{PQC}}$** and **LR$_{\text{post}}$** are the learning rates of the PQC and the pre-processing layer respectively. **Reward Scaling** indicates which factor for reward scaling was used and **Qubits $\times$ Layers** resembles the latent feature space dimension in terms of the number of qubits and encoding layers.

4.3.3 Plotting the Q-value surface

In Chapter 3, the function classes that PQCs can represent were introduced, referring to a thorough analysis [SSM21, SP21]. There, the equivalence to partial Fourier series, natural to rotational encoding, was demonstrated. To investigate this behaviour in the hybrid quantum-classical model, the following approach is taken: First, the output vector of the convolutional layers (after flattening) is stored for each step of a single episode of Breakout. The resulting collection of 3136-dimensional vectors is used to compute an average output vector of the convolutional network. Then, a copy of the hybrid model is created without the convolutional layers. Next, two of the 3136 input nodes of the truncated model are selected randomly. For plotting the Q-value surface in two dimensions, these two values are varied, while the other input nodes of the truncated model are given their respective average values as computed earlier. The range of variation for the two selected values is selected based on the range observed during the episode that was played. A visualization of the truncated model is provided in Figure 4.6.

4.3.4 Evaluation metrics

As the main evaluation metric for the different agents tested in this work, the learning curves of the agents are used. The learning curve represents the rewards (or scores in the game) the agent obtains while learning to solve the game environments. These rewards show a high variance, even in later stages of training, because the ϵ-greedy policy, together with random sampling from the replay buffer, leads to greatly varying performances of the agent from one game episode to another. For evaluation purposes, a moving average

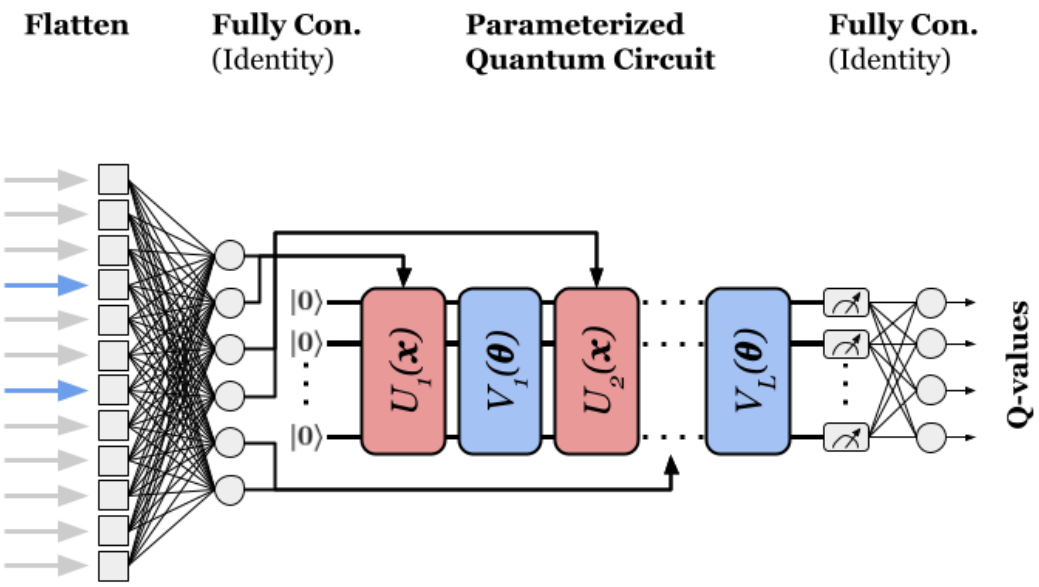


Figure 4.6: A truncated hybrid model is used to plot the predicted Q-values as a function of two values produced by the convolutional layers. All except for two inputs (blue arrows) to the truncated model are kept constant. The two inputs are varied in order to obtain the Q-value landscape as a function of these values.

is applied to the gathered data: First, the total undiscounted rewards for each episode played during training are collected. Second, 10 (in Pong) or 250 (in Breakout) successive episode rewards are averaged to obtain a smoother curve. As suggested in [MBT+17], no evaluation runs are conducted, where the agent fully exploits its learned policy without any parameter updates.

4.3.5 Implementation details

It is crucial to emphasize that the numerical experiments carried out in this research did not utilize real quantum computers. Despite the increasing accessibility of quantum hardware [Mec24], simulating quantum computations offers significant advantages. Simulations provide a setting free from errors and noise, which are common challenges in actual quantum systems. In addition, they allow for a fast evaluation of gradients, making optimization of parameterized quantum circuits straightforward. Simulation also leads to faster prototyping, eliminating the long waiting times typically associated with accessing quantum computing resources. In this study, TensorFlow Quantum [BVM+21] was employed for simulating quantum circuits and models, while TensorFlow Agents [HDV18] was used for implementing reinforcement learning algorithms. The game environments of Pong and Breakout are provided by OpenAI Gym. The codebase of this work can be accessed through the following GitHub link [Fre].

CHAPTER 5

Results and Analysis

In the following sections, the questions raised at the beginning of this work are addressed by presenting and discussing the results of numerical experiments carried out according to the methodology introduced in Chapter 4. Most importantly, the ability of a hybrid quantum-classical model to solve the high-dimensional game environments of Pong and Breakout is assessed first. An analysis of the latent features produced by the pre-processing layer is presented, providing insights into the interplay of the classical and quantum components of the hybrid model. Results of using different learning rates for the post-processing layer are given as well, which plays an important role when reward scaling in the environment is applied. A comparison to the classical reference models introduced in Chapter 4.1.2 is given where suitable. It is also interesting to look at the Q-function landscape the hybrid model produces, especially when compared to its classical counterpart, and what implications this has on learning performance. An evaluation of a sample episode of Breakout as "played" by the baseline model is discussed, where the predicted Q-values are compared to the actual discounted returns. To account for the limitations imposed by the low-dimensional latent space, the scores achieved by models having a higher-dimensional feature space are reported. The chapter ends with a comparison of the best-performing hybrid and classical model in Breakout.

