ACIT5900

MASTER THESIS

in

Applied Computer and Information Technology (ACIT)

May 2023

Applied Artificial Intelligence

A Comparative Study of Classical and Quantum Deep Q-Learning Agents in OpenAI Gym

Mathias André Eriksen Sandnes

Department of Computer Science

Faculty of Technology, Art and Design

1 Preface

I have been fascinated by both machine learning and quantum mechanics for many years. When I attended a talk by Kristian Wold on Quantum Machine Learning during the unveiling of quantum computers at OsloMet in 2021, I became captivated by the intersection of these remarkable fields. Driven to understand the potential of these cutting-edge technologies better and explore their applications in solving OpenAI Gym problems, I decided to focus my thesis on comparing classical and quantum Deep Q-Learning (DQL).

Throughout my research journey, I have been deeply grateful for my advisors, peers, and family's support and guidance. Their encouragement and constructive feedback have been invaluable in shaping this work. I also express my gratitude to the researchers and developers whose publications have laid the foundation for my studies.

This thesis represents the culmination of my exploration into the fascinating world of quantum computing and machine learning. I hope my work contributes to the growing body of knowledge in these fields and inspires others to delve further into these exciting areas.

Oslo, Norway - 15.05.2023 Mathias André Eriksen Sandnes

Keywords

Machine Learning, Reinforcement Learning, Deep-Q Learning, Quantum Mechanics, Entanglement, Parameterized Quantum Circuits with re-uploading, Quantum Computing, Quantum Neural Networks, Quantum Deep Q-Learning, OpenAI Gym

2 Abstract

This study delves into the potential of quantum Deep Q-Learning (DQL) as an efficient approach to solving OpenAI Gym problems, setting it against the backdrop of classical DQL. Within the dynamic landscape of quantum computing, harnessing quantum properties such as entanglement and superposition in Quantum Neural Networks (QNNs). The project focuses on a selection of OpenAI Gym problems, Cart Pole, and Lunar Lander, and specific QNN architectures using Parameterized Quantum Circuits (PQCs) with re-uploading, combined with DQL algorithms. The methodology hinges on implementing and training classical and quantum DQL models, subsequently comparing their performance to highlight key distinctions. Emerging findings suggest that Quantum DQL (QDQL) can attain comparable performance to classical DQL, albeit with considerably fewer trainable parameters. However, it's important to note that this performance may come with increased sensitivity to hyperparameters. A unique aspect of this study is the exploration of trainable entanglement as opposed to the static entanglement commonly addressed in the literature. Preliminary results indicate that a specific level of entanglement can benefit QNNs, although these findings remain inconclusive due to the short duration of training. Therefore, we propose a QDQL agent with trainable entanglement. In conclusion, this research underscores the potential of QDQL in reinforcement learning problems and the trade-offs involved, setting the stage for future investigations and optimizations of quantum computing approaches in machine learning. Future work could delve into a broader range of QNN architectures, alternative entanglement strategies, and diverse problem domains.

Contents

List of Figures

3 Introduction

Quantum computing has emerged as a promising technology with the potential to revolutionize various fields, including cryptography, drug discovery, optimization, and machine learning. In recent years, researchers have started to explore the intersection of quantum computing and machine learning, aiming to develop novel algorithms that can leverage the unique capabilities of quantum systems. One such area of interest is the combination of quantum computing techniques with reinforcement learning, a branch of machine learning where agents learn to make decisions by interacting with an environment.

This thesis combines deep Q-learning with quantum computing, a powerful reinforcement learning technique. By harnessing the computational power of quantum systems, our goal is to enhance the capabilities of deep Q-learning agents, leading to improved performance and efficiency in solving complex tasks. The following paragraphs outline the objectives and scope of our project.

This project explores the potential benefits of combining deep Q-learning (DQL) with quantum computing techniques. Our initial focus is developing and training DQL agents on classical computers and using them to solve simple game environments. We will investigate whether the use of quantum techniques can surpass the performance of these agents beyond state-of-the-art classical approaches in terms of actual performance, training speed, and the number of parameters.

In addition, we plan to tackle more challenging benchmarks, such as previously unsolved environments by quantum agents, to further test the potential benefits of combining quantum computing and DQL. By evaluating the performance of our quantum DQL agents on more complicated problems, we hope to gain insights into the potential advantages of using quantum computing techniques for machine learning tasks on exponentially more complex problems. Ultimately, our goal is to contribute to developing more powerful and efficient machine learning algorithms that can tackle increasingly complex optimization problems.

Previous work comparing classical and quantum agents have either used hybrid

solutions or focused on improving the quantum model more than the classical one. This often results in an unfair comparison, as the quantum models are compared to classical models that have not been optimized to their full potential. Furthermore, in cases where a quantum model is inferior to a classical one, the research is rarely published, creating a bias toward the performance of quantum models. Our work aims to address this issue by comparing classical and quantum DQL agents fairly by optimizing each approach by the same amount.

This thesis focuses on combining quantum computing techniques with reinforcement learning, specifically DQL. Through the development and evaluation of quantum DQL agents, we aim to explore the potential benefits and limitations of integrating quantum computing with reinforcement learning. This research is expected to provide valuable insights into the advantages and challenges of quantum machine learning and contribute to developing more efficient and powerful algorithms for solving complex optimization problems.

3.1 Machine Learning

Classical machine learning is a subfield of artificial intelligence that involves developing algorithms that allow computers to learn from data and improve their performance over time. It has gained significant attention in recent years due to its ability to process large amounts of data and make accurate predictions or decisions in various domains, such as computer vision, natural language processing, and robotics [Bishop, 2006, Goodfellow et al., 2016].

In classical machine learning, various techniques and algorithms are employed to analyze and model data. These methods can be broadly categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves learning from labeled data to make predictions or classifications, while unsupervised learning involves discovering patterns or structures in unlabeled data. Reinforcement learning, on the other hand, focuses on training agents to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties [Goodfellow et al., 2016].

Despite its many successes, classical machine learning approaches have limitations, particularly when solving complex problems requiring extensive data processing or dealing with high-dimensional feature spaces. In these cases, the computational complexity of classical algorithms can often scale exponentially with the problem size, making them impractical for real-world applications [Garey and Johnson, 1979].

The development of quantum machine learning, which combines the principles of quantum computing with classical machine learning techniques, aims to address some of these challenges by leveraging the unique capabilities of quantum systems to achieve significant speedups in data processing and analysis potentially.

3.2 Quantum Machine Learning

Quantum machine learning (QML) is an emerging interdisciplinary field that combines the principles of quantum computing with machine learning techniques. The primary objective of QML is to leverage the unique capabilities of quantum systems, such as superposition and entanglement, to enhance the performance of machine learning algorithms and enable them to tackle increasingly complex problems more efficiently.

One of the critical advantages of quantum machine learning is the potential for significant speedups in the processing and analysis of data. Due to the inherent parallelism of quantum systems, specific quantum algorithms can provide exponential or polynomial speedups over their classical counterparts [Shor, 1999, Nielsen and Chuang, 2000]. This could be particularly beneficial for addressing large-scale optimization problems, high-dimensional data analysis, and other computationally demanding tasks currently intractable using classical methods.

However, implementing quantum machine learning algorithms comes with several challenges, including the need for efficient classical-to-quantum data encoding, robust error correction, and addressing the current limitations of quantum hardware [Nielsen and Chuang, 2000, Terhal, 2015]. One of the main issues with

present-day quantum hardware is the significant amount of noise inherent in the systems. This noise accumulates over time and renders results from long algorithms unreliable, which presents a significant hurdle for complicated, practical applications of quantum machine learning.

Another area for improvement is the availability of only a few qubits in current quantum devices. This restricts the size and complexity of problems that can be effectively tackled using quantum algorithms. As quantum computing technology advances, the number of qubits is expected to increase, allowing for more powerful quantum computations.

In the quest to harness the power of quantum computing for solving complex problems, it is essential to understand the various combinations of data types and processing methodologies that can be employed. This section will examine four distinct scenarios, each representing a unique pairing of classical and quantum data with their respective processing methods. By investigating the intricacies and challenges of each combination, we aim to provide a comprehensive understanding of the landscape and lay the foundation for our primary focus on classical data and quantum processing.

Classical Data and Classical Processing (CC): CC represents the traditional computing paradigm where classical data is processed using classical computing algorithms [Cormen et al., 2009]. This is the most common approach in data science and has proven effective in numerous applications such as data analysis, machine learning, and optimization problems [Bishop, 2006, Goodfellow et al., 2016]. The primary limitation of this approach lies in its inability to solve some complex issues efficiently, as classical algorithms often scale exponentially with problem size [Garey and Johnson, 1979]. Figure 3.1 shows the four ways of combing classical and quantum processing and data.

Classical Data and Quantum Processing (CQ): CQ explores the potential of harnessing quantum computing to process classical data [Preskill, 2018] and is the intersection studied in this thesis. By encoding classical data into quantum states and applying quantum algorithms, we can potentially achieve significant speedup over classical algorithms for specific problems [Shor, 1999, Grover,

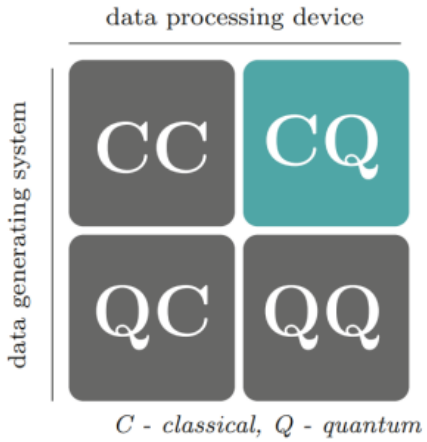


Figure 3.1: The four ways of combining classical and quantum computing concerning data and processing [Schuld and Petruccione, 2018]

1996]. Promising applications include optimization, factoring, and quantum machine learning [Farhi et al., 2014a, Wiebe et al., 2012]. This approach, however, comes with challenges, such as the need for efficient classical-to-quantum data encoding, error correction, and the current limitations of quantum hardware [Nielsen and Chuang, 2000, Terhal, 2015].

Quantum Data and Classical Processing (QC): QC investigates the feasibility of processing quantum data using classical algorithms [Aaronson and Arkhipov, 2013]. This approach is less common and typically constrained by the exponential growth of classical resources required to process quantum data [Nielsen and Chuang, 2000]. One application area of QC is quantum state tomography, where classical algorithms estimate the state of a quantum system [Paris and Rehacek, 2004]. However, the scalability of this approach remains an open challenge [Cramer and Plenio, 2010].

Quantum Data and Quantum Processing (QQ): QQ represents the form of quantum computing, where quantum data is processed using quantum algorithms. This paradigm is expected to provide exponential speedup for specific problems like quantum simulations and cryptography [Lloyd, 1996, Shor, 1994]. While promising, the development of practical QQ applications needs to be improved by technological limitations. These include high error rates in quantum systems,

limited qubit coherence times that affect the stability of quantum information, and the availability of large-scale quantum hardware [Preskill, 2018, Devoret and Schoelkopf, 2013]. These challenges need to be addressed before the full potential of QQ can be realized in real-world applications.

Despite these obstacles, quantum machine learning holds great promise in advancing the field of machine learning. It can potentially revolutionize various domains, such as drug discovery, finance, and cryptography. By leveraging the unique properties of quantum systems, it is possible to solve complex problems more efficiently than classical approaches, opening up new avenues for research and innovation. As quantum hardware continues to evolve and improve, we can anticipate even more significant breakthroughs and applications in the near future.

3.3 Ethical Considerations

As with any scientific research, ethical considerations must be considered when working with quantum computing and machine learning. While our primary focus in this thesis is on developing novel algorithms and techniques, we must also consider the potential implications of our work in the broader context. In particular, developing more powerful machine learning algorithms may raise concerns about their application in surveillance, automated decision-making, and the potential misuse of technology.

One concern is potentially exacerbating existing biases in classical machine learning systems. Quantum machine learning algorithms could amplify these biases, leading to unfair or discriminatory outcomes. It is crucial to ensure that the data used to train quantum machine learning models is representative and unbiased and that researchers remain vigilant about identifying and addressing any bias that may arise during the development and deployment of these systems.

Another ethical consideration is the impact of quantum computing on privacy and data security. Quantum algorithms, such as Shor's algorithm, could undermine widely-used encryption methods, putting sensitive data at risk. Researchers and developers should be cognizant of these risks and work on developing

post-quantum cryptography to protect data in a world where quantum computers are prevalent.

Additionally, the rapid advancement of quantum machine learning may have implications for the job market and societal dynamics. As more efficient algorithms are developed, these technologies can displace human workers in specific industries or exacerbate income inequality. Researchers, policymakers, and stakeholders need to engage in dialogue about the equitable distribution of the benefits that quantum computing and machine learning can provide and the development of strategies to mitigate potential negative consequences.

To address these concerns, we will ensure that our research adheres to ethical guidelines and best practices and actively discuss the responsible use and development of quantum computing and machine learning technologies. It is our responsibility as researchers to not only contribute to the scientific advancements of these fields but also to consider their broader implications and promote their ethical development and application. By fostering an environment of open communication and collaboration among researchers, policymakers, and the public, we can work together to ensure that quantum computing and machine learning technologies are developed and employed to benefit society.

3.4 Thesis Structure

The paper is structured to provide a holistic understanding of this complex domain, starting with foundational elements before transitioning into the practical application of quantum concepts within machine learning methodologies.

The journey begins with a comprehensive dive into theoretical foundations, exploring the nuances of Deep Q-Learning and Quantum Mechanics. Crucial elements such as Supervised Learning, Neural Networks, Optimizers, and DQL are explained in detail. The mysterious realm of quantum mechanics is unpacked, highlighting wave-particle duality, superposition, Schrödinger's equation, the Heisenberg uncertainty principle, and entanglement. It then expands these discussions into the realm of Quantum Computing and Quantum Deep

Q-Learning, demonstrating how these core quantum principles find their application in advanced computing and machine learning.

The methodology chapter demarcates the tools and environments utilized in this exploration, focusing on Open AI Gym, and sets the stage for the subsequent experiments. The implementation chapter elucidates the mechanics behind both Classical Deep Q-Learning and Quantum Deep Q-Learning implementations, emphasizing the reproducibility of the experiments and the process for reporting results.

The thesis then shifts gears towards a hands-on exploration of the concepts through a series of six key experiments, documented in the ”Experiments” chapter. Each experiment unravels a distinct facet of the central hypothesis, starting with applying classical Deep Q-Learning to Cart Pole and Lunar Lander scenarios. These foundational experiments set the benchmark for the next set, which applies quantum DQL to the same scenarios. The final set of experiments introduces trainable entanglement into the quantum DQL algorithm, examining how this uniquely quantum property might enhance machine learning algorithms when applied to the exact scenarios.

The Results and Discussion chapter takes on the critical role of interpreting the experimental data. It provides a detailed analysis of the raw results, followed by an in-depth interpretation within the context of the overarching hypothesis. The chapter critically examines performance metrics such as learning rate, number of network weights, number of layers, and the role of entanglement, drawing parallels between classical and quantum scenarios. The final segment of this chapter delves into a multi-hyperparameter sensitivity analysis, offering insights into the interaction between various hyperparameters and their cumulative impact on the efficacy of the DQL algorithm.

4 Theoretical Background

This chapter provides the theoretical background necessary to understand the concepts discussed in this thesis. The chapter is organized into four main sections: Deep Q-Learning, Quantum Mechanics, Quantum Computing, Quantum Deep Q-Learning, and Related Work. Each section delves into specific concepts, techniques, and methods essential for a comprehensive understanding of the research presented in the subsequent chapters.

4.1 Deep Q-Learning

This section explores Deep Q-Learning, a powerful combination of deep neural networks and reinforcement learning. We cover the basics of Supervised Learning, which focuses on learning from labeled data, and introduce Neural Networks, the building blocks of deep learning architectures. We discuss various Optimizers used to update network parameters during training. Then, we delve into Reinforcement Learning, a paradigm where agents learn through trial and error, and Q-Learning, a popular reinforcement learning technique. Finally, we illustrate how deep learning and Q-Learning merge to form Deep Q-Learning, enabling the learning of complex tasks and strategies. As we progress through each section, keep in mind the underlying connections and how each concept builds on the ones before it, ultimately culminating in the comprehensive understanding of Deep Q-Learning.

4.1.1 Supervised Learning

Supervised learning is a machine learning technique where the algorithm learns from labeled data through a process called training. The training process involves making predictions, measuring the prediction's deviation from the actual value using a loss function, and adjusting the model based on the magnitude of the deviation. This means that for a given input x, there exists a desired output y. The goal is to build a model that can map x to y and make accurate predictions on new, unseen data by generalizing to underlying patterns and mechanisms that generate

the data.

In supervised learning, the balance between model complexity and generalization is crucial. The concepts of bias, variance, overfitting, and underfitting are essential to understand this balance. Bias refers to the error introduced by approximating the actual underlying function with a simpler model. High bias can lead to underfitting, where the model fails to capture the complexity of the problem and performs poorly both during training and evaluation. Conversely, variance is the error resulting from the model's sensitivity to small fluctuations during training. High variance can cause overfitting, in which the model performs well during training but poorly during evaluation due to learning noise in the data rather than the actual underlying patterns.

Several techniques can be employed to address these issues, such as regularization and early stopping. Regularization adds a penalty term to the model's loss function, discouraging overly complex models and helping to reduce overfitting. Early stopping involves monitoring the model's performance over time on a validation dataset and halting the training process once the performance during evaluation starts to degrade, preventing the model from overfitting the training data.

4.1.2 Neural Networks

Neural networks are a machine learning model inspired by the structure and function of the biological brain. They consist of artificial neurons, called nodes, connected in a structure called a network. This network is structured in an input layer, an output layer, and a number of layers between them called hidden layers. Figure 4.1 shows an example of a neural network with four inputs, two hidden layers with four nodes, and an output layer with two nodes. Information flows through the network sequentially, passing through each layer. The nodes process and transform data using activation functions, introducing non-linearities into the network. The output of one layer is used as the input to the next layer, with the final layer producing the model's output. When information only flows in one direction in a neural network, it is classified as a feed-forward neural network.