5.1 Performance of Hybrid Baseline and Classical Reference

When looking at the rewards the agents obtained in the course of training, some interesting observations can be made about the hybrid model's performance in both games Pong and Breakout. Figure 5.1 shows these learning curves as an average of five instances of each model (four in the case of Pong) and the standard deviation as a shaded area around each curve. As explained in Section 4.3.2, a running mean was applied to each curve to obtain smoother graphs. The learning curve of the hybrid model in Figure 5.1

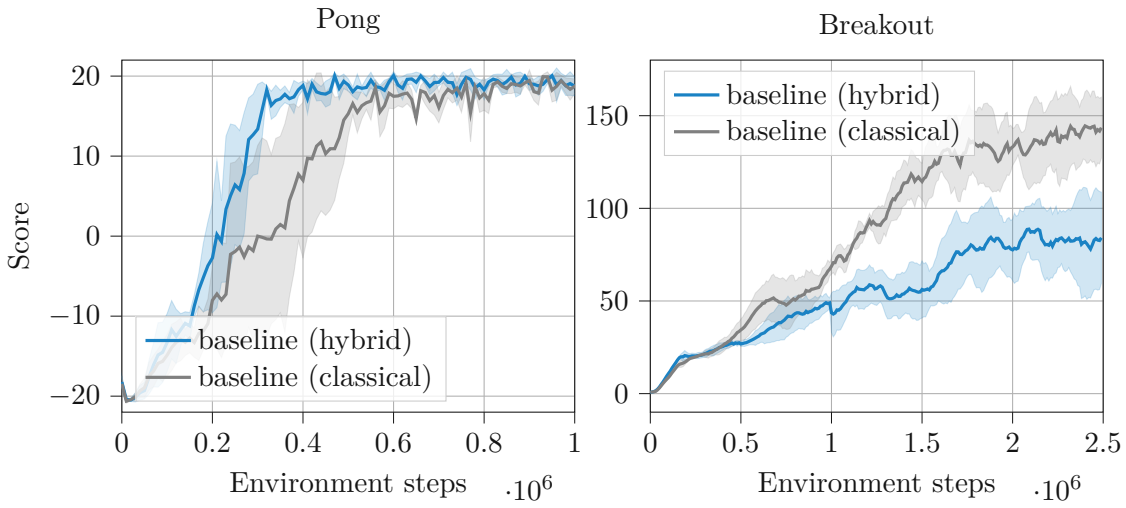


Figure 5.1: Rewards obtained during training for the hybrid and classical model. Shaded areas indicate the standard deviation of multiple runs. Left: The hybrid agent (blue) and the classical reference (grey) show differences in learning performance in Pong. In this environment, the hybrid agent appears to learn faster and more consistently across multiple runs. Right: Hybrid agent (blue) and classical reference (grey) in Breakout. Here, clearly, the classical reference takes the lead.

on the left clearly shows that the model is able to learn and solve the Pong environment with scores close to 20 after 400,000 environment steps. The classical reference reaches the same level of performance, however, takes significantly longer to do so. The standard deviations in both curves indicate that the individual hybrid model runs exhibit more consistent behaviour. The success rate in learning the Pong game is significantly lower in the classical case as fewer runs achieved a score >= 20 overall (see Section 4.3.2) and the performance of the successful runs is noticeably lower.

However, the classical reference without the constraint of a bottleneck layer scores drastically better (see Figure 7.1 in the Appendix 7). In that case, the model achieves average scores of 20 after around 200,000 steps and does so in a more consistent manner; that is, the standard deviation between individual runs is marginal. This suggests that the bottleneck in the reference model might be too much of a constraint for the single non-linear layer in the reference model.

For the hybrid model, the results in Pong indicate that the high-dimensional input of the game environment is effectively encoded into a low-dimensional latent representation that can further be processed by the PQC to compute optimal Q-values. This observation is further supported by the rewards the hybrid agent achieves in Breakout, which is a more complex learning task. In Figure 5.1 the scores achieved in the game of Breakout are depicted on the right. Within 2.5 million environment steps the agent obtains on average

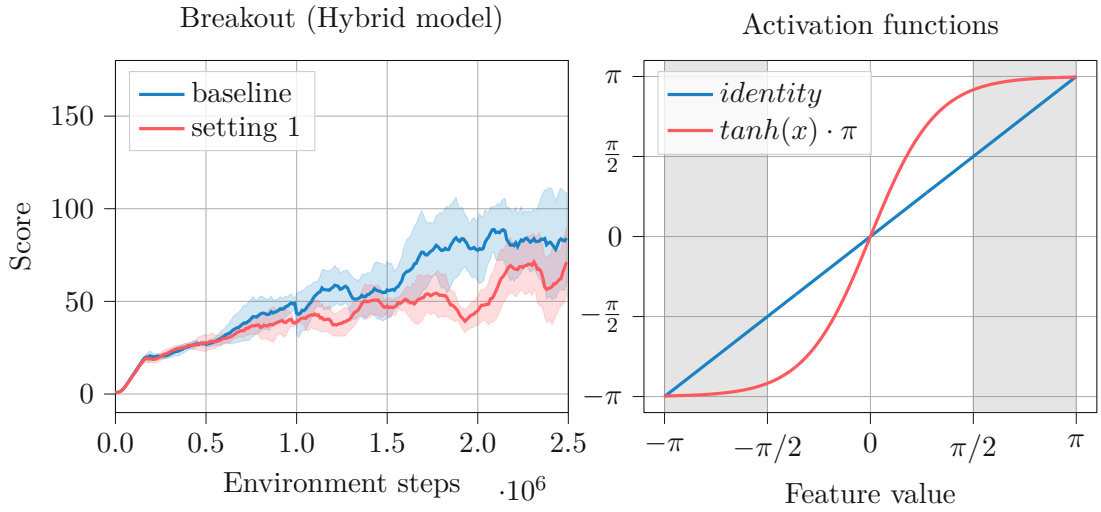


Figure 5.2: Left: Scores achieved by the baseline hybrid model (baseline) and a hybrid model with tanh $\cdot\pi$ activation function in the pre-processing layer (setting 1) in Breakout. The tanh $\cdot\pi$ activation in the pre-processing layer leads to worse performance. Right: Comparison of the identity function (blue) and a *tanh* function scaled by a factor of π (red). Applying tanh $\cdot\pi$ to feature values in the shaded area $\pm(0.5\pi, \pi)$, maps them to a very narrow region in the image of tanh $\cdot\pi$, possibly resulting in information loss.

rewards slightly above 80. In Breakout, the classical reference model with the bottleneck layer performs significantly better compared to the hybrid model. However, also in this setting, the bottleneck imposes a strict limitation, as can be seen when comparing the performance of the classical reference with the unconstrained version (see Figure 7.1 in the Appendix 7). Considering that the hyperparameters were originally optimized for the classical agent, the performance of the hybrid model is particularly promising, suggesting a strong potential for further improvements of this model.

5.2 Effects of Activation Function in Pre-Processing Layer

As discussed in Section 4.1, the pre-processing layer uses unrestricted identity activation functions, which implies that there is no limitation to the magnitudes of the values of individual latent features produced by this classical layer. However, the PQC encodes inputs as rotations, therefore, the values of distinct features must not exceed a range of 2π. One way to enforce that this range is not exceeded is by using a scaling activation function as outlined in 4.3.2.

Figure 5.2 (left) compares the baseline hybrid model to a hybrid model with $tanh \cdot \pi$ activation function (setting 1). The baseline model performs noticeably better, indicating that scaling latent features before encoding might not be necessary. A reason for the poor behaviour of the scaling activation function might be the following: Comparing the

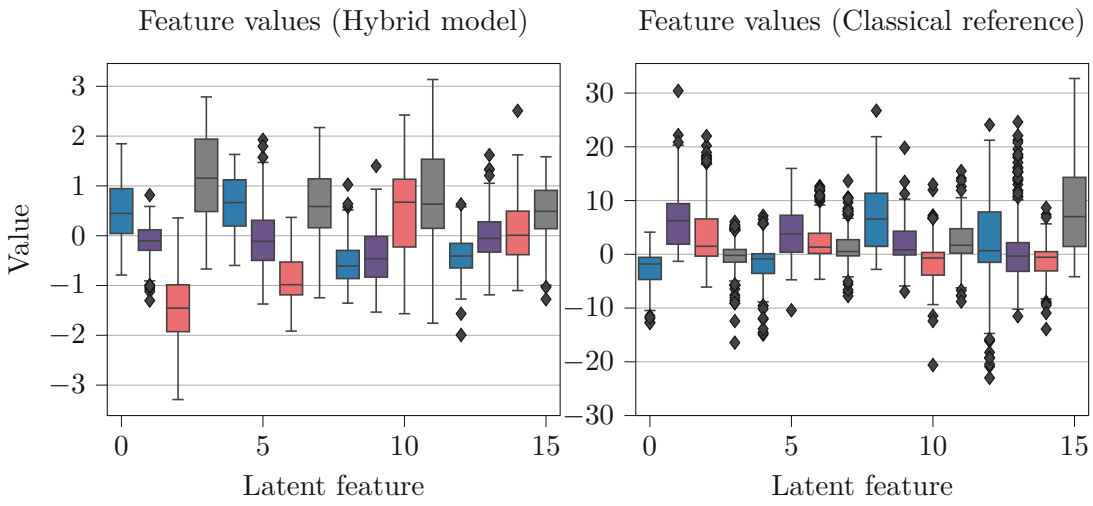


Figure 5.3: Typical values of individual latent features produced by the pre-processing layer in the course of training. Left: The latent feature values of the hybrid model each fall within a range of $\pm\pi$. Right: The latent features of the classical reference exceed this range. This indicates that the classical pre-processing components of the hybrid model learn to adhere to the constraint imposed by rotational encoding.

graphs of an identity function and $tanh \cdot \pi$ (Figure 5.2, right), shows that the image of feature values close to $\pm\pi$ is compressed to a very small interval. This distortion might influence the impact of large latent features (in absolute terms) negatively and lead to less relevant information encoded in the PQC.

Results in Figure 5.3 (left) suggest that a linear activation function in the pre-processing layer suffices: The distributions of the values of each individual feature, which were sampled every 10,000 environment steps during training, show that all values of a specific feature remain well within a range of 2π. Indeed, most of the values fall within a range of π or less, which indicates that the classical layers preceding the PQC, especially the linear pre-processing layer, "learn" to regulate the latent feature magnitudes throughout the training process. This observation is further supported by the range of feature values produced by the pre-processing (or, in this case, bottleneck) layer of the classical reference model in Figure 5.3 (right). Since the purely classical model does not suffer from the limitation imposed by the rotation encoding, the magnitudes of features go well beyond the $\pm\pi$ range observed in the hybrid model.

5.3 The Q-function Surface

In Section 4.3.3, the procedure for comparing the Q-value function learned by the hybrid and classical models was explained. Figure 5.4 depicts the output of the classical (top) and hybrid (bottom) model as a function of two randomly chosen inputs to the pre-processing layer. The close-up figures on the left highlight a section of the input domain, that reflects

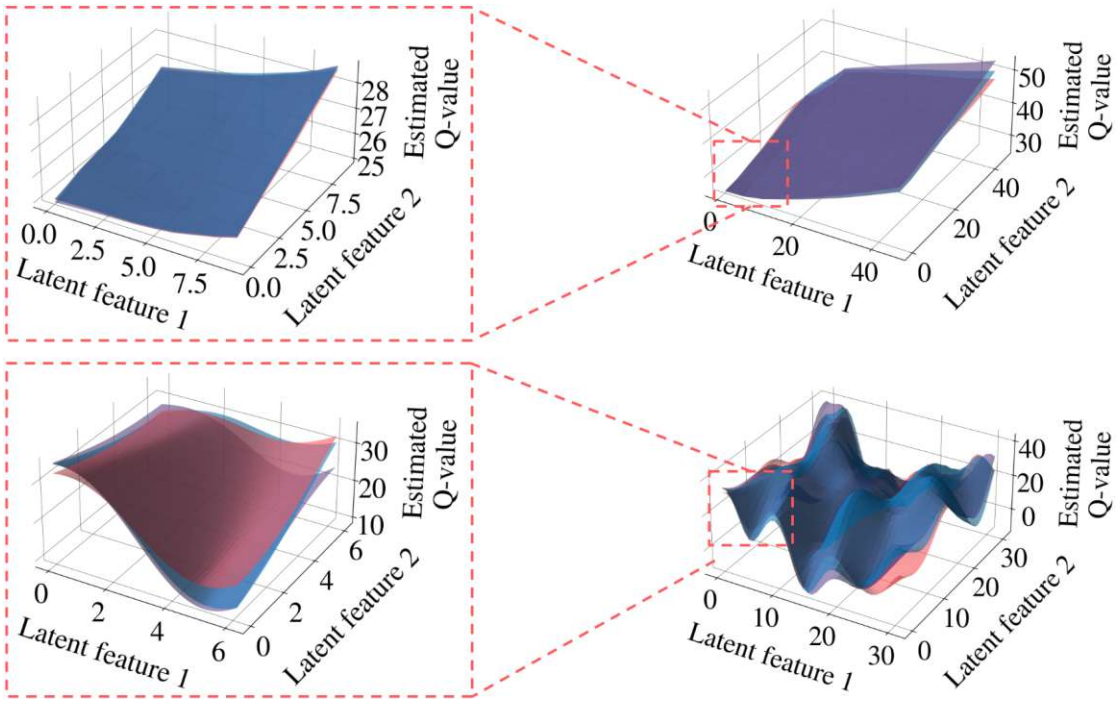


Figure 5.4: Q-value surface as obtained from the classical (top) and hybrid model (bottom). The Q-value predictions are plotted as a function of two randomly chosen inputs to the pre-processing layer. For a detailed explanation see Section 4.3.3. Left: Close-up of the Q-value surface, where the range of input values reflects a typical range as observed during an episode of Breakout.

a typical range of values observed during an episode of Breakout. Here, the piece-wise linear behaviour of the Q-function produced by the classical model differs significantly from the sinusoidal function obtained from the quantum model. The linear regularity of the classical model is a direct consequence of the sole presence of *relu* and linear activation functions in all layers of the model. The graphs on the right depict a wider section of the feature domain providing a pronounced view of the different function shapes produced by the classical and quantum models. The impact of the different Q-function shapes on the agent's learning performance can only be motivated: Naturally, a smooth and continuous function landscape is preferable. High oscillations in the Q-value hyperplane might lead to regions where the Q-value of an otherwise sub-optimal action takes on the largest value, leading to a wrong action selected by the agent.