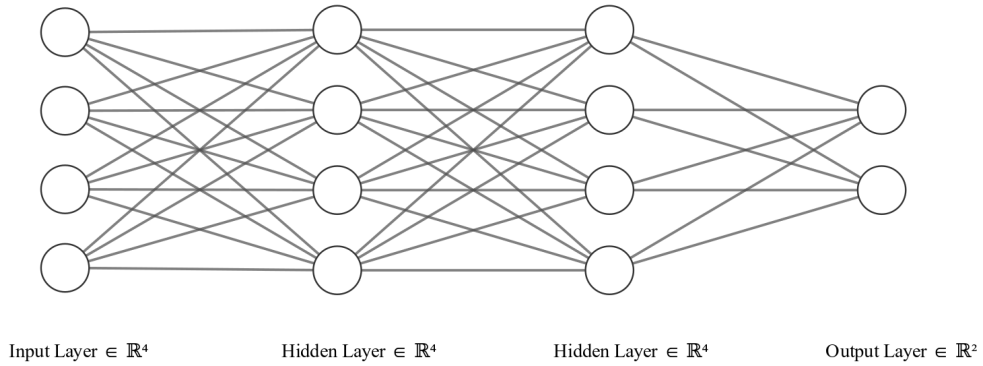


Figure 4.1: Architecture of a neural network consisting of input, hidden, and output layers.

One of the benefits of neural networks is their ability to learn complex, non-linear relationships in data. This has led to significant advances in fields such as computer vision [Krizhevsky et al., 2012], natural language processing [Mikolov et al., 2013], and speech recognition [Hinton et al., 2012].

In a dense neural network, there is a connection between every node in one layer and every node in the next. A connection is called a weight w_{ij} and consists of a trainable number multiplied by the node's output. This means that the input of a node in a dense layer is the weighted sum of the output of all previous nodes. This can be expressed as

$$\mathrm{in}_j = \sum_{i=1}^{n} w_{ij}\mathrm{out}_i + b_j, \tag{1}$$

where out_i is the output of the i-th node in the previous layer, b_j is the bias of the j-th node, and n is the number of nodes in the previous layer.

The process of training a neural network consists of adjusting the output of the network to better align with the desired output. This involves feeding the training data through the network and adjusting the weights of the connections between

neurons to minimize the difference between the predicted and actual outputs. This is done by minimizing a loss function L that measures the difference between the predicted output $\hat{y}$ and the actual output y. The choice of loss function depends on the problem at hand. For example, the mean squared error (MSE) loss function is commonly used for regression problems is represented as

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{2}$$

where n is the number of training samples.

The weights of a neural network are updated by an optimization algorithm, such as stochastic gradient descent [Rumelhart et al., 1986] or Adam [Kingma and Ba, 2015]. These algorithms iteratively adjust the weights toward the steepest descent of the loss function, intending to minimize the network's prediction error. This process continues until the loss function converges or reaches a stopping criterion. However, given the high dimensionality and non-convex nature of the loss landscape in deep neural networks, these optimization algorithms are susceptible to getting trapped in local optima, which can lead to suboptimal performance and limit the generalization capabilities of the model [Choromanska et al., 2015].

Neural networks can also be prone to overfitting, where the model fits too closely to the training data and does not generalize well to new data. Regularization techniques, such as dropout [Srivastava et al., 2014], can be used to mitigate this issue. Dropout works by stochastically nullifying the output of random nodes. This ensures that no single connection becomes too strong and that information flow is distributed throughout the network. On the opposite side, there is the problem of underfitting. Strategies for addressing underfitting include increasing model complexity, using more expressive architectures, or adding additional features to the input data [Goodfellow et al., 2016].

One of the main limitations of supervised learning is its reliance on a large amount of labeled data. Obtaining high-quality labeled data can be both time-consuming and expensive, as it often requires manual effort by domain experts. In many

real-world applications, labeled data may be scarce or unavailable, limiting the applicability of supervised learning techniques [Zhu and Goldberg, 2009]. Furthermore, supervised learning models are only as good as the quality of the training data, which means that if the training data is biased or noisy, the model's performance may suffer. To address these challenges, researchers have explored alternative approaches such as unsupervised, semi-supervised, and transfer learning, which leverage unlabeled data, partial supervision, and pre-trained models to improve learning efficiency and reduce the reliance on large, labeled datasets [Pan and Yang, 2009].

Despite their remarkable success in various domains, neural networks have inherent limitations. First, they can be computationally expensive to train, mainly when dealing with large-scale datasets and deep architectures. This may necessitate the use of specialized hardware, such as Graphical Procession Units (GPUs) or Tensor Processing Units (TPUs) can be costly and may not be readily accessible to all researchers and practitioners [LeCun et al., 2015]. Second, neural networks are often characterized as "black boxes" due to their complex internal structure and lack of interpretability. This can make understanding how the model makes predictions challenging and can be problematic in domains where explainability and trust are crucial, such as healthcare and finance [Guidotti et al., 2018]. Lastly, neural networks are vulnerable to adversarial examples, carefully crafted inputs that fool the model into making incorrect predictions. This vulnerability raises concerns about the robustness and security of neural network-based systems in safety-critical applications [Szegedy et al., 2013].

ReLU activation is a widely used activation function in neural networks. It stands for Rectified Linear Unit, defined as $max(0, x)$, where x is the input to the function. According to a paper by Nair and Hinton [2010], the ReLU activation function has several advantages over other activation functions, such as sigmoid and tanh, including faster convergence during training and reduced likelihood of vanishing gradients. These properties make ReLU a popular choice for deep learning models.

Lastly, neural networks are vulnerable to adversarial examples, carefully crafted

inputs that fool the model into making incorrect predictions. This vulnerability raises concerns about the robustness and security of neural network-based systems in safety-critical applications [Szegedy et al., 2013]. The development of robust defenses against adversarial examples remains an active area of research in deep learning, with several proposed techniques such as adversarial training and defensive distillation [Madry et al., 2017, Papernot et al., 2016].

4.1.3 Optimizers

Optimizers are critical to training deep learning models, including Deep Q-Learning. Optimizers are algorithms that adjust the neural network weights during training to minimize the loss function. One popular optimizer is stochastic gradient descent (SGD), which adjusts the weights by taking small steps toward the negative gradient of the loss function [Rumelhart et al., 1986]. Thanks to its simplicity, SGD can converge slowly and get stuck in local minima [LeCun et al., 2012].

Adam optimizer is another popular algorithm used in deep learning models for gradient-based optimization, including DQL. It is an extension of the SGD algorithm and is known for its efficiency in minimizing the loss function during training. According to the original paper by Kingma and Ba [2015], the Adam optimizer updates the learning rate adaptively for each parameter based on the mean and variance of the gradients. This makes it less sensitive to the initial learning rate and the choice of hyperparameters. The Adam optimizer can also solve non-convex optimization problems common for deep learning models.

Other commonly used optimizers include Adagrad [Duchi et al., 2011], RMSprop [Tieleman and Hinton, 2012], and Adadelta [Zeiler, 2012]. Adagrad adjusts the learning rate of each weight based on the historical gradients of that weight. RMSprop and Adadelta are extensions of Adagrad that attempt to solve its diminishing learning rate problem. Each optimizer has its strengths and weaknesses, and the choice of optimizer depends on the specific problem and the desired tradeoffs in terms of convergence speed, robustness to local minima, and sensitivity to hyperparameters.

Backpropagation is an algorithm used to calculate the gradients of the loss function concerning the parameters of a neural network. The algorithm works by propagating the error or loss from the output layer back to the network's input layer, and it is based on the chain rule of calculus. According to a classic paper by Rumelhart et al. [1986], backpropagation is a crucial component of training deep neural networks.

The backpropagation algorithm consists of two main steps: the forward pass and the backward pass. The input data is passed through the network to compute the output in the forward pass, and the loss function is evaluated. In the backward pass, the gradients of the loss function concerning each weight and bias are computed. These gradients are then used to update the network parameters in the direction of the steepest descent of the loss function.

By iteratively updating the neural network parameters using the backpropagation algorithm, the network learns to minimize the loss function and make more accurate predictions. This optimization continues until a stopping criterion is reached, such as a maximum number of training steps, early stopping, or when the loss function converges to a minimum value. Backpropagation is a cornerstone of deep learning and is critical to the success of deep learning models.

4.1.4 Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning that deals with decision-making in an environment. To motivate desired behavior, the agent generally receives a positive reward for performing desirable behavior while receiving a negative reward for undesirable behavior. In RL, an agent interacts with an environment to learn a policy, i.e., a strategy, that maximizes a long-term reward. The goal is to learn a policy that selects actions that maximize the cumulative reward over time.

This process of observing the environment, performing an action, and receiving feedback is called a Markov Decision Process (MDP). Interacting with an MDP is continued until a stopping criterion terminates the process. Termination occurs when the problem is solved, too much time has passed, or the environment is in

an unrecoverable state. An episode is a sequence of interactions with an MDP from the start until termination. The RL process can be modeled as a Markov Decision Process (MDP), which is defined by a set of states S, a set of actions A, the new state s' which occurs when performing action A in state S, and a reward function $R(s, a, s')$ that specifies the reward received when transitioning from state s to state s' by taking action a. The goal of the agent is to learn a policy $\pi(a|s)$ that maps state to action to maximize the expected cumulative reward, written as

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{3}$$

where G_t is the cumulative reward at time t, R_{t+k+1}, is the reward received at time $t + k + 1$ and γ is a discount factor that trades off long-term rewards with immediate rewards.

There are two main approaches to RL: value-based and policy-based methods.

Value-based RL methods learn a value function that estimates the expected cumulative reward of an action in a given state. The value function can be defined as

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s], \tag{4}$$

where $V_\pi(s)$ is the value of state s under policy π, and $\mathbb{E}_\pi$ denotes the expected value over all possible trajectories starting from state s under policy π. The optimal value function $V^*(s)$ is the maximum expected cumulative reward that can be achieved from state s, written as

$$V^*(s) = \max_\pi V_\pi(s). \tag{5}$$

Policy-based RL methods learn a policy directly without explicitly computing a value function. The policy is parameterized by a set of parameters θ, and the goal is to find the optimal set of parameters θ^* that maximizes the expected cumulative

reward, written as

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{\pi_\theta}[G_t]. \tag{6}$$

Policy Gradient methods are a popular family of policy-based RL algorithms that use gradient ascent to update the policy parameters. The gradient of the expected cumulative reward concerning the policy parameters can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[G_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right], \tag{7}$$

where $J(\theta)$ is the performance objective, typically the expected cumulative reward, and ∇_θ denotes the gradient concerning the policy parameters θ. By computing the gradient of $J(\theta)$, Policy Gradient methods can update the policy parameters θ toward a higher expected cumulative reward while avoiding the need for an explicit value function.

4.1.5 Q-Learning

Q-Learning is a model-free reinforcement learning algorithm that learns an optimal policy by estimating state-action pairs' values. It operates by iteratively updating the Q-values of state-action pairs using the Bellman equation, given by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)], \tag{8}$$

where s_t is the current state, a_t is the current action, r_{t+1} is the reward received after taking action a_t in state s_t and transitioning to state s_{t+1}, γ is the discount factor, and α is the learning rate. The Bellman equation expresses the Q-value of a state-action pair as the sum of the immediate reward and the discounted value of the best action that can be taken from the next state.

Q-Learning has to balance learning new information about the environment and exploiting the knowledge it has already acquired. This concept is called the

exploration-exploitation-tradeoff. A standard method for achieving this balance is ϵ-greedy exploration, where with probability ϵ, the agent chooses a random action, and with probability $1 - \epsilon$, it chooses the action with the highest Q-value.

One of the drawbacks of Q-Learning is that it requires a table to store the Q-values of all state-action pairs, which can be infeasible for large state spaces. To address this issue, researchers have developed variants of Q-Learning that use function approximation to estimate the Q-values, called deep Q-learning.

4.1.6 Deep Q-Learning

Deep Q-Learning is a sub-field in Reinforcement Learning that utilizes the function approximation properties of neural networks to approximate the Q-function [Mnih et al., 2015]. A Q-function attempts to approximate the expected return by performing a specific action in a specific state. In RL, an agent interacts in an environment through observations and actions while receiving rewards. The agent decides on actions based on experience by estimating the expected reward gained by performing a specific action in a specific state and following the same policy.

In DQL, the neural network input shape matches the observation space. Also, there is one output node linked to each possible action. The output value in each node attempts to approximate the expected reward by performing that action and following the policy until termination, meaning that the action with the highest reward is selected every time. As neural networks are universal function approximators [Bishop, 1994], this technique performs better than previous techniques, such as traditional Q-learning. A DQL agent learns patterns from similar situations instead of treating every situation as unique. To focus on situations that occur in the near future, a discount factor γ is used, multiplied by the reward, decreasing with each additional step into the future.

When training a DQL agent, experience is stored in a replay buffer. Experience is the current and previous observation, previous action, and reward. These values are used to create training data for the neural network. This technique is called *replay experience* and was introduced by Mnih et al. [2013]. The traditional update function for DQL is written as

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{a'} Q^*(s',a')|s,a], \tag{9}$$

where Q^* is the neural network, observing a set of states s and actions a. r is the current reward. The functions attempt to maximize Q^* by selecting the best activity a.

Another important RL subject is the *exploration-exploitation-tradeoff*. This refers to the tradeoff between exploiting gained knowledge and exploring and possibly discovering better strategies. This is usually implemented by having an exploration rate ϵ, which determines how often an agent performs an exploratory action instead of an exploitative one. It is common for the exploration rate to decrease over time as more experience is gained and the agent's strategy improves.

4.2 Quantum Mechanics

Quantum Mechanics (QM) is a branch of physics that describes the behavior of matter and energy at the quantum level. Quantum mechanics describes the behavior of tiny objects, such as atoms, molecules, and subatomic particles. In contrast to classical mechanics, quantum mechanics introduces the concept of wave-particle duality, which means that particles can exhibit both particle-like and wave-like behavior [Cohen-Tannoudji et al., 1977].

4.2.1 Wave-Particle Duality

Wave-particle duality is a fundamental concept in quantum mechanics and is often illustrated using the double-slit experiment. In this experiment, a beam of particles, such as electrons or photons, is fired at a barrier containing two small slits. Interference patterns are observed on a screen behind the barrier as if the particles behaved like waves. However, when the experiment is repeated with detectors placed at the slits to measure which slit each particle passes through, the interference pattern disappears, and the particles behave like individual particles.

Wave-particle duality is explained by the wave-like nature of matter, which is described by the wave function in quantum mechanics. The wave function contains all the information about a particle's possible states, such as its position and momentum, and Schrödinger's equation governs its behavior. The wave function collapses when a measurement is made, causing the particle to behave like a classical particle with a definite position and momentum.

The concept of wave-particle duality is central to many quantum phenomena, such as quantum entanglement, superposition, and tunneling. Understanding the behavior of matter and energy at the quantum level is crucial for developing technologies such as quantum computers and quantum cryptography, which have the potential to revolutionize computing and communication.

4.2.2 Superposition

Superposition is a fundamental concept in quantum mechanics that refers to the ability of a quantum system to exist simultaneously in multiple states, with each state having a specific probability amplitude. This property enables quantum systems to represent and process vast amounts of information in parallel, forming the foundation of quantum computers' computational power [Nielsen and Chuang, 2002].

When a quantum system is in a superposition state, and a measurement is performed, the outcome is determined probabilistically. The system collapses into one of the possible states, with the probability for each state given by the square of the magnitude of its amplitude. This probabilistic nature of quantum measurements is described by Born's Rule [Born, 1926b], which is written as:

$$P(x) = |\langle x|\psi\rangle|^2, \tag{10}$$

where $P(x)$ is the probability of finding the quantum system in state x upon measurement. The state x is one of the possible outcomes of the measurement. The symbol ψ represents the overall state of the quantum system before measurement. The angle brackets $\langle x|\psi\rangle$ denotes the inner product between the

bra vector corresponding to state x and the state vector ψ. The square of the magnitude of this inner product yields the probability of the system collapsing into state x upon measurement.

4.2.3 Schrödinger Equation

In quantum mechanics, the state of a system is described by a wave function, typically denoted by the Greek letter psi (ψ). This wave function contains all the information about the system's properties and is a solution to the Schrödinger equation [Sakurai and Napolitano, 2014]. The Schrödinger equation is a fundamental equation in quantum mechanics, which predicts how the quantum state, and therefore the wave function, will evolve over time. The Schrödinger equation is written as

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t), \tag{11}$$

where $\hbar$ represents the reduced Planck's constant, t represents time, $\mathbf{r}$ represents the spatial coordinates, $\Psi(\mathbf{r},t)$ is the wave function, and $\hat{H}$ is the Hamiltonian operator, which represents the total energy of the quantum system.

However, the wave function alone does not directly provide any observable system properties. Instead, it provides the probability distribution for these observables. For instance, the square of the absolute value of the wave function, $|\psi(\mathbf{r},t)|^2$, gives the probability density of finding a particle in a tiny volume around the position $\mathbf{r}$ at time t [Griffiths, 2018]. Other observables like momentum and energy are obtained through specific quantum mechanical operators acting on the wave function.

4.2.4 The Heisenberg uncertainty principle

The Heisenberg uncertainty principle is a crucial concept in quantum mechanics that imposes fundamental limitations on the simultaneous knowledge of specific pairs of observables, such as position x and momentum p. This principle arises from the non-commutative nature of quantum mechanical operators corresponding

to these observables, leading to an inherent uncertainty in their measurements [Heisenberg, 1927].