5.4 A Sample Episode

In addition to the shape of the Q-value landscape, insights can be gained by comparing the Q-value estimates of the agent with the actual discounted returns. In Figure 5.5, the hybrid model's predictions (illustrated by the green graph) and the returns observed in

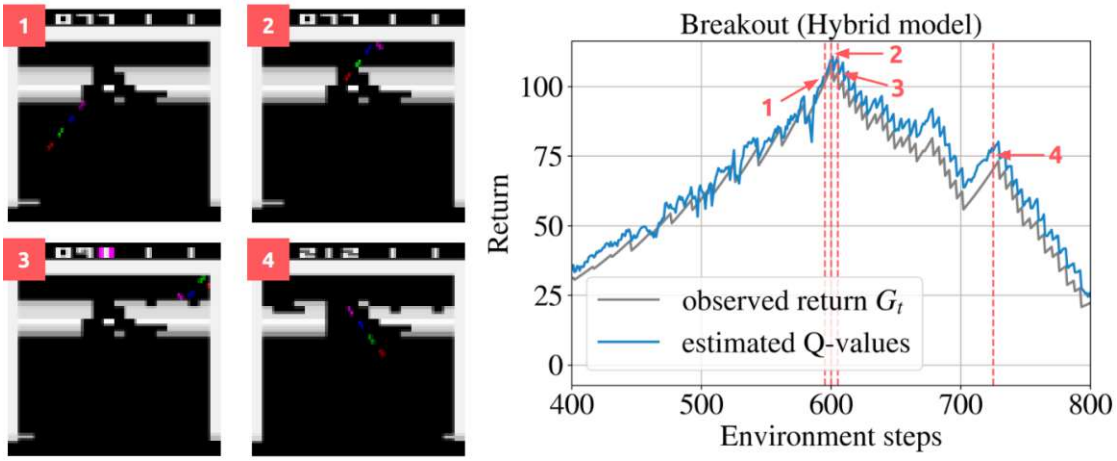


Figure 5.5: Left: Screenshots at selected timesteps of a sample episode, shortly before, during and after the ball passes the layer of bricks. Right: The according predicted Q-values and target Q-values. The predictions closely follow the pattern of the target values, but are slightly overestimated. Both predicted and target Q-values peak whenever the ball is about to pass the layer of bricks, shortly before it bounces back an forth between the top of the screen and the top row of bricks.

the environment (depicted by the blue graph) during an episode of the Breakout game are displayed. Four red vertical bars highlight specific moments in the in-game state - just before, at, and after the peak rewards achieved in the episode. Corresponding snapshots of the game screen for these markers are shown in Figure 5.5 on the left. The Q-value curve reveals a common phenomenon observed in classical deep Q-learning algorithms, where Q-values tend to be overestimated due to the *max* operation when forming the TD target in the approximate Q-learning update rule. Nevertheless, the predicted Q-values closely approximate the target Q-values, showing similar patterns throughout the episode. Both curves display a similar behaviour as observed in [MKS+15]: The Q-values sharply rise shortly before the ball traverses the layers of bricks (1). The maximum Q-values are obtained once the ball reaches the top of the screen (2) when the agent anticipates the highest rewards due to the ball bouncing back and forth between the upper boundary of the screen and the top rows of bricks. Subsequent actions are less likely to yield such high rewards since the top two rows offer the highest scores (7 points per brick), and it will require some effort for the agent to navigate the ball through these rows again. This scenario occurs in step (4) where the agent successfully guides the ball to the top of the screen, resulting in a second, albeit lower, peak in the Q-values.

5.5 Effects of Reward Scaling and Learning Rate

Observing the Q-values for each action as predicted by the hybrid agent reveals that they lie close together in most cases (see Figure 5.4). This behaviour is not unique to the

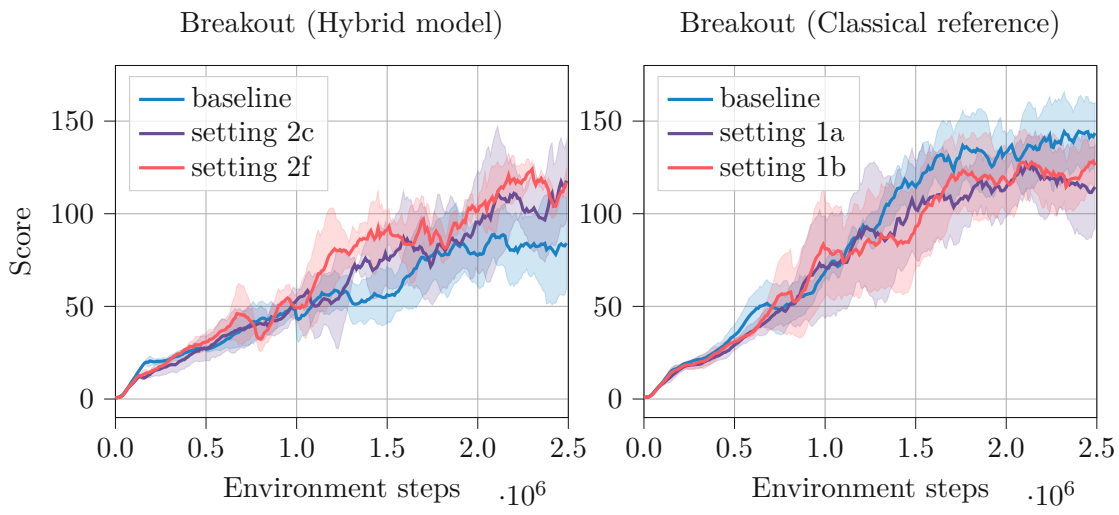


Figure 5.6: *Left*: The hybrid baseline model (blue) as well as a hybrid model trained with 10x scaled rewards and a final layer learning rate of 2.5e-2 (setting 2c) and a hybrid model trained with 100x reward scaling and a final layer learning rate of 2.5e-1 (setting 2f). *Right*: The classical reference model (blue) and a classic reference trained with 10x reward scaling and final layer learning rate of 2.5e-2 (setting 1a) and 100x reward scaling and final layer learning rate of 2.5e-1 (setting 1b). While the hybrid model benefits from the modifications, the performance of the classical model deteriorates.

hybrid agent but also applies to the classical reference model. This motivates re-scaling the rewards issued by the environment to the agent by a constant factor in order to promote a greater separation in the Q-values predicted by the model. Closely related to the magnitude of the target Q-values is the learning rate of the post-processing layer following the PQC in the hybrid model. As discussed in Chapter 4, different combinations of reward scaling factors and learning rates of the pre-processing layer were tested. The best results are presented in Figure 5.6 (left) and compared to the hybrid baseline in blue. Table 4.3 in Section 4.3.2 explains the different parameter settings shown. In both cases, the graphs show an average of three models each. From the graphs, it is evident that the scaled rewards combined with a higher learning rate in the final layer of the hybrid model lead to significantly higher rewards (setting 2c and 2f). For both cases, the average reward clearly surpasses 100 and gets close to a score of 120. Results from literature [SJD22] support the observation that higher learning rates in the post-processing part are beneficial. Rewards obtained in settings 2a, 2b as well as 2d and 2e of Table 4.3 are shown in the Appendix 7 in Section 7.2. The classical model does not benefit from scaling the rewards nor a higher learning rate in the final layer: In Figure 5.6 (right) the same settings as for the hybrid model are used for the classical reference (setting 1a and 1b), both of which exhibit lower scores of around 120 which is lower than the classical reference shown in blue.

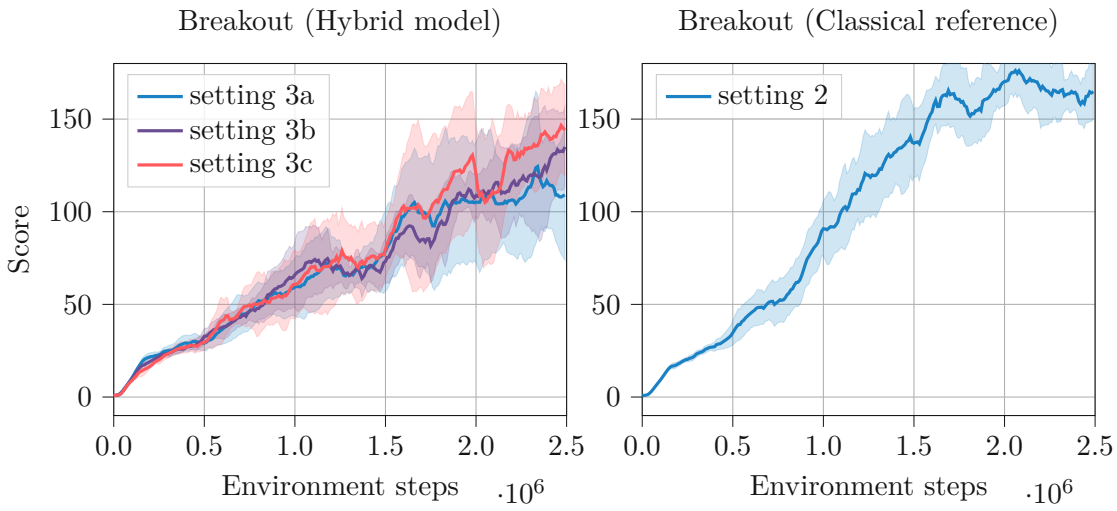


Figure 5.7: Left: Scores achieved by the hybrid model when the latent feature space is increased by using more encoding gates in the PQC and expanding from 4 to 6 qubits in setting 3a. Settings 3b and 3c combine these relaxed constraints with reward scaling and higher learning rates in the post-processing. Right: The classical reference with increased latent space dimension also achieves noticeably higher scores when the constraint imposed by the bottleneck layer is loosened.

5.6 Effects of Latent Space Dimension

Presumably, the most limiting factor for the performance of the hybrid (and classical) models is the dimension of the latent space representation following the convolutional layers. The removal of the bottleneck layer in the classical model leads to a drastic increase of the maximum rewards achieved (see Appendix 7). To find out whether a similar observation can be made for the quantum model, the results of a test with a modified baseline model are shown in Figure 5.7.

The left figure shows the scores achieved by the hybrid model. Again, the exact specifications are listed in Table 4.3 in Section 4.3.2. All three settings share one difference contrasting the baseline: The PQC in the hybrid model consists of 6 qubits (instead of 4 as in the baseline model) and encodes 6 different features on each qubit (instead of 4). This loosens the tight bottleneck of the baseline hybrid model from 16 latent features to a latent space of dimension 36, more than double the size. Settings 3b and 3c additionally have scaled rewards (10x and 100x respectively) and increased learning rates in the post-processing layer (2.5e-2 and 2.5e-1 respectively).

Even without reward scaling, the less constrained model, setting 3a, achieves scores over 100, a significant improvement over the baseline hybrid model. These numbers suggest that a small increase in latent space dimension has a similar impact on the

overall performance of the model as tuning multiple hyperparameters as discussed in Section 5.5. Combined with reward-sacling, in setting 3c, the scores converge to a mean reward of almost 150 within the 2.5 million steps training period. This represents the best performance achieved by the hybrid model given the experimental constraints and available computational resources.

To provide a fair comparison, the blue graph on the right of Figure 5.7 shows a classical reference with a relaxed bottleneck layer of dimension 36 and, subsequently, more latent features to process for Q-value predictions. The margin of improvement is relatively similar to that of the hybrid quantum-classical model.

It is important to note that excessively increasing the latent space dimension may negatively impact the hybrid model. Larger latent spaces require more qubits and deeper quantum circuits to encode all the features, which increases the complexity of the quantum circuit. This added complexity can lead to trainability issues, such as the barren plateau phenomenon, where gradients become exponentially small. Exploring this regime numerically is however computationally demanding and beyond the computational resources available for this project.

The results and analysis presented in the previous chapter illustrate the performance and capabilities of hybrid quantum-classical models in high-dimensional reinforcement learning environments, such as Atari Pong and Breakout. Through a series of numerical experiments, various aspects of these models were explored, including their learning rates, activation functions, Q-value surfaces, and the impact of latent space dimensions. These findings not only demonstrate the feasibility of hybrid models in solving complex tasks but also highlight key factors that influence their effectiveness. The following chapter will summarize the main findings, discuss the implications of the results, and outline potential directions for future research to further enhance the application of hybrid quantum-classical models in high-dimensional reinforcement learning scenarios.

CHAPTER 6

Conclusion

Parameterized quantum circuits have become the preferred approach for quantum machine learning on NISQ devices. Hybrid models that integrate both PQCs and classical processing elements enhance the functionality of these approaches, allowing their application to a wider range of problems. Among the three classic fields of machine learning - supervised learning, unsupervised learning, and reinforcement learning - the latter remains the least studied in quantum machine learning, particularly in high-dimensional state spaces, commonly encountered in in real-world scenarios.

6.1 Findings and Contributions

This thesis provides a framework for assessing the performance of hybrid quantum-classical agents in high-dimensional reinforcement learning settings. Specifically, hybrid models are tested as Q-function approximators in a deep Q-learning setting. The hybrid models consist of a parameterized quantum circuit as well as classical convolutional and fully-connected layers that serve as pre- and post-processing units. Comprehensive comparisons of the hybrid models are conducted against an appropriate classical reference model. The performance of both the hybrid and reference models is assessed in the Atari 2600 game environments Pong and Breakout, as both environments offer a high-dimensional state space, consisting of by game images.