Mathematically, the uncertainty principle can be formulated as

$$\Delta x \Delta p \geq \frac{\hbar}{2}, \tag{12}$$

where Δx represents the uncertainty in position, Δp denotes the uncertainty in momentum, and $\hbar$ is the reduced Planck constant. The inequality indicates that the product of the uncertainties in position and momentum must be greater than or equal to a constant value, which is half of the reduced Planck constant. As a result, if we obtain precise information about the position of a quantum particle, its momentum becomes more uncertain, and vice versa.

The uncertainty principle is not limited to position and momentum; it also applies to other pairs of non-commuting observables, such as energy E and time t, which are subject to a similar constraint, denoted as

$$\Delta E \Delta t \geq \frac{\hbar}{2}. \tag{13}$$

The Heisenberg uncertainty principle has profound implications for understanding quantum systems' nature and behavior. It highlights the intrinsic limitations in the simultaneous measurement of specific physical properties, emphasizing the probabilistic nature of quantum mechanics.

4.2.5 Entanglement

Another important principle of QM is entanglement, where two or more quantum systems become correlated so that one system's properties depend on the other's, even if they are separated by large distances [Einstein et al., 1935].

The application of QM has led to many significant technological advancements, including transistors, lasers, and medical imaging devices [Wienke, 2007]. One of the most promising applications of QM is in quantum computing, which has the

potential to revolutionize computing by providing exponential speedup over classical computing for specific problems [Nielsen and Chuang, 2002].

In quantum computing, quantum mechanics encodes and manipulates information like quantum bits or qubits. By exploiting the properties of entanglement and superposition, quantum computers can perform certain computations much faster than classical computers. However, the development of practical quantum computers is still in its infancy, and many challenges must be overcome before they become viable technology [Devoret and Schoelkopf, 2013].

Overall, quantum mechanics is a fascinating and complex field that has revolutionized our understanding of the nature of reality. Its applications range from designing new materials to developing revolutionary computing technologies, making it an exciting area of research with vast potential for future advancements.

4.3 Quantum Computing

Quantum computing (QC) is a field that aims to exploit the computational advantages of quantum systems over classical computing [Nielsen and Chuang, 2002]. Simulating quantum systems on classical computers is a computationally demanding task [Feynman, 1982]. This indicates some computational complexity in quantum effects, such as entanglement, that we hope to exploit as a resource with QC. Quantum computing has the potential to revolutionize a range of fields, including cryptography, optimization, and chemistry. For example, quantum computers can solve specific problems exponentially faster than classical computers, such as factoring large numbers [Shor, 1994] and searching unsorted databases [Grover, 1996].

This chapter provides an overview of the fundamental principles of quantum computing. Starting with the quantum bit called the *qubit*, we explore the basic unit of quantum information. We then delve into quantum gates, which manipulate qubits, and quantum circuits, which are sequences of these gates. We discuss the unique process of quantum measurement before concluding with an exploration of quantum entanglement, a phenomenon that sets quantum computing apart from

its classical counterpart.

4.3.1 The Qubit

In quantum computing, qubits are the fundamental information storage and processing elements. A distinguishing feature of qubits is their ability to exist in a superposition of states, demonstrated by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{14}$$

where the coefficients α and β are complex numbers representing the probability amplitudes for the qubit to be in the $|0\rangle$ and $|1\rangle$ states, respectively. The constraint $|\alpha|^2 + |\beta|^2 = 1$ ensures that the total probability remains normalized under the fundamental principles of quantum mechanics.

For a system comprising n qubits, the general superposition can be expressed as

$$|\psi\rangle = \sum_{v_1=0}^{1} \sum_{v_2=0}^{1} ... \sum_{v_n=0}^{1} \alpha_{v_1,v_2,\cdots,v_n} |v_1 v_2 \cdots v_n\rangle. \tag{15}$$

This equation implies that 2^n unique bit strings of the form $v_1 v_2 \cdots v_n$ are required for a general superposition, necessitating 2^n probability amplitudes $\alpha_{v_1,v_2,\cdots,v_n}$. The exponential increase in information emphasizes the potential of quantum computing for representing and manipulating large data sets using a limited number of qubits.

Qubits are often modeled using the quantum mechanical property of spin, particularly the spin of an electron or a photon. Spin can be visualized as an intrinsic form of angular momentum carried by quantum particles, where the *spin up* and *spin down* states correspond to the two states of a qubit, often denoted as $|0\rangle$ and $|1\rangle$, respectively.

The qubit's quantum state, $|\psi\rangle$, is a linear combination of the basis states $|0\rangle$ and $|1\rangle$. Upon measuring a qubit in this superposition state, the outcome is determined

probabilistically based on the square of the magnitudes of the probability amplitudes. Specifically, the probability of measuring the qubit in state $|0\rangle$ is given by $P(|0\rangle) = |\alpha|^2$, while the probability of measuring the qubit in state $|1\rangle$ is given by $P(|1\rangle) = |\beta|^2$. This probabilistic nature of quantum measurements underpins various quantum algorithms and is a core aspect of quantum mechanics.

In a quantum circuit, qubits can be thought of as two-level systems, analogous to classical bits, but with the added feature of being able to exist in superposition states. At the start of quantum computation, all qubits are set to the spin-down state, which corresponds to the zero states $|0 \cdots 0\rangle$ in the computational basis. The computational basis consists of all possible states that can be represented by classical bit strings, such as $|000\rangle$, $|001\rangle$, ..., $|110\rangle$, and $|111\rangle$. The first step of a quantum circuit is to encode information into the qubits. This can be done by, for example, parameterized quantum circuits (PQC). In PQC, input is encoded into the quantum system rotations on the qubits. Rotation refers to the transformation of a qubit's state by applying a quantum gate, which can be visualized as a rotation of the state vector around an axis on the Bloch sphere, changing the probabilities of measuring different outcomes. The rotation can be encoded using quantum gates, which rotate the qubit's state based on a parameter [Benedetti et al., 2019].

Once the desired input is encoded into the quantum system, the desired logic can be applied. This is performed with quantum gates, unitary operations that transform a qubit state into another. Finally, the output is measured, and the wave function of all qubits collapses into either the spin-up or spin-down state.

In quantum computing, understanding and visualizing qubits' behavior is essential to comprehending quantum systems. To facilitate this understanding, the Bloch sphere is employed to represent qubits, the fundamental quantum bits. The Bloch Sphere is a three-dimensional representation used to visualize the state of a single qubit. It is a unit sphere centered at the origin in a three-dimensional space with orthogonal axes typically labeled X, Y, and Z. The state of a qubit is described as a point inside or on the surface of the sphere.

As depicted in Figure 4.2, the Bloch Sphere incorporates a qubit's entire states: the sphere's north pole represents the $|0\rangle$ state, and the south pole symbolizes the

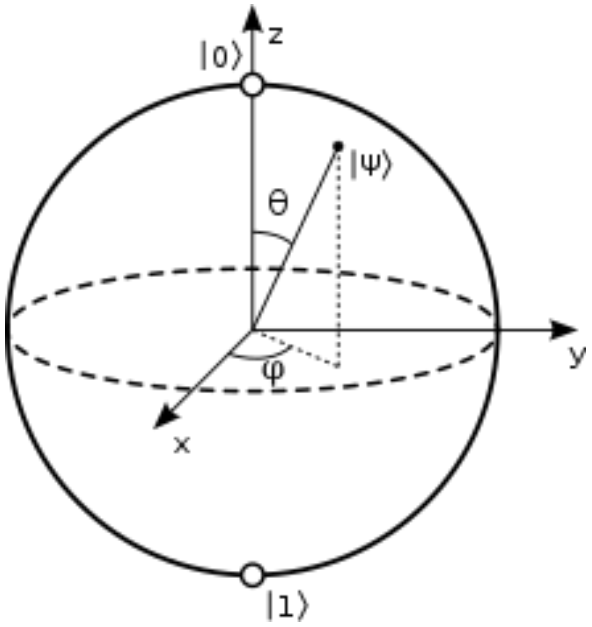


Figure 4.2: A representation of a qubit in the Bloch Sphere. [Wikipedia, 2023]

$|1\rangle$ state. The sphere's surface represents the states where the qubit is in a superposition of the $|0\rangle$ and $|1\rangle$ states.

4.3.2 Quantum Gates

Quantum gates are an essential component of quantum circuits, and they are used to manipulate qubits in various ways. All quantum gates can be represented as a matrix acting on a vector representing the qubits' state. A few commonly used quantum gates are the Pauli gates (X, Y, and Z), which rotate the qubit along each axis with respect to a parameter θ. The Pauli gates can be further broken down into Pauli rotations, R_x, R_y, and R_z, which rotate the qubit about the X, Y, and Z axis, respectively. These gates are useful for encoding information into individual qubits. Hadamard is another important quantum gate that creates or collapses existing superposition. When applied to a qubit initially in the state $|0\rangle$ (spin up) or $|1\rangle$ (spin down), the Hadamard gate transforms the qubit into a superposition of both states. Importantly, this operation is reversible, meaning applying the Hadamard gate again will return the qubit to its original state, even though the superposition state is indistinguishable, regardless of starting conditions. This

ability to create and manipulate superpositions, combined with the property of reversibility, gives rise to unique properties useful in quantum cryptography. Both Hadamard and the Pauli gates are used to transform individual qubits. CNOT is a type of quantum gate that affects multiple qubits. CNOT works by conditionally flipping the spin state of one qubit based on the state of one or more other qubits [Chuang and Nielsen, 2000].

Here are some examples of quantum gates, their effects on qubits, and their matrix representations. For reference, the state of one qubit can be represented in vector notation as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{16}$$

while a two-qubit system can be represented as

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{17}$$

Pauli-X Gate:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{18}$$

$$\begin{aligned} X|0\rangle &= |1\rangle \\ X|1\rangle &= |0\rangle \end{aligned} \tag{19}$$

The Pauli-X gate is commonly used to flip the state of a qubit.

CNOT Gate:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{20}$$

$$\text{CNOT}|c\rangle|t\rangle = |c\rangle|t \oplus c\rangle, \tag{21}$$

The CNOT flips the second bit if the first bit is measured as one.

$$\begin{aligned} \text{CNOT}|0\rangle|0\rangle &= |0\rangle|0 \oplus 0\rangle = |0\rangle|0\rangle \\ \text{CNOT}|0\rangle|1\rangle &= |0\rangle|1 \oplus 0\rangle = |0\rangle|1\rangle \\ \text{CNOT}|1\rangle|0\rangle &= |1\rangle|0 \oplus 1\rangle = |1\rangle|1\rangle \\ \text{CNOT}|1\rangle|1\rangle &= |1\rangle|1 \oplus 1\rangle = |1\rangle|0\rangle \end{aligned}$$

Hadamard Gate:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{22}$$

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle \ + |1\rangle) \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle \ - |1\rangle) \\ HH|0\rangle &= \frac{1}{\sqrt{2}}H(|0\rangle \ + |1\rangle) = |0\rangle \end{aligned} \tag{23}$$

The Hadamard gate creates an equal superposition of the basis states. When applied twice, it returns the qubit to its initial state.

Pauli Rotation Gates:

$$R_x(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{24}$$

$$R_x(\theta)|0\rangle = \cos(\theta/2)|0\rangle - i\sin(\theta/2)|1\rangle \tag{25}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{26}$$

$$R_y(\theta)|0\rangle = \cos(\theta/2)|0\rangle - \sin(\theta/2)|1\rangle \tag{27}$$

The $R_x(\theta)$ and $R_y(\theta)$ gates are Pauli rotations that rotate the qubit around the X and Y axes, respectively, with respect to a parameter θ. These gates are useful for encoding information into a qubit, allowing for arbitrary rotation of the qubit's state.

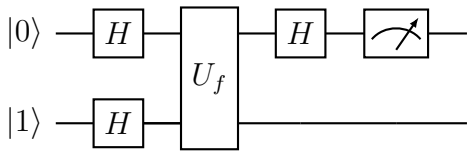
In quantum mechanics, the Hilbert space is a complex vector space that accommodates the quantum states of a system. It enables the representation and manipulation of quantum states using linear algebra. In quantum computing, the Hilbert space expands exponentially with the number of qubits. For example, with n qubits, the size of the Hilbert space is 2^n. Different entanglement schemes and quantum gates provide access to varying portions of the Hilbert space, affecting the expressibility and computational power of the quantum circuits.

4.3.3 Quantum Circuits

Quantum circuits provide a graphical representation of the sequence of quantum operations applied to qubits. Each wire in a quantum circuit represents a qubit, and the quantum gates are depicted as boxes along the wire. The horizontal direction represents the time progression of the quantum operations, with the

initial state on the left and the final state on the right. The power of quantum circuits lies in their ability to manipulate and entangle qubits, resulting in complex quantum states that can be harnessed to perform complex computations.

As an example, consider the following quantum circuit that implements the Deutsch-Josza algorithm [Deutsch and Jozsa, 1992], which is designed to solve a specific problem more efficiently than classical computers:



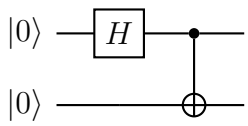
(28)

In this circuit, the first qubit represents the input register, and the second qubit represents the output register. The Hadamard gates are applied to both qubits to create an equal superposition of all possible input states. The oracle gate U_f encodes the function $f(x)$ and is applied to both qubits. Finally, the Hadamard gate is applied again to the input register, and the measurement of the first qubit reveals whether the function is constant or balanced.

A Bell State is a special type of entangled state of two qubits with many applications in quantum communication and quantum computing. The Bell State can be generated by applying a Hadamard gate on one qubit and then applying a CNOT gate with that qubit as the control and the other as the target. The resulting Bell State is:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) \tag{29}$$

The bell state can be prepared with the following quantum circuit:



(30)

In this circuit, the Hadamard gate creates a superposition on the first qubit, and the CNOT gate entangles the two qubits, generating the Bell state. These entangled states are crucial in various quantum algorithms and quantum communication protocols.

The Bell State is a maximally entangled state that cannot be written as a product of two qubits. It is expressed as $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, where the two qubits are in a state of superposition. This state is useful for quantum teleportation and quantum key distribution tasks. It is important to note that the Bell State cannot be written as a product of two individual qubits in any way, i.e., it cannot be expressed as $|\psi\rangle \otimes |\phi\rangle$.

Due to the nature of quantum systems, a quantum circuit produces widely different results each time it runs. This uncertainty is caused by both outside noises disturbing the system and the fact that a qubit in a superposition will not be measured identically each time the experiment is performed. Therefore, a quantum circuit is executed many times, and the distribution of each unique measurement is considered the output of the system, not the result of any particular execution [Preskill, 2018].

Overall, quantum computing is an exciting and rapidly developing field with a wide range of potential applications in areas such as chemistry, optimization, and artificial intelligence [Farhi et al., 2014a].

4.3.4 Measurement in Quantum Systems

A crucial aspect of understanding quantum systems is the concept of measurement and its connection to the probability of observing a particular outcome. In quantum mechanics, the probability of obtaining a specific measurement result is determined by the square of the amplitude associated with that outcome, a relationship known as the Born rule [Born, 1926a]. This fundamental principle governs the behavior of quantum systems and has significant implications for quantum computing and entanglement [Nielsen and Chuang, 2002]. By examining the probabilistic nature of quantum mechanics through the measurement lens, we can better comprehend the challenges and opportunities associated with developing and applying quantum technologies

[Preskill, 2018]. The measurement process not only reveals information about the system's state but also leads to the wave function's collapse, affecting the system's state post-measurement [Zurek, 2003]. This intrinsic property of quantum systems introduces unique considerations in designing quantum algorithms and interpreting their results, distinguishing quantum computing from its classical counterpart [Mermin, 2007].

4.3.5 Entanglement

Entanglement in quantum computing refers to creating a correlation between two or more qubits, resulting in a state that cannot be described as a product of individual qubit states. The entanglement of qubits is a fundamental resource for quantum computing and quantum communication [Horodecki et al., 2009]. Different entanglement schemes must be assessed when entangling qubits. The choice of the scheme has various trade-offs, with some providing access to a more significant portion of Hilbert space than others.

Quantum teleportation is based on the principles of entanglement and relies on sharing a maximally entangled state, such as a Bell state, between two parties. By performing specific quantum operations and classical communication, the quantum state of a qubit can be transferred to a distant qubit, effectively "teleporting" the quantum information. This process plays a crucial role in long-distance quantum communication and the development of quantum networks, as it enables the secure and efficient transfer of quantum information over large distances.

The simplest entanglement scheme involves no entanglement at all, which can result in a quantum circuit failing to utilize a quantum system's properties fully. However, in some cases, such as in the optimization process, entangling gates can exacerbate the barren plateau problem, leading to poor performance. In these cases, having no entanglement is preferable. Figure 4.3 shows this entanglement scheme.

In the *ladder* entanglement schemes, each qubit controls a CNOT operation connected to the next qubit. The structure, when visualized, looks like a ladder.

Figure 4.3: A quantum circuit containing no entanglement scheme

This results in a weak entanglement throughout the entire circuit. The main benefit of the ladder entanglement scheme is that it has a low ratio of extra CNOTs added per additional qubit. The cost is only n-1 CNOTs, where n is the number of qubits. Figure 4.4 shows this entanglement scheme.

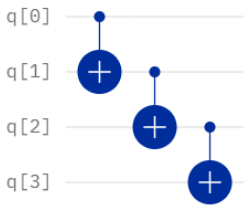


Figure 4.4: A quantum circuit containing the ladder entanglement scheme

The double ladder is an extension of the ladder entanglement scheme. The double ladder has an extra ladder at the end in the other direction. This results in properties like the ladder scheme, but the entanglement is stronger, and there are twice as many CNOTs, which might lead to more noise in the system. Figure 4.5 shows this entanglement scheme.