The findings presented in this study demonstrate that hybrid quantum-classical models based on PQCs are capable of learning in high-dimensional RL settings and can solve Atari Pong according to its specifications [ATA97] and surpass a professional human player in the game of Breakout [MKS+15]. This research extends previous studies that first demonstrated the learning capabilities of quantum and hybrid models in an approximate Q-learning setting [CYQ+20, LS20, JGM+21] in simple benchmarking environments from the OpenAI Gym [BCP+16]. In contrast to earlier findings by [LS21], which reported that hybrid models were unable to learn in the Atari environments due to a lack of

expressibility, this study shows that hybrid models are very well capable of solving the environments of Atari Pong and Breakout. This discrepancy suggests that with the appropriate model design and parameter tuning, hybrid quantum-classical models can achieve successful learning outcomes even in complex, high-dimensional RL settings.

The results underscore the importance of careful hyperparameter tuning, revealing that learning rates in the classical post-processing layer, along with scaling of rewards and consequently the Q-values, significantly influence the hybrid model's performance. Furthermore, this research highlights that a primary influence on the model's learning capabilities is the dimension of the latent feature space produced by the classical pre-processing layers and processed by the PQC. Naturally, more features potentially encode more information. However, as the latent feature space dimension grows, the demands for quantum hardware or simulation also increase. More encoding gates are needed, requiring quantum circuits of greater depth and width, which can lead to trainability issues such as barren plateaus.

Finally, the research emphasizes the necessity of establishing a fair classical baseline for comparison with the hybrid model, since an unconstrained classical model clearly outperforms the hybrid model. By subjecting the classical model to the same constraints as the quantum model, it becomes evident that the difference in performance is much less pronounced, and the hybrid model nearly matches the performance of the classical reference.

Overall, these results provide a significant step towards the application of hybrid quantum-classical models in real-world RL scenarios. These results highlight the critical role of hyperparameter tuning and direct focus towards the most impactful design considerations. The results contribute to the understanding of hybrid quantum-classical models, particularly the interplay between quantum and classical components.

6.2 Limitations and Future Research

While this research attempted to evaluate the impact of certain hyperparameters on the performance of hybrid agents, a significant number of them remained unexplored. Notably, the learning rates in the classical pre-processing layer were not modified, nor were those of the PQC. Additionally, the significant reduction in dimensionality enforced by the pre-processing layer could potentially be delegated in part to the convolutional layers, to better capture important spatial features. Ultimately, most of the hyperparameters for both the model and the surrounding DQL algorithm were initially tailored to the classical model proposed in [MKS+15] and were not adjusted for the hybrid model used in this study.

Furthermore, due to computational constraints, it remains an intriguing question as to how the performance of the quantum model improves with the expansion of the latent space dimension, implying less information compression. As the dimension of the latent

space increases, improvements in the performance of the quantum model are anticipated, until a threshold is reached beyond which training difficulties are expected.

Another consideration lies in the environments used to assess the performance of the agents. Although both selected environments exhibit high-dimensional state spaces, their action spaces are relatively simple, with only three discrete actions. An interesting open question remains as to how well hybrid models handle high-dimensional environments offering more complex action spaces.

An important next step is to test this algorithm on real quantum hardware. While the methods used in this work are compatible with today's NISQ devices, it remains to be seen how noise and other technical limitations affect the feasibility and performance of the proposed framework. Testing on actual quantum devices will provide valuable insights into the robustness and practicality of hybrid quantum-classical models in real-world conditions. Additionally, understanding the impact of quantum noise, gate fidelity, and hardware-specific constraints will be crucial for refining these models and enhancing their applicability to more complex and diverse reinforcement learning tasks.

CHAPTER 7

Appendix

7.1 Performance of Classical Model without Bottleneck

Figure 7.1 shows the rewards obtained by the classical model without the limitation of the bottleneck layer in both Pong and Breakout. It is evident that the bottleneck layer creates a harsh limitation, as the model clearly converges much faster and shows fewer fluctuations in Pong and achieves much higher scores in Breakout.

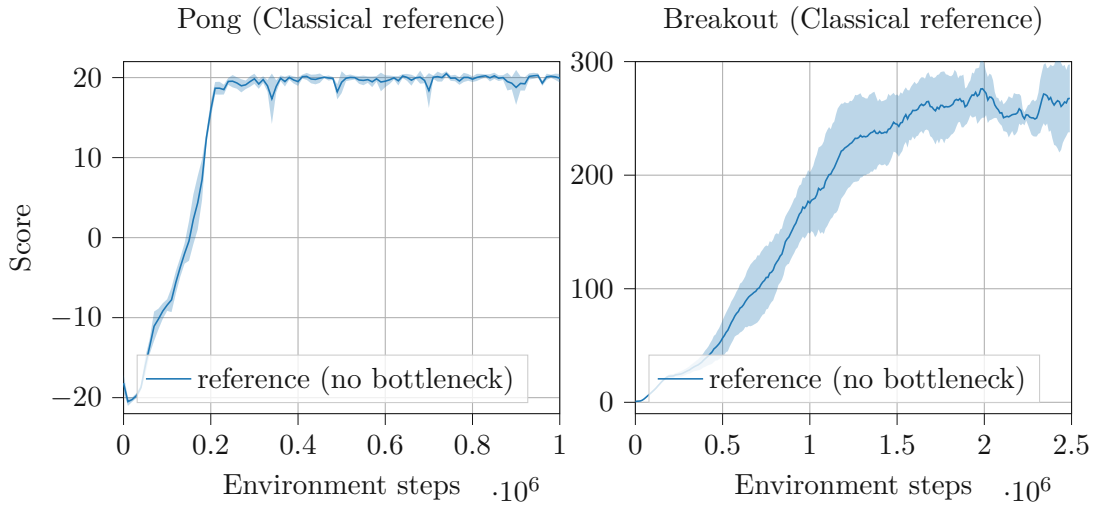


Figure 7.1: Scores of a classical model following the architecture proposed in [MKS+15]. The left figure shows the rewards obtained in the game of Pong and the right figure shows the rewards obtained in Breakout. Without the bottleneck, the classical model achieves significantly higher scores in Breakout and, similarly, much faster converges to the optimum in the case of Pong.

7.2 Additional Settings not listed in the Results

As discussed in Section 4.3.2, a grid search was conducted on the learning rate of the post-processing layer and the reward scaling factor. Settings 2c to 2f in Table 4.3 were evaluated in this grid search, with settings 2c and 2f performing the best. Settings 2d and 2e also showed improvements over the baseline setting, as demonstrated in Section 5.5. In Figure 7.2, the right side displays the scores achieved during training for these settings. The left side of Figure 7.2 illustrates settings 2a and 2b from Table 4.3, where setting 2a uses a learning rate in the post-processing layer that is ten times higher than the baseline. Setting 2b also incorporates tenfold reward scaling. These settings do not result in any improvement over the baseline model.