An entanglement scheme with a very high CNOT to qubit ratio is the *full* scheme. Here, every qubit is connected to all others. This results in a minimum ratio of n^2 CNOTs to qubits. However, if the physical structure of the circuit is not considered, this number can be immensely higher as long as CNOTs are exchanged for many swap gates. In this exchange, a single CNOT is swapped for at least seven. Three are used to swap two qubits, one to perform the CNOT, and the final three to swap the two qubits back. Figure 4.6 shows this entanglement scheme.

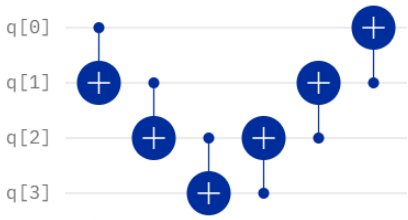


Figure 4.5: A quantum circuit containing the double ladder entanglement scheme

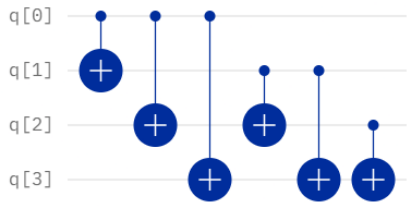


Figure 4.6: A quantum circuit containing the full entanglement scheme

The entanglement schemes mentioned can also be based on the controlled RX (C-RX) gate. The C-RX gate is a two-qubit gate that rotates around the Bloch sphere's X-axis relative to a parameter θ and is controlled by the state of the control qubit. When the control qubit is in the $|1\rangle$ state, the target qubit rotates by an angle θ. This gate allows for creating various entangled states with a single operation, offering flexibility in the design of quantum circuits. It is important to note that when $\theta = \pi$, the C-RX gate is equivalent to the CNOT gate, and when $\theta = 0$, it corresponds to the identity gate (I) [Nielsen and Chuang, 2002]. This highlights the versatility of the C-RX gate in quantum circuit design, as it can generate a wide range of entangled states and perform various operations depending on the chosen parameter value.

The entanglement scheme employing the C-RX gate can adjust the degree of entanglement between qubits through the rotation angle θ. This adjustable nature of the C-RX gate allows the amount of entanglement in the network to become a trainable parameter. The motivation for using the C-RX gate is based on the trade-off between accessing a more significant portion of the Hilbert space by exploiting quantum mechanics and the observation that, in some cases, having no

entanglement (i.e., the ladder scheme) performs better than highly entangled schemes [Grant et al., 2019]. The question arises whether an intermediate level of entanglement would yield the best results. The C-RX gate offers a solution to this problem by enabling the generation of a desired amount of entanglement, providing a unique advantage in the design and optimization of quantum circuits.

However, C-RX gates are not natively supported on all quantum hardware. In such cases, the gate must be decomposed into a sequence of native gates, which could increase the overall gate count and negate the benefits of using the C-RX gate for entanglement. For example, IBM's quantum hardware supports the native gates U3, CNOT, and CZ but not the C-RX gate directly. Implementing the C-RX gate on IBM hardware must be combined with these native gates, potentially increasing the circuit depth and susceptibility to noise.

Figure 4.7 shows an example of a quantum circuit employing the controlled RX gate for entanglement in the ladder scheme.

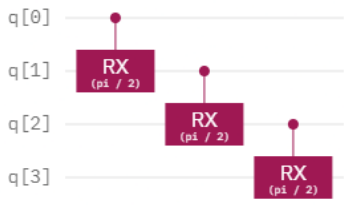


Figure 4.7: A quantum circuit containing the ladder scheme with C-RX instead of CNOT

4.4 Quantum Deep Q-Learning

In this section, we investigate Quantum Deep Q-Learning, an innovative approach that combines quantum computing with deep Q-learning. We first describe Quantum Neural Networks, quantum analogs of classical neural networks, and their potential to enhance learning capabilities. We then illustrate how these networks can be integrated with Deep Q-Learning, resulting in a novel learning algorithm that harnesses the power of quantum computing to tackle complex problems.

4.4.1 Quantum Neural Networks

Quantum Neural Networks (QNNs) [Benedetti et al., 2019] are a type of Parameterized Quantum Circuit (PQC), also known as Variational Quantum Circuits, which share similarities with Artificial Neural Networks (ANNs). QNNs and ANNs receive input, process it through a set of weights, and produce an output. The network or circuit weights are then optimized based on the difference between the actual and desired output [Wold, 2021]. In the following subsections, we will discuss the four primary steps involved in operating a PQC and how they contribute to the overall functionality of QNNs.

Step 1: Encoding the Input

The first step of a PQC involves encoding the input into the system. This is done by transforming the input into a corresponding rotation and applying that rotation to the corresponding qubit [Lloyd et al., 2020]. By encoding classical information into qubits, we map low-dimensional data to a high-dimensional space known as the Hilbert space. This process is reminiscent of kernel methods in classical machine learning and can be a strong foundation for developing effective models.

Step 2: Applying Parameterized Rotations

The second step concerns the parameterized part of PQC. The parameterized Pauli gates rotate the qubits.

Step 3: Entangling Qubits

In the third step, qubits are entangled using specific entangling gates, such as CNOT gates. Entanglement is essential for accessing a more significant part of the Hilbert space and leveraging quantum mechanical properties to achieve efficient computation. However, it has been shown that, in some cases, having no entanglement works better in some scenarios. This raises the question of whether an intermediate level of entanglement might be optimal. The C-RX gate, for instance, can be used to achieve a controlled degree of entanglement.

Step 4: Measuring the Qubits

Once the parameterized transformations and entanglement have been applied, the state of the qubits is measured. These steps are repeated many times (often $10^4 <$), resulting in a distribution of results. The distribution is interpreted as the model output, providing insights into the system's behavior.

This combination of re-uploading, parameterized rotations, and entanglement is called a *layer*. The more layers a PQC has, the more functions it can approximate [Schuld et al., 2021]. At the same time, with more layers, more parameters result in a larger model space that is more difficult to optimize. This property results in stacked layers, such that many sets of entangling and parameterized rotations are after each other.

A common problem in classical ANN is exploding gradients. This occurs when the weights/parameters of the network grow significantly large. Regularization is a technique that is used to punish large weights/parameters. However, exploding gradients are not a concern in QNNs. No matter how much you rotate a particle, its angle can always be represented as $\epsilon[0, 2\pi]$.

As quantum computing matures, many tools have been developed to make quantum computing more accessible. PennyLane is a Python framework for general quantum computing, which has recently included TensorFlow and PyTorch support. TensorFlow is a machine learning library for Python that has recently included functionality for dealing with QNNs.

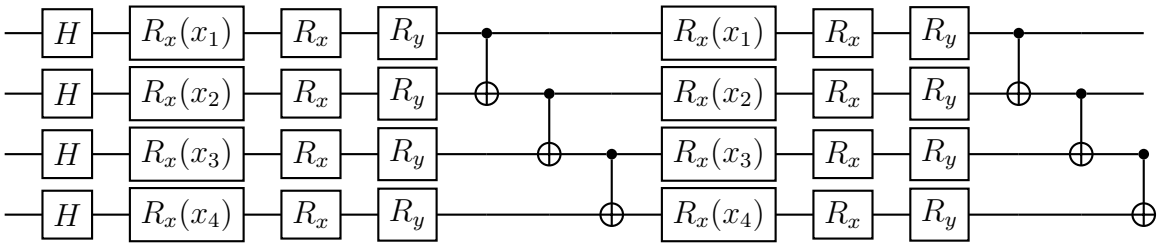
The barren plateau is a phenomenon in quantum computing that can arise in training large-scale quantum circuits, such as quantum neural networks [McClean et al., 2018]. It is a significant challenge for quantum machine learning, as it can severely limit the performance and scalability of quantum algorithms [Holmes et al., 2021]. Several techniques have been proposed to mitigate this issue, such as specific circuit architectures [Skolik et al., 2021], initialization strategies [Grant et al., 2019], and adaptive optimization methods [Khatri et al., 2020]. Researchers continue to explore and develop new approaches to overcome barren plateaus and unlock the full potential of quantum machine learning [Cerezo et al., 2021].

4.4.2 Quantum Deep Q-Learning

Quantum Deep Q-Learning (QDQL) is similar to DQL [Kwak et al., 2021]. The main difference is that in QDQL, parts of the network are replaced with a QNN. This means that the network layers are not classical fully connected layers but instead, combinations of Pauli rotations followed by CNOT gates. Similarly to an ANN, a QNN has weights that will be optimized with the Bellman equation

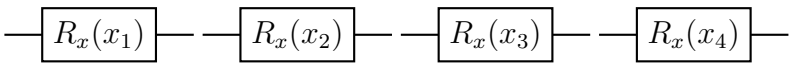
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)],$$

Circuit 31 shows PQC with four qubits, two layers, and the ladder entanglement scheme. The first column of Pauli-X rotations in each layer is the re-uploading embedding.



(31)

In a Parameterized Quantum Circuit (PQC), the angle encoding process, which utilizes Pauli-X rotations, serves as a cornerstone for quantum state preparation. This procedure is represented as follows:



(32)

In this subcircuit, each qubit experiences a rotation enacted by a Pauli-X gate. The corresponding input parameter x_n determines the rotation's angle, effectively encoding the classical data into the quantum states of the qubits.

What is crucial to note here is the repetitive nature of this angle encoding process.

It is performed at the onset of each layer in a PQC employing a data re-uploading strategy. By repeating the encoding at each layer, the quantum model achieves enhanced flexibility and expressiveness, creating a more sophisticated relationship between the input parameters and the resulting quantum states. This repetitive procedure significantly influences the PQC's learning capabilities and overall performance.

Skolik et al. [2022] investigated this "PQC with Re-uploading" technique in the context of Quantum Deep Q-Learning (QDQL). The technique involves repetitively re-introducing the input at the beginning of each PQC layer. This approach has been found to amplify the performance of QDQL by promoting efficient data processing and learning, despite the limitations of qubits and layers. The technique has shown superior performance compared to initial, naive encoding in a PQC, suggesting the immense potential of quantum machine learning in tackling complex problems.

In QDQL, the re-uploading technique is implemented through a series of input-dependent Pauli-X rotations at the beginning of each QNN layer (as illustrated in circuit 31). This process embeds the classical information into the quantum state, enabling the QNN to process and learn from the input data effectively, even when constrained by the number of qubits and layers. This adaptability is critical to the practical implementation of QDQL.

4.5 Literature Review

The burgeoning field of quantum machine learning has witnessed numerous advances in recent years, with contributions spanning from theoretical frameworks to practical applications. Several quantum algorithms have been proposed and benchmarked, demonstrating their potential in various machine learning tasks. This literature review highlights key advancements underpinning our work, focusing on Parameterized Quantum Circuits (PQCs), Metric Quantum Learning, Quantum Circuit Networks, Quantum Reinforcement Learning, and the relationship between Quantum Models and Partial Fourier Series. In each area, we delve into the significant findings, their implications for the field, and the open

challenges that motivate ongoing research.

4.5.1 Parameterized Quantum Circuits

Sim et al. [2019] provided a theoretical framework for characterizing and comparing parameterized quantum circuits (PQCs) without focusing on specific algorithms or applications. The work identified the saturation and saturated value rate indicators of PQC performance and offered insights for designing and selecting suitable circuits. The study also proposed a classical simulation framework to address PQC challenges by defining computable and comparable quantities for circuit designs and fragments. The authors discussed the trade-off between expressibility and optimizability in PQCs and laid the groundwork for future research [Sim et al., 2019].

4.5.2 Metric Quantum Learning

Lloyd et al. [2020] presented ”metric quantum learning” as an innovative methodology within the realm of quantum machine learning. This approach involves embedding classical data points into quantum states and performing comparisons using optimal quantum measurements. The key strength of this method lies in its compatibility with small, shallow quantum circuits, making it suitable for deployment on near-term quantum computing platforms.

Interestingly, the architecture proposed by Lloyd et al. [2020] resembles a ResNet-like structure characterized by a sizeable classical network supporting a comparatively more minor quantum component. This design leverages the strengths of both classical and quantum computation, providing a robust and flexible framework for tackling complex learning tasks.

Despite these promising features, it is essential to note that classical computers typically struggle to replicate the results produced by quantum embeddings, primarily due to the quantum-specific phenomena leveraged in the process. Consequently, this raises the question of whether a practical quantum advantage can be attained through this approach. Despite the impressive achievements

reported by Lloyd et al. [2020], this question remains an open challenge within the field of quantum machine learning.

4.5.3 Quantum Circuit Networks

Wold [2021] presented a Python framework for implementing and training dense neural networks (DNNs), quantum neural networks (QNNs), and quantum circuit networks (QCNs) on various datasets. The study highlighted the vanishing gradient phenomenon in QCNs and DNNs and demonstrated the potential for training QCNs to be more expressive than similarly sized DNNs. The authors also compared QCNs and DNNs on real-world datasets, revealing that QCNs may be better suited for specific training problems [Wold, 2021].

Jerbi et al. [2021] investigated the design of quantum reinforcement learning (RL) agents based on PQCs. The researchers proposed several constructions, including the SOFTMAX-PQC model, and examined the impact of design choices on learning performance. The study contributed to understanding quantum RL agents' potential advantages in near-term quantum devices [Jerbi et al., 2021].

4.5.4 Quantum Reinforcement Learning

A recent study provided numerical evidence supporting the potential advantages of quantum RL over classical RL in terms of sample efficiency and reduced parameter requirements [Wang et al., 2020]. The authors benchmarked quantum RL agents against classical fully-connected neural networks in OpenAI Gym environments such as CartPole and Acrobot. The quantum RL agents displayed faster convergence rates and required significantly fewer trainable parameters than classical RL agents. Moreover, the study presented the first quantum RL agent capable of completing the LunarLander task in OpenAI Gym, demonstrating the potential of quantum RL in practical applications.

However, a balanced critique of these results raises a few pertinent issues. Firstly, the quantum RL agent employed in this study is a hybrid, incorporating a classical layer. Thus, the architecture is not purely quantum but a blend of classical and quantum elements. Therefore, while a quantum-classical hybrid solution has

demonstrated its ability to solve the LunarLander task, the challenge of achieving this with a purely quantum agent remains unsolved.

Secondly, their model could be characterized as a classical model with an unusual feature scaling and increased noise, given that the quantum portion of their model only induces minor shifts in the initial input values. The potential of quantum systems, particularly in reinforcement learning, likely extends far beyond what this hybrid model has demonstrated.

Additionally, the absence of entanglement in their model suggests that they needed to fully utilize the power of quantum systems to explore large problem spaces efficiently.

Most critically, the study did not try to optimize the classical model. This introduces a bias as the quantum technique received more attention and fine-tuning, possibly skewing the comparison in its favor. Therefore, while the presented results are promising, the potential of quantum systems in reinforcement learning requires further unbiased investigation, taking full advantage of quantum capabilities and ensuring a fair comparison with optimized classical models.

4.5.5 Quantum Models and Partial Fourier Series

Schuld et al. [2021] systematically mapped a wide range of quantum machine learning models and partial Fourier series. The study examined the impact of data encoding mechanisms on model expressivity and established connections between quantum machine learning and classical machine learning concepts. The authors raised several open questions, such as how the structure of trainable circuit blocks influences accessible Fourier coefficients and whether knowledge of expressed function classes can inform applications for quantum machine learning [Schuld et al., 2021].

5 Methodology

5.1 Environment and Tools

Here is a brief description of each library and framework used in the implementation:

The implementation was carried out using Python [van Rossum, 1995]. This versatile, high-level programming language is widely used for scientific computing, data analysis, and machine learning tasks due to its readability, ease of use, and extensive library support. Several libraries and frameworks were utilized, including:

TensorFlow [Abadi et al., 2016]: An open-source machine learning library developed by Google Brain that allows users to quickly build and deploy machine learning models. TensorFlow supports various types of neural networks, including feed-forward, recurrent, and convolutional networks. It is designed for efficient numerical computing and can utilize GPUs and TPUs for accelerated training.

Keras [Chollet et al., 2015]: A high-level neural networks API that can run on top of TensorFlow. Keras simplifies the process of building and training neural networks by providing an intuitive interface and pre-built layers. It is designed for rapid prototyping and is often used for deep learning research and applications.

Numpy [Oliphant, 2006, Van Der Walt et al., 2011]: A library for numerical computing in Python that provides support for arrays, linear algebra, and various mathematical functions. Numpy is a library for scientific computing and is widely used in machine learning and data analysis tasks.

Pandas [McKinney, 2010]: A powerful data manipulation library for Python that provides data structures, such as DataFrame and Series, to handle and analyze data flexibly and efficiently. Pandas is particularly useful for dealing with tabular data and supporting various formats, such as CSV, Excel, and SQL.

PennyLane [Bergholm et al., 2018]: An open-source quantum computing library that focuses on quantum machine learning and optimization. It provides a unified interface for defining and simulating quantum circuits with various quantum

computing backends, such as Qiskit, Cirq, and Strawberry Fields. PennyLane is designed to integrate seamlessly with existing machine learning libraries, such as TensorFlow and PyTorch, enabling the development of hybrid quantum-classical models.

By using these libraries and frameworks, the implementation of the classical and quantum Deep Q-Learning models can be conducted efficiently and effectively, leveraging the strengths of each library to achieve the desired results.

5.2 Open AI Gym

Open AI Gym [Brockman et al., 2016] is a popular toolkit for developing and comparing reinforcement learning algorithms. The toolkit provides various environments researchers and practitioners use to test their algorithms, from simple toy problems to complex real-world scenarios. The environments are designed to be easy to use and modify, and they provide standardized interfaces for interacting with the environment, collecting observations, and receiving rewards. This section introduces two environments in this work's Open AI Gym framework: Cart Pole and Lunar Lander. Those environments are commonly used in the reinforcement learning literature as benchmarks for evaluating the performance of different algorithms. They are challenging but tractable problems that require agents to learn sophisticated control policies to achieve high scores. Also, there is a sense of progression in difficulty, with the cart pole having a small observation and action space, while the lunar lander has larger search spaces and is more complex.