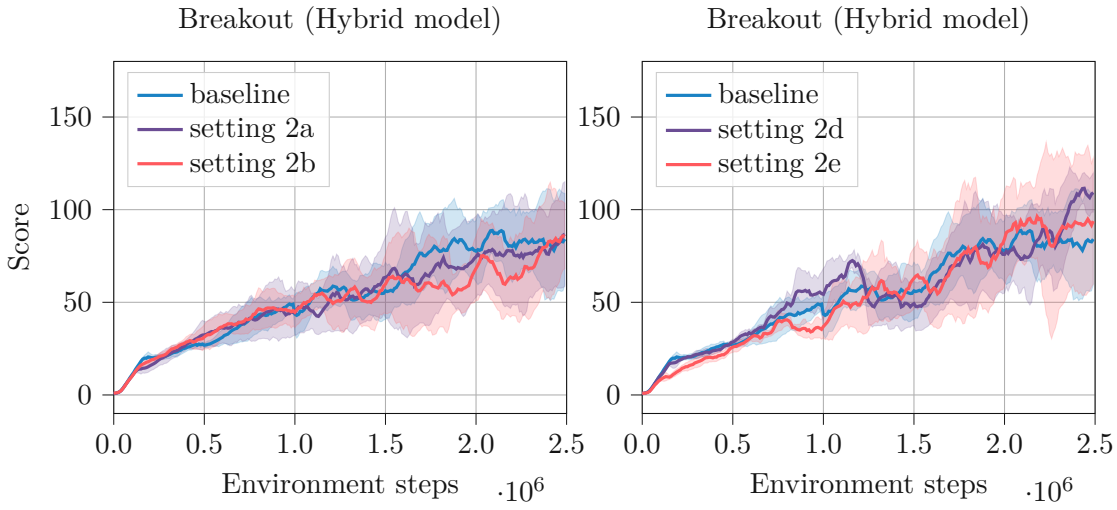


Figure 7.2: *Left*: The hybrid baseline model (blue), along with a hybrid model trained with a final layer learning rate of 2.5e-3 (setting 2a), and another hybrid model trained with 10x reward scaling and a final layer learning rate of 2.5e-3 (setting 2b). *Right*: The hybrid baseline model (blue), a hybrid model trained with 100x reward scaling and a final layer learning rate of 2.5e-2 (setting 2d), and a hybrid model trained with 10x reward scaling and a final layer learning rate of 2.5e-1 (setting 2e).

The better performance of settings 2c and 2f (refer to Figure 5.6) compared to settings 2d and 2e, as shown above, is likely attributed to the alignment of learning rates and reward scaling factors in settings 2c and 2f. The lack of improvement in settings 2a and 2b suggests that a significantly higher learning rate in the post-processing layer, as observed in settings 2c to 2f, is indeed a crucial factor when combined with reward scaling.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AA02] M. Andrecut and M. K. Ali. A quantum neural network model. *International Journal of Modern Physics C*, 13(01):75–88, January 2002. Publisher: World Scientific Publishing Co.

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019. Publisher: Nature Publishing Group.

[AKH+23] Amira Abbas, Robbie King, Hsin-Yuan Huang, William J. Huggins, Ramis Movassagh, Dar Gilboa, and Jarrod R. McClean. On quantum backpropagation, information reuse, and cheating measurement collapse, May 2023. arXiv:2305.13362 [quant-ph].

[Alt01] M. V. Altaisky. Quantum neural network, July 2001.

[ATA97] ATARI. AtariAge - Atari 2600 Manuals (HTML) - Video Olympics (Atari), 1997.

[BCLK+22] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, 94(1):015004, February 2022. Publisher: American Physical Society.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, June 2016. arXiv:1606.01540 [cs].

[BLS+19] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, November 2019. Publisher: IOP Publishing.

[BNS+00] E. C. Behrman, L. R. Nash, J. E. Steck, V. G. Chandrashekar, and S. R. Skinner. Simulations of quantum neural networks. *Information Sciences*, 128(3):257–269, October 2000.

[BVM+21] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J. Martinez, Jae Hyeon Yoo, Sergei V. Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, Evan Peters, Owen Lockwood, Andrea Skolik, Sofiene Jerbi, Vedran Dunjko, Martin Leib, Michael Streif, David Von Dollen, Hongxiang Chen, Shuxiang Cao, Roeland Wiersema, Hsin-Yuan Huang, Jarrod R. McClean, Ryan Babbush, Sergio Boixo, Dave Bacon, Alan K. Ho, Hartmut Neven, and Masoud Mohseni. TensorFlow Quantum: A Software Framework for Quantum Machine Learning, August 2021. arXiv:2003.02989 [cond-mat, physics:quant-ph].

[CAB+21] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, September 2021. Publisher: Nature Publishing Group.

[CBB+23] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O'Brien. Quantum error mitigation. *Reviews of Modern Physics*, 95(4):045005, December 2023. Publisher: American Physical Society.

[CCL19] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum Convolutional Neural Networks. *Nature Physics*, 15(12):1273–1278, December 2019. arXiv:1810.03787 [cond-mat, physics:quant-ph].

[CCL+23] Hao-Yuan Chen, Yen-Jui Chang, Shih-Wei Liao, and Ching-Ray Chang. Deep-Q Learning with Hybrid Quantum Neural Network on Solving Maze Problems, December 2023. arXiv:2304.10159 [quant-ph].

[CMD+20] Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. The Born supremacy: quantum advantage and training of an Ising Born machine. *npj Quantum Information*, 6(1):1–11, July 2020. Publisher: Nature Publishing Group.

[CYQ+20] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational Quantum Circuits for Deep Reinforcement Learning. *IEEE Access*, 8:141007–141024, 2020. Conference Name: IEEE Access.

[FN18] Edward Farhi and Hartmut Neven. Classification with Quantum Neural Networks on Near Term Processors, August 2018. arXiv:1802.06002 [quant-ph].

[Fre] Dominik Freinberger. Spiegeldondi/A-Hybrid-Quantum-Classical-Framework-for-Reinforcement-Learning-of-Atari-Games: This repo explores the capabilities of hybrid quantum-classical models based on variational quantum algorithms (VQAs) in high dimensional reinforcement learning (RL) environments such as the Atari 2600 classics Pong and Breakout.

[Gar23] Gartner. Invest Implications: Forecast Analysis: Artificial Intelligence Software, 2023-2027, Worldwide, September 2023.

[GPC19] Ivan Glasser, Nicola Pancotti, and J. Ignacio Cirac. From probabilistic graphical models to generalized tensor networks for supervised learning, December 2019. arXiv:1806.05964 [cond-mat, physics:quant-ph, stat].

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search, May 1996.

[GWO+19] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, December 2019. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.

[GZ02] Sanjay Gupta and R. K. P. Zia. Quantum Neural Networks, January 2002.

[HDV18] Danijar Hafner, James Davidson, and Vincent Vanhoucke. TensorFlow Agents: Efficient Batched Reinforcement Learning in TensorFlow, October 2018. arXiv:1709.02878 [cs].

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):150502, October 2009. Publisher: American Physical Society.

[IDC23] IDC. Worldwide Spending on AI-Centric Systems Forecast to Reach $154 Billion in 2023, According to IDC, July 2023.

[JFPN+23] Sofiene Jerbi, Lukas J. Fiderer, Hendrik Poulsen Nautrup, Jonas M. Kübler, Hans J. Briegel, and Vedran Dunjko. Quantum machine learning beyond kernel methods. *Nature Communications*, 14(1):517, January 2023. Publisher: Nature Publishing Group.

[JGM+21] Sofiene Jerbi, Casper Gyurik, Simon Marshall, Hans Briegel, and Vedran Dunjko. Parametrized Quantum Policies for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 28362–28375. Curran Associates, Inc., 2021.

[Kak95] Subhash C. Kak. Quantum Neural Computing. In Peter W. Hawkes, editor, *Advances in Imaging and Electron Physics*, volume 94, pages 259–313. Elsevier, January 1995.

[KMT+17] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, September 2017. Publisher: Nature Publishing Group.

[LAT21] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, September 2021. arXiv:2010.02174 [quant-ph].

[LS20] Owen Lockwood and Mei Si. Reinforcement Learning with Quantum Variational Circuit. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):245–251, October 2020. Number: 1.

[LS21] Owen Lockwood and Mei Si. Playing Atari with Hybrid Quantum-Classical Reinforcement Learning, July 2021. arXiv:2107.04114 [quant-ph].

[LSI+20] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning, February 2020. arXiv:2001.03622 [quant-ph].

[LVL22] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model, November 2022. arXiv:2211.02001 [cs].

[MBI+20] Andrea Mari, Thomas R. Bromley, Josh Izaac, Maria Schuld, and Nathan Killoran. Transfer learning in hybrid classical-quantum neural networks. *Quantum*, 4:340, October 2020. arXiv:1912.08278 [quant-ph, stat].

[MBK21] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. Estimating the gradient and higher-order derivatives on quantum hardware. *Physical Review A*, 103(1):012405, January 2021. Publisher: American Physical Society.

[MBS+18] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, November 2018. Number: 1 Publisher: Nature Publishing Group.

[MBT+17] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents, November 2017. arXiv:1709.06009 [cs].

[Mec24] The Quantum Mechanic. Quantum Computers For Public Use? What You Need To Know For Success In The Coming Quantum Age., March 2024. Section: Quantum Cloud.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. arXiv:1312.5602 [cs].

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, February 2015.

[MNK+18] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, September 2018. Publisher: American Physical Society.

[MSP+23a] Nico Meyer, Daniel Scherer, Axel Plinge, Christopher Mutschler, and Michael Hartmann. Quantum policy gradient algorithm with optimized action decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24592–24613. PMLR, 23–29 Jul 2023.

[MSP+23b] Nico Meyer, Daniel D. Scherer, Axel Plinge, Christopher Mutschler, and Michael J. Hartmann. Quantum natural policy gradients: Towards sample-efficient reinforcement learning. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 02, pages 36–41, 2023.

[MUP+24] Nico Meyer, Christian Ufrecht, Maniraman Periyasamy, Daniel D. Scherer, Axel Plinge, and Christopher Mutschler. A Survey on Quantum Reinforcement Learning, March 2024. arXiv:2211.03464 [quant-ph].

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[NDM+21] Rakshit Naidu, Harshita Diddee, Ajinkya Mulay, Aleti Vardhan, Krithika Ramesh, and Ahmed Zamzam. Towards Quantifying the Carbon Emissions of Differentially Private Machine Learning, July 2021. arXiv:2107.06946 [cs].

[PGL+21] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon Emissions and Large Neural Network Training, April 2021. arXiv:2104.10350 [cs].

[Pre18] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.

[PSCLGF+20] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, February 2020. arXiv:1907.02085 [quant-ph].

[PSLNGS+21] Adrián Pérez-Salinas, David López-Núñez, Artur García-Sáez, P. Forn-Díaz, and José I. Latorre. One qubit as a Universal Approximant. *Physical Review A*, 104(1):012405, July 2021. arXiv:2102.04032 [quant-ph].

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum Support Vector Machine for Big Data Classification. *Physical Review Letters*, 113(13):130503, September 2014. Publisher: American Physical Society.

[Sac23] Goldman Sachs. AI investment forecast to approach $200 billion globally by 2025, January 2023.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[SBG+19] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, March 2019. Publisher: American Physical Society.

[SBS+20] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, March 2020. arXiv:1804.00633 [quant-ph].

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP, June 2019. arXiv:1906.02243 [cs].

[Sho95] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, August 1995.

[SJAG19] Sukin Sim, Peter D. Johnson, and Alan Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, December 2019. arXiv:1905.10876 [quant-ph].

[SJD22] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning. *Quantum*, 6:720, May 2022. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.

[SK19] Maria Schuld and Nathan Killoran. Quantum Machine Learning in Feature Hilbert Spaces. *Physical Review Letters*, 122(4):040504, February 2019. Publisher: American Physical Society.

[SP21] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, Cham, 2021.

[SQA+16] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay, February 2016. arXiv:1511.05952 [cs].

[SS16] Edwin Stoudenmire and David J Schwab. Supervised Learning with Tensor Networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[SSB23] André Sequeira, Luis Paulo Santos, and Luis Soares Barbosa. Policy gradients using variational quantum circuits. *Quantum Machine Intelligence*, 5(1):18, April 2023.

[SSM21] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. The effect of data encoding on the expressive power of variational quantum machine learning models. *Physical Review A*, 103(3):032430, March 2021. arXiv:2008.08605 [quant-ph, stat].

[vHGS15] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning, December 2015. arXiv:1509.06461 [cs].

[vN93] J. von Neumann. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. Conference Name: IEEE Annals of the History of Computing.

[Vri23] Alex de Vries. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194, October 2023. Publisher: Elsevier.

[WBL12] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum Algorithm for Data Fitting. *Physical Review Letters*, 109(5):050505, August 2012. Publisher: American Physical Society.

[WSH+16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning, April 2016. arXiv:1511.06581 [cs].

[ZFF19] Zhikuan Zhao, Jack K. Fitzsimons, and Joseph F. Fitzsimons. Quantum assisted Gaussian process regression. *Physical Review A*, 99(5):052331, May 2019. arXiv:1512.03929 [quant-ph, stat].

[ZLH+22] Kaining Zhang, Liu Liu, Min-Hsiu Hsieh, and Dacheng Tao. Escaping from the Barren Plateau via Gaussian Initializations in Deep Variational Quantum Circuits, December 2022. arXiv:2203.09376 [quant-ph].

[ZLW19] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum Generative Adversarial Networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, November 2019. Publisher: Nature Publishing Group.

[ZLW21] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Variational quantum Boltzmann machines. *Quantum Machine Intelligence*, 3(1):7, February 2021.