5.2.1 Cart Pole

Cart Pole is an environment where a cart with a pole attached moves left and right along a frictionless track to balance the pole upright. The agent aims to keep the pole balanced for as long as possible. There are two possible actions, apply force to the left or the right. The agent receives a reward of 1 for each timestep the pole remains balanced. The episode terminates if the angle of the pole exceeds a certain threshold ($\pm$12 degrees). An observation of the environment contains the

position and velocity of the cart and the angle and angular velocity of the pole. Figure 5.1 shows a visual representation of the environment. The environment is initialized with a random pole angle, close to upright. The default length of an episode is 200 timesteps.

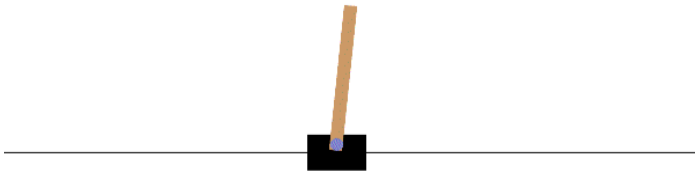


Figure 5.1: Snapshot from the Cartpole v1 OpenAI Gym environment

5.2.2 Lunar Lander

is an environment where the agent aims to safely land a lunar lander spacecraft on a landing pad in moon-like conditions. There are four possible actions, fire the left engine, fire the main engine, fire the right engine, or do nothing. The side engines have one-tenth the power of the main engine. The agent receives a reward based on the position and velocity of the lander and the fuel consumption. The episode terminates if the lander crashes, runs out of time, or if it lands successfully on the landing pad. The observation of the environment contains the lander's x and y position, the lander's x and y velocity, and the lander's angle and angular velocity if the left leg is in contact with the ground and if the right leg is in contact with the ground. Figure 5.2 shows a visual representation of the environment. The starting velocity of the lander is initialized randomly. The default length of an episode is 200 timesteps.

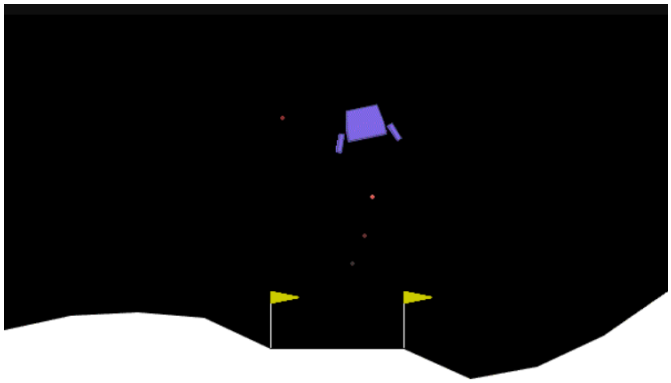


Figure 5.2: Snapshot from the Lunar Lander v2 OpenAI Gym environment

6 Implementation

The purpose of the Implementation section is to detail the process of implementing the classical and quantum Deep Q-Learning models for the chosen OpenAI Gym environments. This section highlights the programming languages, libraries, and frameworks used and discusses the rationale behind the design and implementation choices. Documenting the implementation process ensures reproducibility and facilitates the understanding and replication of the experiments by others.

6.1 Classical DQL Implementation

The classical models were implemented using feed-forward neural networks as function approximators in the DQL framework, following the well-established approach of representing Q-functions in classical deep reinforcement learning algorithms [Mnih et al., 2015]. These networks have been proven effective in learning complex, nonlinear relationships between states and actions in various environments [Mnih et al., 2013].

The chosen architecture for the classical models consisted of several fully connected layers with a flexible number of nodes per layer, as determined during the hyperparameter search process. This is explained further in the experiments

section. These networks utilized rectified linear unit (ReLU) activation functions [Nair and Hinton, 2010] in the hidden layers, which have been shown to improve training efficiency and mitigate the vanishing gradient problem often encountered in deep learning models [Glorot et al., 2011]. The output layer had a linear activation function to provide a continuous range of Q-values for each action.

Mean squared error (MSE) is the loss function, as it is commonly used in DQL-based algorithms [Mnih et al., 2015]. The loss function measures the discrepancy between the predicted and target Q-values obtained using the Bellman equation. Minimizing this loss results in better Q-value approximations and improved learning.

The random search for hyperparameter tuning was performed to find the best combination of the number of layers, learning rate, and the number of nodes per layer. The search space for the number of layers ranges from 2 to 4, as deeper networks might lead to overfitting or slower convergence in some cases [Zhang et al., 2016]. The learning rate search space included values of 0.03, 0.01, 0.003, 0.001, and 0.0003, which cover a broad range of commonly used rates in the literature [Smith, 2018]. The number of nodes per layer was tested with values of 4, 8, 16, and 32, allowing the model to adapt to various levels of complexity. The values selected ensure that the network weights are comparable in classical and quantum agents.

The classical models were updated using optimizer Adam [Kingma and Ba, 2014] during the training process. The training procedure involved multiple episodes, during which the agent interacted with the environment and collected experience to improve its policy. The use of experience replay buffer [Lin, 1993] was also incorporated to enhance the stability and convergence of the learning process.

6.2 Quantum DQL Implementation

The quantum models were implemented using parameterized quantum circuits as function approximators in the DQL framework, following recent advancements in quantum machine learning that leverage quantum systems to perform complex

tasks [Biamonte et al., 2017]. These circuits have shown promise in learning intricate, nonlinear relationships between states and actions in various settings [Schuld et al., 2020].

The chosen architecture for the quantum models incorporated different types of circuit designs, with angle encoding used for qubit encoding, a widely adopted technique for encoding classical information into quantum states [Schuld et al., 2020]. The circuit comprised Hadamard gates applied to all qubits, followed by layers, where each consists of three steps: input uploading, rotations based on trainable parameters, and entanglement. Pauli-X rotations based on the input values (referred to as embedding) [Farhi et al., 2014b]. Then, several a pair of Pauli-X and Pauli-Y rotations rotates the qubits based on trainable parameters θ followed by an entanglement scheme. The supported entanglement schemes included ladder, double ladder, full, and none, previously explored in the context of variational quantum algorithms [Peruzzo et al., 2014]. The circuit output was connected to a dense output layer with linear activation, providing a continuous range of Q-values for each action.

The random search for hyperparameter tuning was conducted to find the best combination of the number of layers, learning rate, entanglement scheme, and rotation gates. The search space for the number of layers ranges from 2 to 4, following the same rationale as classical models [Zhang et al., 2016]. The learning rate search space comprised values of 0.03, 0.01, 0.003, 0.001, and 0.0003, which cover a broad range of commonly used rates in the literature [Smith, 2018]. The entanglement schemes and rotation gates were tested with various configurations, allowing the model to adapt to different levels of complexity.

The quantum models were updated using the Adam optimizer [Kingma and Ba, 2014] during the training process. The Adam optimizer is an advanced gradient descent method that maintains a running average of previous gradients. This characteristic aids in avoiding getting stuck in local minima, facilitating more efficient learning.

The training procedure was conducted over multiple episodes. In each episode, the agent interacted with the environment, generating experiences to refine its

policy and incrementally improving its decision-making capabilities.

In addition to this episodic learning, an experience replay buffer was implemented [Lin, 1993]. This technique allows the model to store and revisit past experiences, providing a diverse set of training samples and avoiding the issue of correlated experiences. Using an experience replay buffer enhanced the stability and convergence of the learning process, offering a level of consistency with the training methodologies employed for classical models.

6.3 Deep Q-Learning Algorithm

The implemented DQL algorithm used the Bellman equation for updating the Q-values. State, next state, reward, action, and termination information were stored in a replay buffer, and the model was trained on a batch of data at every timestep. No specific adaptations were made to the DQL algorithm to accommodate the quantum models.

The learning algorithm implementation is described in algorithm 1

Algorithm 1 Deep Q-Learning with Replay Memory

Initialize replay memory D
Initialize agents action function Q with random weights θ
Initialize exploration rate ϵ
Set minimum exploration rate ϵ_{min}
Set exploration rate decrement factor α
Set discount factor γ
Set mini-batch size B
for episode $= 1$ to M **do**
    Initialize state s
    for timestep $= 1$ to T **do**
        With probability ϵ select a random action a_t
        Otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
        Execute action a_t in the environment and observe reward r_{t+1} and next state s_{t+1}
        Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in D
        if memory contains at least B transitions **then**
            Sample a random mini-batch of transitions (s_i, a_i, r_i, s_{i+1}) from D
            Compute q-values for state and next state batches using the model
            Compute target q-values y_i based on the Bellman equation
            Compute gradients based on target and predicted q-values
            Update q-values for state-action pairs in the state batch using the computed gradients
            Update the model using the updated state and q-value batches
        if exploration rate is higher than minimum exploration rate **then**
            Decrease the exploration rate using the decrement factor α
        $s \leftarrow s'$

6.4 Code Repository and Reproducibility

The code for the implementation is available at https://github.com/mathiassandnes/QuantumDQN. A set seed was used to control the randomness in the implementation; however, the seed does not control the environments' random initialization of each episode. Other steps to ensure the reproducibility of the experiments include using version control and providing clear documentation of the implementation process.

7 Experiments

This chapter presents a suite of experiments designed to explore and compare the performance of classical and quantum Deep Q-Learning (DQL) models on various reinforcement learning tasks. The experiments specifically focus on two OpenAI gym environments, the Cart Pole and the Lunar Lander, which serve as canonical benchmarks in reinforcement learning research.

The first two experiments delve into the performance of classical DQL models, utilizing feed-forward neural networks as function approximators. By conducting a random search over various hyperparameters, such as the number of layers, learning rate, and number of nodes per layer, we aim to understand their influence on the model's performance. Moreover, these experiments provide a classical baseline against which the performance of quantum models can be compared.

Subsequently, in the third and fourth experiments, we transition to quantum models, employing Parameterized Quantum Circuits (PQCs) as function approximators. Cart Pole and Lunar Lander are the same environments for consistency and comparability. Here, the hyperparameters under investigation encompass the number of layers, learning rate, number of qubits, entanglement scheme, and the types of rotation gates.

Building upon these, the final two experiments introduce additional complexity to the quantum models - trainable entanglement. In these cases, we add a separate set of trainable parameters that control the degree of entanglement between the qubits, allowing for dynamic entanglement during the learning process.

Throughout all these experiments, we track various metrics during training and evaluation, encompassing aspects such as episode number, scores, exploration rates, actions, and raw observations. The overarching aim of these experiments is not only to understand the performance of classical and quantum DQL models but also to delve into the nuances of their operation and the factors that influence their success.

7.1 Experiment 1: Classical DQL on Cart Pole

In this experiment, we employed feed-forward neural networks as function approximators in the DQL framework to solve the Open AI gym environment Cart Pole. We aimed to investigate the impact of hyperparameters on the performance of the classical models. We conducted a random search over the following hyperparameters:

- **Number of layers:** Networks with 2 to 4 hidden layers were investigated, exploring various depths and their effect on learning and model complexity.
- **Learning rate:** Learning rates of 0.03, 0.01, 0.003, 0.001, and 0.0003 were considered, aiming to identify the optimal balance between convergence speed and stability.
- **Number of nodes per layer:** Networks with 4, 8, 16, and 32 nodes per layer were explored to examine the impact of model capacity on performance.

We tracked the following metrics during training and evaluation:

- **Episode:** The episode number.
- **Training score:** The cumulative reward obtained during the training episode.
- **Evaluation score:** The cumulative reward obtained during the evaluation episode.
- **Epsilon:** The exploration rate used during this training episode.
- **Training actions:** A list containing the actions performed in this training episode.
- **Evaluation actions:** A list containing the actions performed in this evaluation episode.
- **Evaluation observations:** A list containing the raw observations in this evaluation episode.
- **Model output:** A list containing the raw output in this evaluation episode.

After the random search, we selected the best hyperparameters and performed another run with a narrower search space based on the results. Finally, we chose the best model based on the evaluation score achieved during training.

7.2 Experiment 2: Classical DQL on Lunar Lander

This experiment was conducted like experiment 1, but the Lunar Lander environment was used instead. The random search was narrowed slightly based on the results from experiment 1. This is discussed further in the results section.

7.3 Experiment 3: Quantum DQL on Cart Pole

This experiment employed parameterized quantum circuits (PQC) as function approximators in the DQL framework to solve the Open AI gym environment Cart Pole. We aimed to investigate the impact of hyperparameters on the performance of the quantum models. We conducted a random search over the following hyperparameters:

- **Number of layers:** Circuits with 1 to 10 layers were investigated to examine the impact of circuit depth on model performance and complexity.
- **Learning rate:** Learning rates of 0.3, 0.1, 0.03, 0.01, and 0.003 were considered, mirroring the learning rates explored in the classical models while reducing each value by order of magnitude.
- **Number of qubits:** The number of qubits equal to the input features in each environment was used, ensuring that the quantum models could effectively represent the state spaces.
- **Entanglement scheme:** Different entanglement schemes, including none, ladder, double ladder, and full entanglement, were explored to study their effect on the learning capabilities of the quantum models.
- **Rotation gates:** RX and RY were used in the parameterized quantum circuits, providing a simple yet expressive means to manipulate qubit states.

We tracked the same metrics as in the classical experiments and used the same selection process to choose the best hyperparameters and model as in experiments 7.1 and 7.2.

7.4 Experiment 4: Quantum DQL on Lunar Lander

This experiment was conducted like experiment 2, but the Lunar Lander environment was used instead. The random search was narrowed slightly based on the results from experiment 2. This is discussed further in the results section.

7.5 Experiment 5: Quantum DQL with Trainable Entanglement on Cart Pole

This experiment employed parameterized quantum circuits (PQC) with trainable entanglement as function approximators in the DQL framework to solve the Open AI gym environment Cart Pole. We aimed to investigate hyperparameters' impact on the quantum models' performance with trainable entanglement. We conducted a random search over the following hyperparameters:

- **Number of layers:** Circuits with 1 to 10 layers were investigated to examine the impact of circuit depth on model performance and complexity.
- **Learning rate:** Learning rates of 0.3, 0.1, 0.03, 0.01, and 0.003 were considered, mirroring the learning rates explored in the classical models while reducing each value by order of magnitude.
- **Number of qubits:** The number of qubits equal to the input features in each environment was used, ensuring that the quantum models could effectively represent the state spaces.
- **Entanglement scheme:** Different entanglement schemes, including none, ladder, double ladder, and full entanglement, were explored to study their effect on the learning capabilities of the quantum models with trainable entanglement.

- **Rotation gates:** RX and RY were used in the parameterized quantum circuits, providing a simple yet expressive means to manipulate qubit states.
- **Entanglement training:** A separate set of trainable parameters was introduced to control the degree of entanglement between the qubits, allowing for dynamic entanglement during the learning process.

We tracked the same metrics as in the classical and other quantum experiments and used the same selection process to choose the best hyperparameters and model as in previous experiments.

7.6 Experiment 6: Quantum DQL with Trainable Entanglement on Lunar Lander

This experiment was conducted like experiment 5, but the Lunar Lander environment was used instead. The random search was narrowed slightly based on the results from experiment 5. This is discussed further in the results section.

8 Results and Discussion

This section reports the findings of our experiments comparing the performance of quantum DQL with classical approaches on the CartPole and Lunar Lander environments. We conducted a comprehensive hyperparameter search to evaluate the quantum agents' training speed, number of parameters, and scalability. Our goal in this section is to provide an in-depth understanding of the interpretable concepts in the results obtained during the experiments. All the data collected during the experiments are available in our GitHub repository (https://github.com/mathiassandnes/QuantumDQN). The following subsections present an overview of the performance of all trained models, followed by a detailed analysis of performance metrics categorized according to different hyperparameters.

8.1 CartPole

This section includes the results obtained in experiments 1, 3, and 5. We trained classical and quantum agents on the CartPole environment, utilizing a random search over a range of hyperparameters to identify the optimal settings for each approach. The specific parameters used are described in detail in the experiments section 7. The classical model in our study plays a crucial role, serving as a reference point against which we can juxtapose the performance of the quantum model. This comparative analysis is vital for gauging the effectiveness and efficiency of the quantum model concerning an optimized classical agent tailored explicitly for this specific problem. It is imperative to recognize that within our implementation of the CartPole environment, the score range is demarcated with an upper limit of 500, representing the highest possible score, and a lower limit of 0, indicative of the minimum possible score. The spectrum of scores between these two boundaries provides a quantitative measure to evaluate the agents' performance under consideration.

The models are executed throughout the training for up to 100 timesteps, followed by five evaluation episodes with a maximum of 500 timesteps. This approach ensures rigorous evaluation of the models and credible results, as the actions,

observations, and raw model outputs are saved for all episodes. Remember that a score above 100 is a fair solution, but we want to see which techniques can balance the pole for a long time.

In this section, we present several results using boxplots. The elements of these boxplots are described as follows: The box itself signifies the interquartile range (IQR), which contains the middle 50% of the scores surrounding the median. Inside the box, the orange line denotes the median score, splitting the data into upper and lower halves. Extending from the box are whiskers that indicate the highest and lowest values within the first and fourth quartiles, excluding any outliers. Circles represent outliers; these data points are notably distinct. This standardized visual representation helps us effectively understand the spread and skewness of our data.

8.1.1 Accumulated results

All Classical

First, we analyze the performance of every classical model trained in the CartPole environment after every training episode, including during training.

In Figure 8.1, it is evident that the classical agents generally demonstrate improvement over time, regardless of hyperparameters.

A distinct line is visible at the area where the evaluation score is 500, corresponding to the agents successfully solving the environment. As our implementation of early stopping only selects for improvements, meaning that the agents that have solved the environment will stop training after the early stopping threshold is passed, as it is impossible for performance to increase past the maximum limit.

Another noteworthy observation is that the scatter plot has data populating over a large section, indicating diverse performances obtained over time.

All Quantum

Turning our attention to quantum models, Figure 8.2 showcases the performance of every quantum model evaluated during the training process. Notably, these

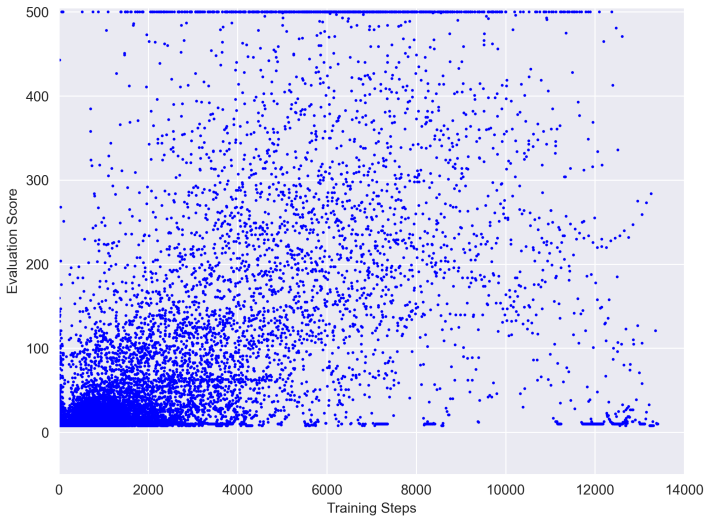


Figure 8.1: Evaluation score as a function of training step for classical models in the CartPole environment over all episodes. The experimental setup can be found in Experiment 7.1.

models are trained for a significantly shorter duration (less than 8000 training steps) than their classical counterparts (less than 14000 training steps). This discrepancy stems from the inherent slow simulation of quantum computing on classical hardware. Despite these constraints, it is clear that the majority of the models fail to attain a stable, high-performance level. This outcome likely reflects the stark contrast in the number of parameters between the quantum and classical models. This aspect will be further dissected in subsequent sections of this analysis.

Interestingly, the range of performance among quantum models is considerably narrower than that observed in classical models. This observation could be attributed to the fewer parameters, which might lead to more restrictive model space and, possibly, heightened sensitivity to hyperparameter selection. Alternatively, this could also be an inherent property of quantum machine learning, hinting at this emerging field's unique characteristics and limitations.

All Quantum with Trainable Entanglement

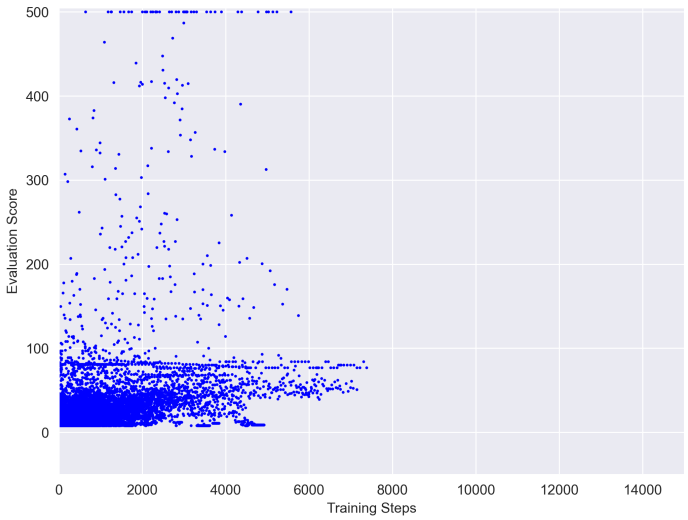


Figure 8.2: Evaluation score as a function of training step for quantum models in the CartPole environment over all episodes. The experimental setup can be found in Experiment 7.3.

The introduction of trainable entanglement, which empowers the model to learn the level of entanglement, must yield a clear improvement in overall performance. This phenomenon is depicted in Figure 8.3, illustrating the inherent instability of these quantum agents operating with a low parameter count. Nevertheless, it is noteworthy that some configurations of hyperparameters can accomplish impressive scores, effectively solving the environment. This highlights the potential power of trainable entanglement when harnessed effectively and suggests that future work could potentially unearth hyperparameter configurations that enable more quantum agents to achieve similarly high performance. Performance could also improve with more extended training. We leave this experiment for future work.

Best Classical

Upon examining the best episode of each classical model configuration run, it is clear that most agents successfully solve the problem. This strongly emphasizes the efficacy of the established Deep Q-Learning (DQL) technique in handling such

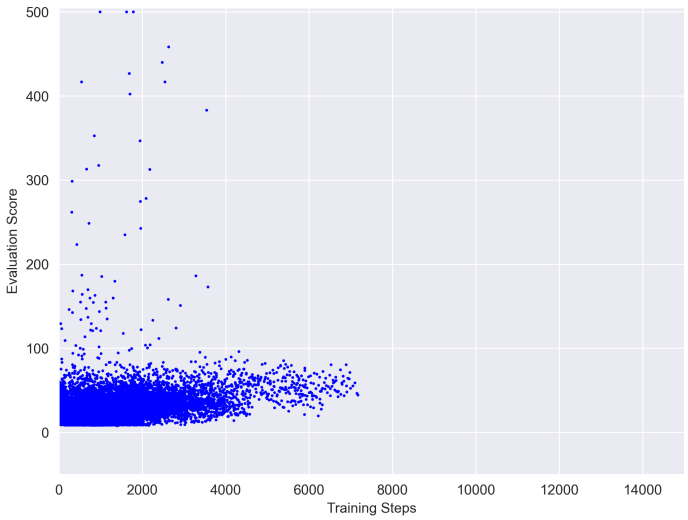


Figure 8.3: Evaluation score as a function of training step for quantum models with trainable entanglement in the CartPole environment over all episodes. The experimental setup can be found in Experiment 7.5.

reinforcement learning tasks. The distribution of these best episodes in relation to performance is presented in Figure 8.4. While some agents might require a longer duration to achieve a perfect score, this nonetheless underscores the effectiveness and resilience of optimization algorithms. The performance of these classical agents also serves as a valuable benchmark for evaluating the performance of quantum agents in similar tasks. It helps to identify the areas where quantum computing can potentially make significant contributions.

Best Quantum

In our analysis of every quantum model run in figure 8.5, we identify four distinct groups of performance. The low-performing group critically fails at solving the environment, obtaining a score close to zero. The two middle groups show performance slightly better than random actions but are still unable to solve the environment thoroughly. The first of these groups is tightly centered between a score of 50 and 100. The second middle group, consisting of outliers scattered between 100 and 400 scores, performs well but cannot reach the highest possible

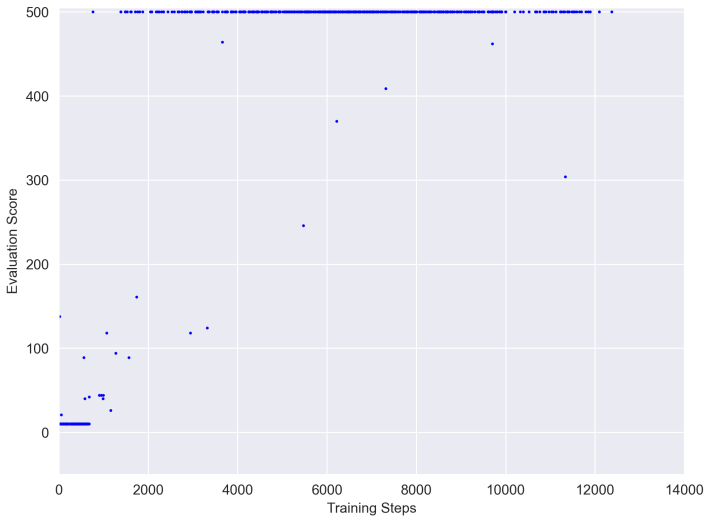


Figure 8.4: Evaluation score as a function of training step of classical models in the CartPole environment for the best episodes. The experimental setup can be found in Experiment 7.1

performance. This group is the smallest, yet we believe that with further training and refinement of the model's policy, it is likely to increase these outliers' performance to the maximum possible score. Lastly, the high-performance group comprises models that successfully solve the environment at an evaluation score of 500, demonstrating the best performance.

Best Quantum with Trainable Entanglement

With the inclusion of trainable entanglements, we can see fewer poor-performing agents. This is also because the hyperparameter search is narrower and based on the results obtained in experiment 7.3. However, there is a low amount of agents solving the environment on the maximum possible score. This raises concern about the applicability of this approach. It is important to note that all quantum agents are trained for a shorter time than we would like. We believe that further training would prove more convincing results of the benefits of quantum machine learning.

Comparison

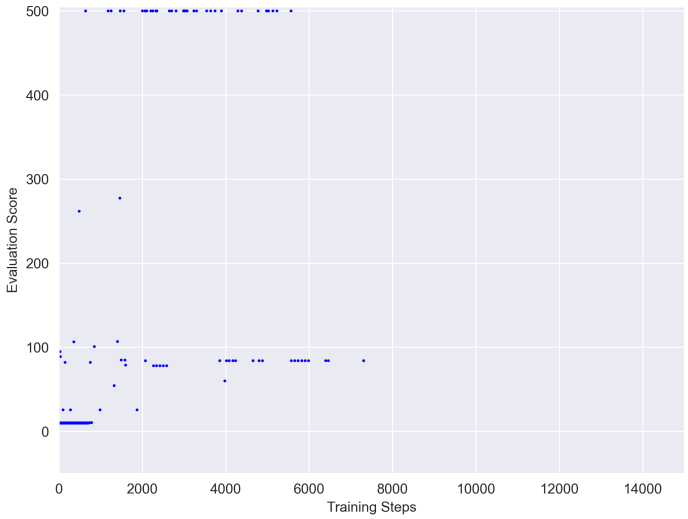


Figure 8.5: Evaluation score as a function of training step for quantum models in the CartPole environment for the best episodes. The experimental setup can be found in Experiment 7.3.

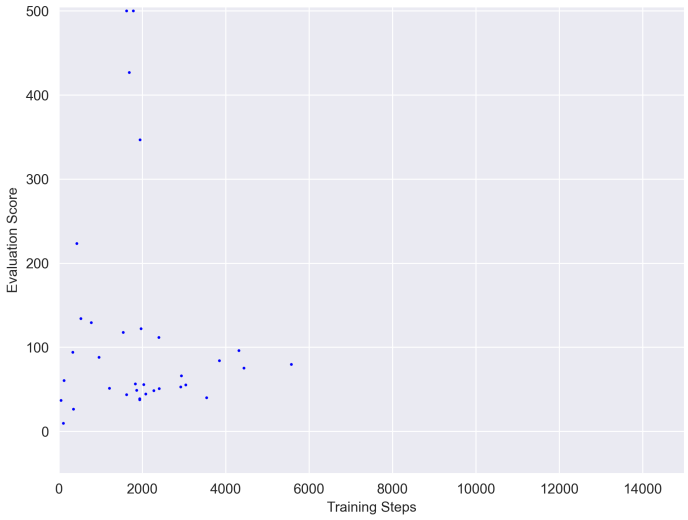


Figure 8.6: Evaluation score as a function of training step for quantum models with trainable entanglement in the CartPole environment for the best episodes. The experimental setup can be found in Experiment 7.5.

Figure 8.7 shows the previous three plots as a boxplot for easier comparison. We can see that the quantum agents with trainable entanglement does better overall, but this might also be because the choice of hyperparameters for that experiment 7.5 was reduced based on experience from experiment 7.3

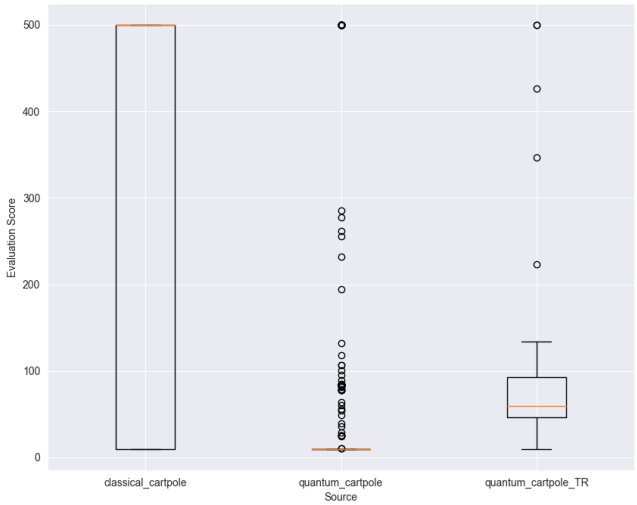


Figure 8.7: Distribution of best scores achieved by classical DQL, quantum DQL, and quantum DQL with trainable entanglement in the CartPole environment, illustrated as boxplots. For a detailed description of the boxplot elements, refer to the introduction of this section. The experimental setups can be found in Experiments 7.1, 7.3, and 7.5 respectively.

Having familiarized ourselves with the range of classical and quantum models trained in our study, the next logical step is to categorize these models based on distinct hyperparameters. This allows us to observe any performance variations attributable to these hyperparameters. Through this examination, we can glean valuable insights into the individual impact of specific hyperparameters on the models, ultimately guiding us toward optimizing their performance.

8.1.2 Learning Rate

In this subsection, we delve into the influence of the learning rate on our models. We begin by investigating the sensitivity of evaluation scores to learning rate changes, then by examining its impact over time for all episodes and the top-performing ones.

Classical

Most classical agents could solve the environment, so high-performance levels were noted across all learning rates. Figure 8.8 is a box plot showcasing the best episodes from each run, clearly visualizing performance variations with different learning rates.

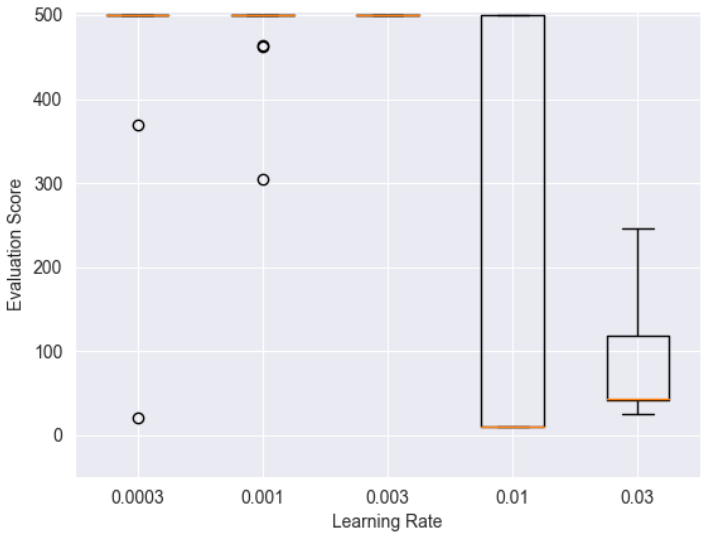


Figure 8.8: Sensitivity of classical models in the CartPole environment to different learning rates, represented as boxplots. For a detailed description of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.1.

Quantum

In contrast, the performance of quantum agents is substantially impacted by the choice of the learning rate. As highlighted in other research, such as the study by Wold [2021], quantum agents generally favor higher learning rates. This pattern is

also reflected in our results, as illustrated in Figure 8.9, where the most promising outcomes were achieved with a learning rate of 0.03. It is worth noting that the results for a learning rate of 0.1 are absent due to a loss of data during the transition to a new data format. Observing that there were outliers who managed to solve the environment with a learning rate of 0.01, we speculate that further investigation into the learning rate interval between 0.01 and 0.3 may yield exciting insights. Future research should consider experimenting with these values.

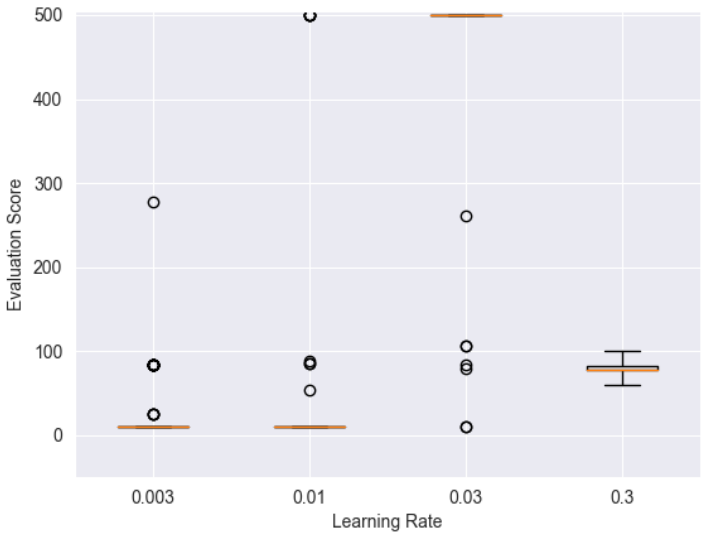


Figure 8.9: Sensitivity of quantum models in the CartPole environment to different learning rates, depicted as boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.3.

Quantum with Trainable Entanglement

With the performance of quantum agents peaking at a learning rate of 0.03, we decided to use this value exclusively for quantum agents with trainable entanglement. The distribution of the results using this learning rate is illustrated in Figure 8.10.

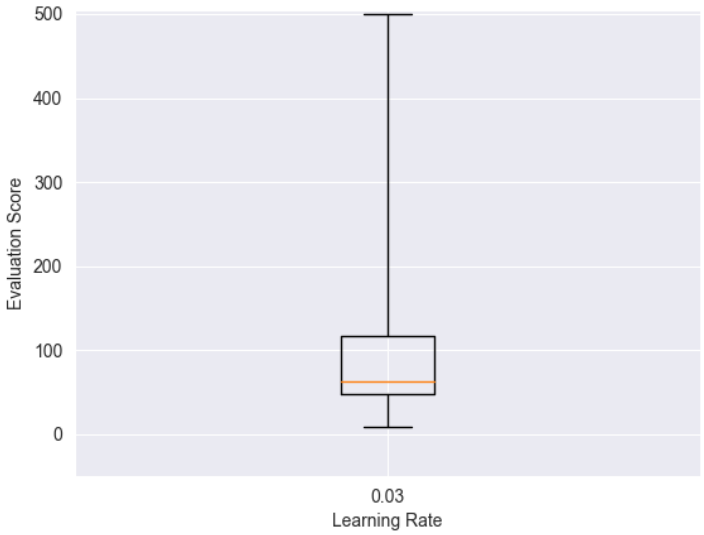


Figure 8.10: Sensitivity of quantum models with trainable entanglement in the CartPole environment to different learning rates, depicted as boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.5.

8.1.3 Number of network weights

In this subsection, we will explore the impact of the number of weights on the performance of our models during the experiments. Our analysis will focus on the sensitivity of this parameter, followed by the examination of results over time for both all and the best episodes.

Classical

The classical agents are categorized into three groups based on the number of parameters: those with less than 1,000 parameters, those with parameters ranging from 1,000 to 10,000, and those with more than 10,000 parameters. On the other hand, the quantum agents, which have significantly fewer parameters, are classified with less than 100 parameters, those with parameters between 100 and 200, and those with more than 200 parameters.

Looking first at the classical models, we observe that the number of weights does not significantly influence the performance of these agents, which consistently

perform well across all groups.

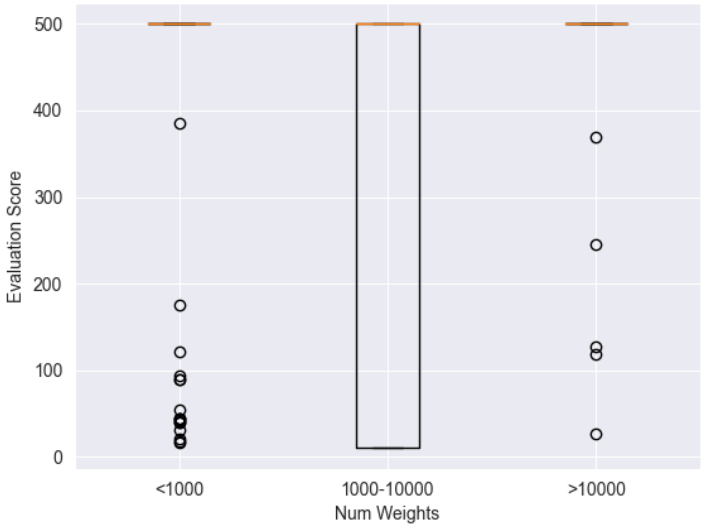


Figure 8.11: Sensitivity of classical models in the CartPole environment to the number of weights, represented as boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.1.

Quantum

In contrast to classical models, quantum models show a more diverse range of outcomes in relation to the number of weights. Our findings suggest that the sweet spot for quantum models lies between 100 and 200 weights, where we observed the most optimal outcomes. However, we must acknowledge a potential bias in this result: our experiments included a relatively limited number of quantum agents with more than 200 weights. Therefore, increasing the number of weights and the extent of training might lead to enhanced performance in quantum models. This hypothesis opens up an exciting avenue for further investigation.

Quantum with Trainable Entanglement

We see increased parameters when we integrate trainable entanglement into the models, which correlates with improved results. Specifically, models with more than 200 weights display a median score significantly higher than those with fewer

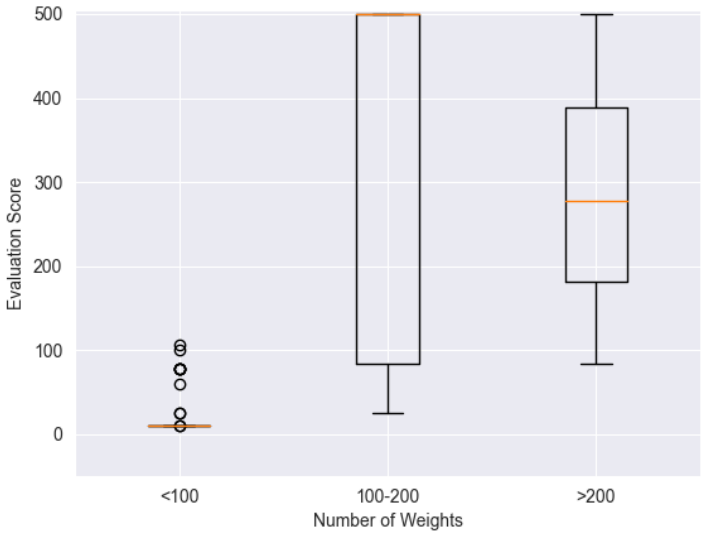


Figure 8.12: Sensitivity of quantum models in the CartPole environment to the number of weights, displayed as boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.3.

weights. However, we should note that the group of models with a medium number of weights harbors a more significant number of high-performing agents, despite both groups featuring individuals that reached the maximum score. It is important to mention that the smallest group is not represented in this plot. Adding additional weights resulted in no quantum agents, with trainable entanglement having less than 100 weights. This suggests a potential investigation into how parameter count, specifically in terms of weights, impacts the performance of quantum models with trainable entanglement.

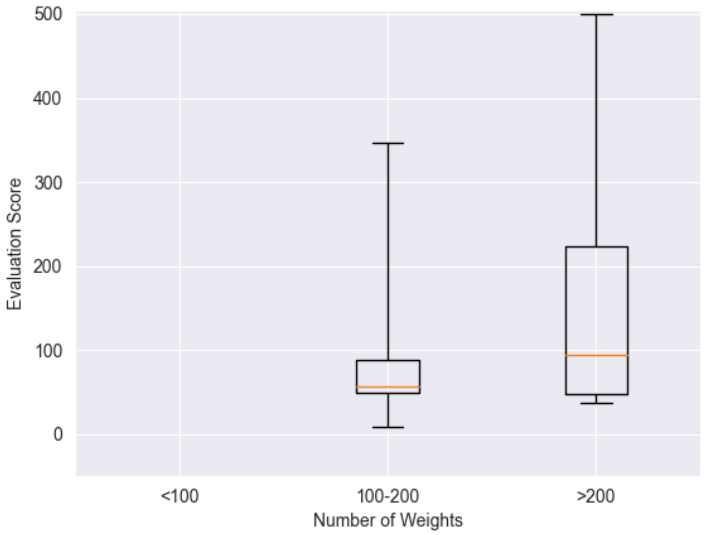


Figure 8.13: Sensitivity of quantum models with trainable entanglement in the CartPole environment to the number of weights, presented as boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.5.

8.1.4 Number of layers

In this subsection, we examine the influence of the number of layers on the performance of our models throughout the experiments. Our exploration commences with an analysis of the sensitivity of this parameter, a critical aspect of network architecture that defines the model's capacity to learn complex patterns. Following this, we focus on the progression of results over time, scrutinizing the general trend across all episodes and the peak performance depicted by the best episodes. This systematic evaluation allows us to gain a more nuanced understanding of how the depth of the model, as determined by the number of layers, contributes to the learning capability and overall performance in the CartPole environment.

Classical

The influence of the number of layers on the performance of classical agents is subtle, as demonstrated in Figure 8.14. While a minor trend indicates fewer poorly

performing agents with fewer layers, this observation is nuanced. Regardless of the number of layers, all groups can obtain a median score at the maximum possible level, illustrating that the layer count does not significantly impact the performance of classical agents in this context.

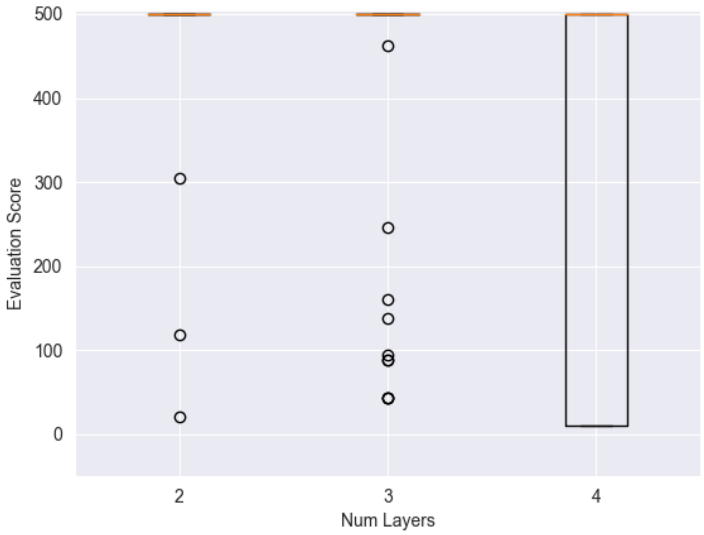


Figure 8.14: Sensitivity of classical models in the CartPole environment to different numbers of layers, as shown by boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.1.

Quantum

In the case of quantum agents, they demonstrate the capability to solve the environment utilizing various layers - namely 3, 5, 7, and 9. Nevertheless, the data indicate an optimal performance zone around 5 and 7 layers, with a noticeable performance decrease at both the lower and higher ends of the spectrum.

With fewer layers, the network's expressive power is limited. Expressivity in this context refers to the ability of a model to represent a wide variety of functions. In our case, the quantum circuit's expressivity is essentially its capacity to generate diverse output states based on different input states. A quantum circuit with low expressivity might not be able to represent complex functions, leading to a subpar performance in complex environments.

On the other hand, as the number of layers increases beyond a certain point, the optimization process becomes increasingly challenging. This can be attributed to the complexity of the quantum system as we add more layers to a quantum circuit. As a result, the search space for optimal parameters becomes more extensive and convoluted, which can make the optimization process less efficient and, in turn, adversely affect the performance of the quantum agent.

These observations underscore the importance of a balanced network configuration. Too few layers can lead to underfitting due to insufficient expressivity, while too many layers can cause optimization difficulties, potentially leading to suboptimal performance. Therefore, identifying the right balance is crucial for achieving optimal performance in quantum reinforcement learning.

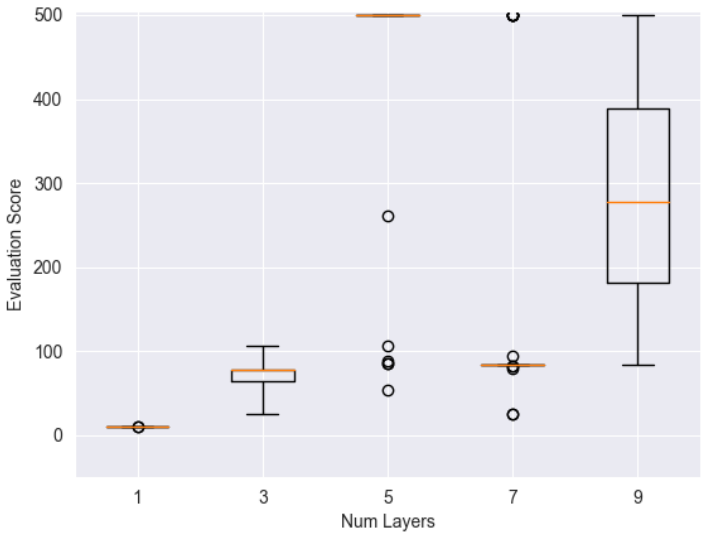


Figure 8.15: Sensitivity of quantum models in the CartPole environment to different numbers of layers, represented by boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.3.

Quantum with Trainable Entanglement

Building on the findings from experiment 3, we restricted the quantum agents with trainable entanglement to 5, 7, and 9 layers. It was intriguing to note that only the

models with 7 and 9 layers managed to solve the environment effectively. However, the median performance across all groups was notably inferior compared to the case of untrainable entanglement. This observation suggests that a mere increase in model complexity does not guarantee improved performance.

Here, it would have been particularly insightful to extend the training duration. A more complex model could reasonably require more data and more training time. After all, if untrainable entanglement proves to be the best approach, then trainable entanglement should, in theory, be able to learn this given enough time, as it has the capacity to learn $theta_entangle = pi$.

A fascinating observation emerges when we juxtapose figures 8.15 and 8.16. We can discern a shift in optimal performance towards a higher number of layers by introducing more parameters through trainable entanglement. This observation may hint at a relationship between the number of parameters and the optimal number of layers in a quantum circuit. As the model becomes more complex with the addition of trainable entanglement, the optimal network configuration shifts towards deeper architectures.

This phenomenon could be attributed to the enhanced expressive power of more layers and parameters, allowing the model to capture more complex relationships. However, the corresponding increase in complexity can also challenge the optimization process, as evidenced by the lower median scores. These findings further emphasize the importance of careful architectural choices in the design of quantum reinforcement learning models.

8.1.5 Entanglement

In this section, we examine the influence of different entanglement schemes on the performance of our quantum models. As in the previous analysis, we will first evaluate the sensitivity of this parameter, then examine the performance over time for all episodes and the best ones.

Quantum

Our results underscore the significance of entanglement in training quantum

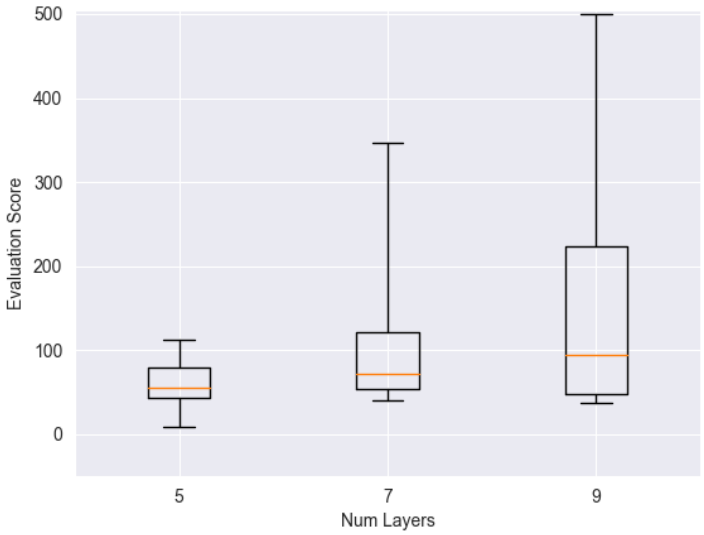


Figure 8.16: Sensitivity of quantum models with trainable entanglement in the CartPole environment to different numbers of layers, depicted by boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.5.

machine learning agents. While several groups managed to achieve a perfect score, the group's performance with no entanglement could have been better. This challenges the assertion by Wang et al. [2020] that a lack of entanglement yields superior results. In the context of purely quantum agents, the absence of entanglement markedly undermines performance, as it severely curtails the expressive capacity of the quantum circuits.

Optimal results were attained with moderate entanglement, particularly with the 'ladder' entanglement scheme, where the median score reached perfection. As performance appears to degrade with increasing entanglement towards the 'double ladder' scheme and deteriorates significantly with decreasing entanglement towards none, we hypothesize that the best results incorporating trainable entanglement would likely be achieved with a modified 'double ladder' scheme featuring reduced entanglement.

This insight further emphasizes the delicate balance required in quantum circuits

between expressiveness and complexity, with the right level of entanglement playing a critical role in achieving optimal performance.

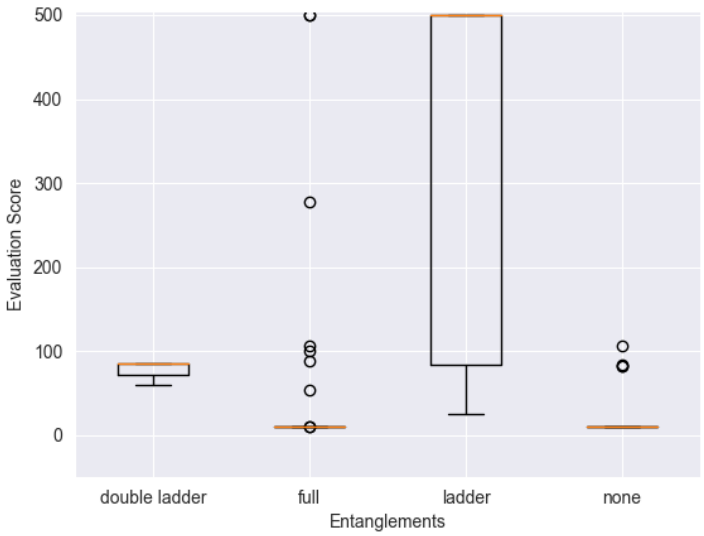


Figure 8.17: Sensitivity analysis of quantum models in the CartPole environment on different entanglement schemes, represented by boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.3.

Quantum with Trainable Entanglement

Interestingly, the highest-performing results with trainable entanglement were obtained using the 'ladder' entanglement scheme, the only one to achieve a perfect score. This suggests that the optimal degree of entanglement for the CartPole environment is less than what the 'ladder' scheme provides. Compared to the 'full' entanglement scheme, the performance shows a substantial improvement, implying that excessive entanglement may not be ideal for this environment.

In Experiment 7.5, the 'no entanglement' scheme was also included for comparison purposes with the trainable ones. Given that the absence of entanglement inherently cannot be trainable, it is a contrast. Notably, 'no entanglement' performed better overall in this experiment than in Experiment 7.3.

This improvement is likely attributable to the more refined selection of other hyperparameters based on insights gleaned from previous experiments.

These results underline the importance of careful hyperparameter tuning in quantum machine learning and the nuanced impact that the degree of entanglement can have on the overall performance of a quantum agent in a specific environment.

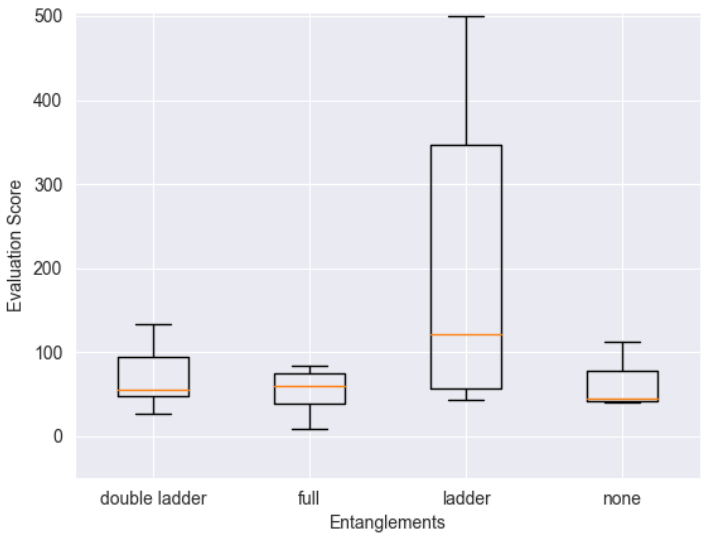


Figure 8.18: Sensitivity analysis of quantum models with trainable entanglement in the CartPole environment on different entanglement schemes, depicted by boxplots. For a detailed understanding of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.5.

8.1.6 Multihyperparemer sensitivity

This section examines the effect and sensitivity of changing two hyperparameters at once. The numbers displayed are the mean of the best score from every run.

Classical By observing the learning rate in relation to the number of layers, it is evident that lower learning rates provide good results overall. When the learning rate increases with the number of layers, the agents fail to optimize, resulting in poor performance. When analyzing the number of layers in unison with the

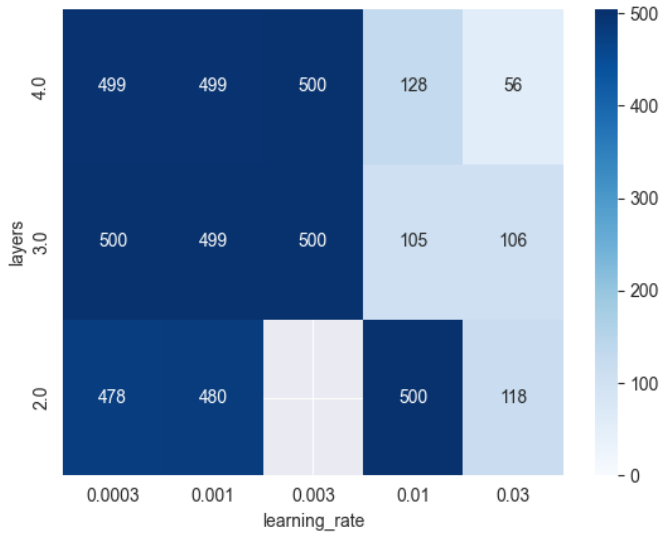


Figure 8.19: Performance heatmap for classical models in the CartPole environment across different numbers of layers and learning rates. The experimental setup can be found in Experiment 7.1.

number of neurons in each layer, there is no obvious difference in performance. Most combinations result in well-working agents, besides an unlucky outlier at the cross-section between four layers and 16 neurons. **Quantum** Figure 8.21 shows that a narrow combination of learning rate and layers works well for quantum models. The data hints at a diagonal line from a high number of layers and a low learning rate towards a low number of layers and a high learning rate. Figure 8.22 illustrates that utilizing either a single layer or no entanglement scheme significantly underperforms in all scenarios, while most other combinations achieve

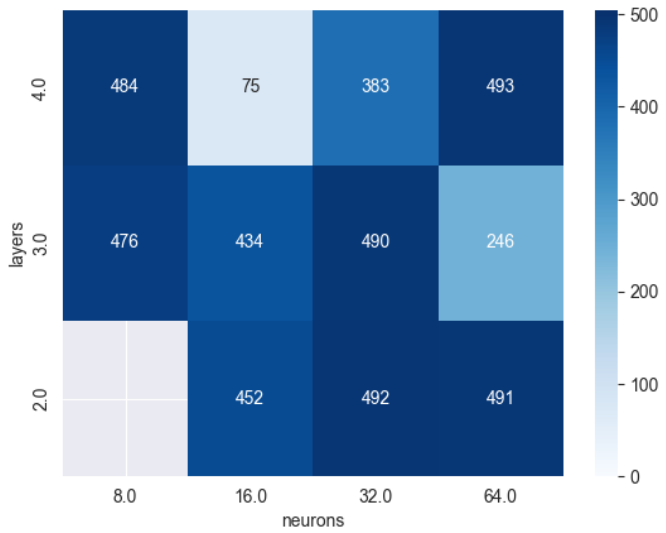


Figure 8.20: Performance heatmap for classical models in the CartPole environment across different numbers of layers and neurons. The experimental setup can be found in Experiment 7.1.

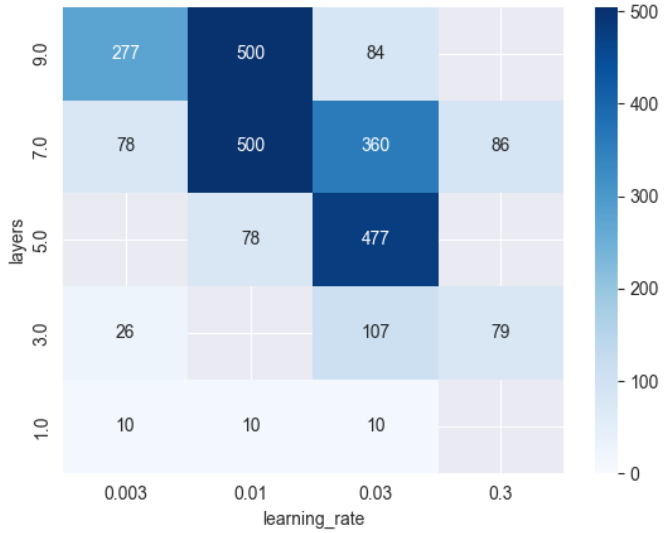


Figure 8.21: Performance heatmap for quantum models in the CartPole environment across different numbers of layers and learning rates. The experimental setup can be found in Experiment 7.3.

satisfactory scores. Importantly, the ladder entanglement scheme stands out, achieving near-optimal and optimal scores. Figure 8.23 displays the performance

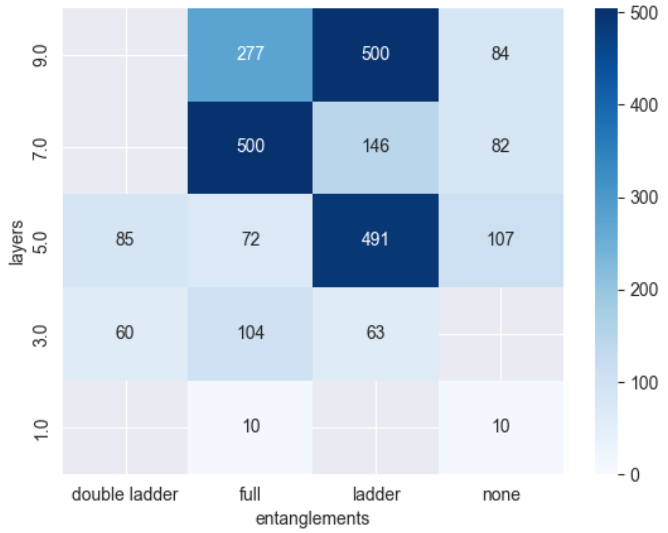


Figure 8.22: Performance heatmap for quantum models in the CartPole environment across different numbers of layers and entanglement schemes. The experimental setup can be found in Experiment 7.3.

sensitivities of the entanglement scheme and learning rate. Only a select set of combinations yield optimal results. Specifically, the ladder entanglement scheme combined with a moderate learning rate provides the most favorable outcomes. Utilizing an extremely low learning rate or a scheme with no entanglement generally results in suboptimal performance. **Quantum with trainable entanglement** Figure 8.24 examines the model's sensitivity to changes in the number of layers, given that a single learning rate is employed. This visualization clarifies that increasing the number of layers correlates positively with improved performance. This suggests that the expression of trainable entanglement might necessitate more layers for optimal performance. However, further research and experiments are needed to validate this hypothesis conclusively.

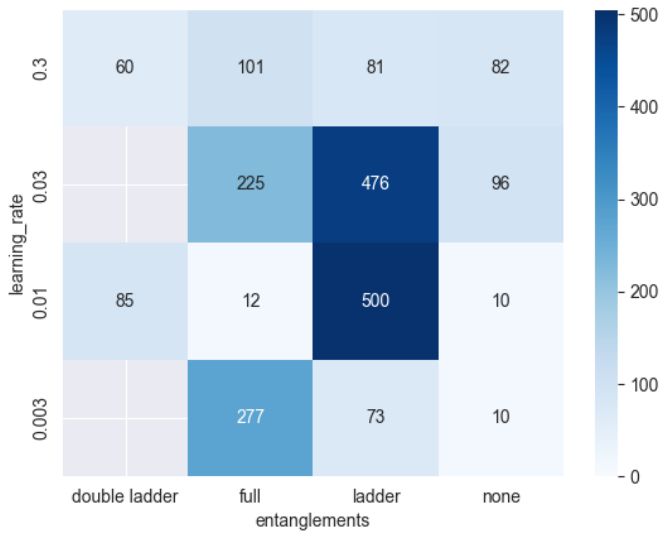


Figure 8.23: Performance heatmap for quantum models in the CartPole environment across different learning rates and entanglement schemes. The experimental setup can be found in Experiment 7.3.

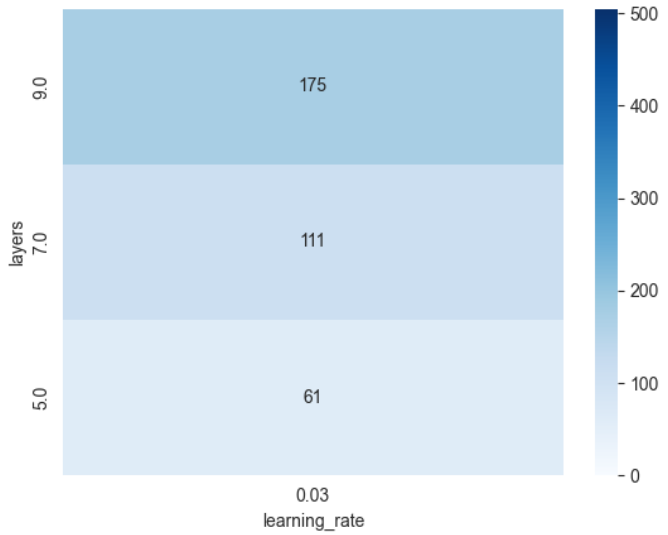


Figure 8.24: Performance heatmap for quantum models with trainable entanglement in the CartPole environment across different numbers of layers and learning rates. The experimental setup can be found in Experiment 7.5.

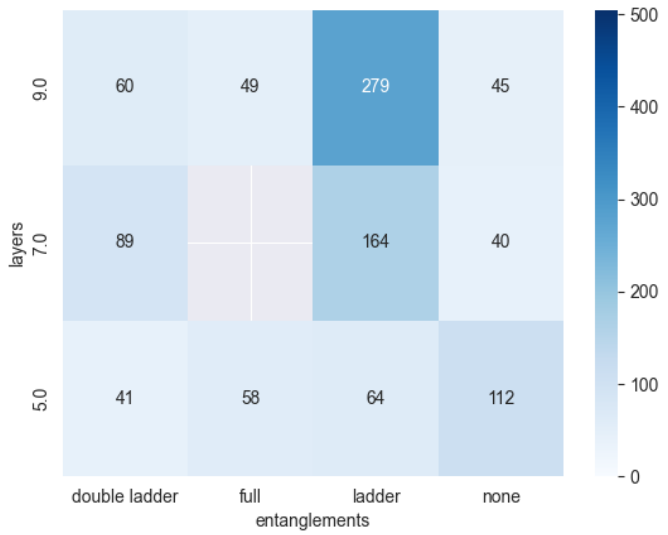


Figure 8.25: Performance heatmap for quantum models with trainable entanglement in the CartPole environment across different numbers of layers and entanglement schemes. The experimental setup can be found in Experiment 7.5.

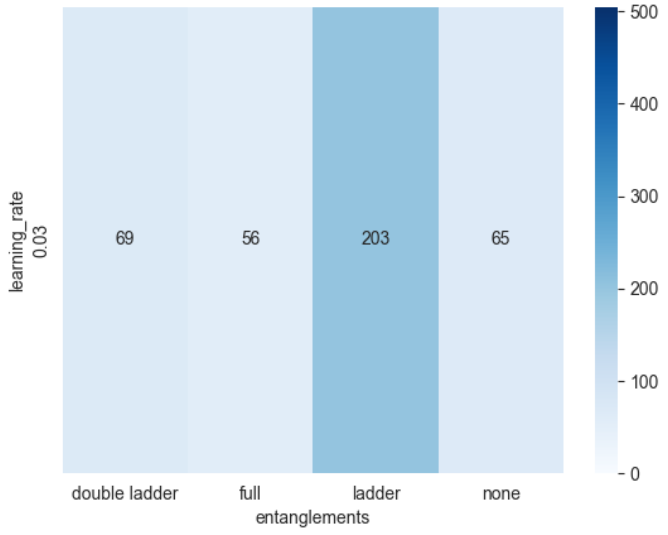


Figure 8.26: Performance heatmap for quantum models with trainable entanglement in the CartPole environment across different learning rates and entanglement schemes. The experimental setup can be found in Experiment 7.5.

In summary, classical and quantum agents have proven their ability to resolve the given environment. Despite exhibiting superior stability and adaptability with less than optimal hyperparameters, classical agents are matched in performance by quantum agents. These quantum agents achieve comparable results with fewer parameters and typically within a shorter timeframe.

8.2 LunarLander

In this section, we delve into the results from Experiments 2, 4, and 6, where we trained classical and quantum agents on the Lunar Lander environment. Our approach encompassed a random search over a range of hyperparameters to pinpoint the optimal configurations for both classical and quantum models. The experiments section elaborates on the specific parameters utilized in these experiments.

The classical models function as the reference point, enabling us to juxtapose the performance of the quantum models against that of optimized classical agents tailored for this particular problem. It is imperative to note that the best possible score in implementing the Lunar Lander environment is around 200. In contrast, the lowest possible score is a huge negative number but is limited to a minimum of -200 in our results.

During the training, the models were executed for up to 100 timesteps, succeeded by five evaluation episodes with a cap of 1000 timesteps each. This rigorous evaluation approach ensures credible results as the actions, observations, and raw model outputs are retained for all episodes. It is worth noting that any score above 100 is deemed a successful solution. However, we aim to ascertain which techniques can maintain the lunar lander's stable landing for extended durations.

Throughout this section, we utilize boxplots to represent our results. The elements of these boxplots are as follows: The box itself denotes the interquartile range (IQR) that contains the middle 50% of the scores around the median. An orange line within the box represents the median score, which bisects the data into upper and lower halves. The whiskers from the box indicate the highest and lowest

values within the first and fourth quartiles, barring any outliers. Outliers, represented by circles, are those data points significantly distinct from the others. This standardized visual representation aids in effectively understanding the spread and skewness of our data.

8.2.1 Accumulated results

All Classical

Upon examining the performance of classical agents in the Lunar Lander environment in figure 8.27, it is apparent that while a handful of runs manage to achieve respectable scores, the overwhelming majority fall short significantly. The bulk of the results are concentrated around the -150 score mark. Policies within this point range typically hover, maintaining their position above the ground without achieving a successful landing. The results procured are notably subpar, which we conjecture may be attributed to the limitations imposed by the smaller models.

Previous research has demonstrated that the Lunar Lander environment can be solved using models with two layers of 64 nodes [Gadgil et al., 2020], a configuration that falls within the scope of our experiments. However, the most reliable and optimal solutions tend to emerge when the number of nodes extends beyond the confines of our current experimental setup. Thus, exploring larger, more complex models in future attempts to solve this environment might be beneficial.

All Quantum Conversely, the quantum agents exhibit a critical failure in the Lunar Lander environment, as depicted in figure 8.28. Despite this, a discernible upward trend suggests that a performance improvement might be possible with extended training time. It is crucial to remember that the number of parameters in our quantum models is substantially lower compared to the classical approaches that have previously managed to solve this environment (approximately 200 in our quantum models versus 18,180 in previous successful classical models).

Given the exceptionally subpar performance of the quantum agents, dissecting the minute variations in performance yields limited insight, as every configuration

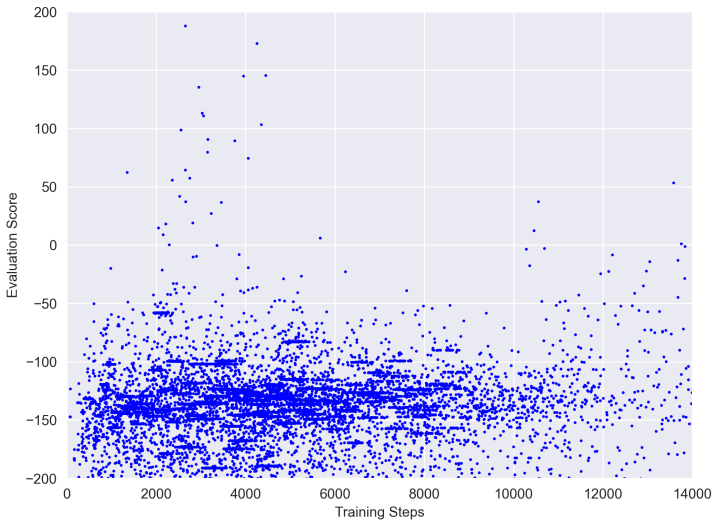


Figure 8.27: Performance over all episodes for classical models in the LunarLander environment. The experimental setup can be found in Experiment 7.2.

ultimately results in critical failure. Consequently, the primary takeaway from this outcome is understanding the areas where quantum agents currently fall short and the necessity for further exploration and development in quantum approaches to reinforcement learning problems of this complexity. c **All Quantum with Trainable Entanglement**

The quantum agents with trainable entanglement also exhibit a striking failure in the Lunar Lander environment. However, an intriguing trend is noticeable in these results. Over time, there is a reduction in the number of poorly performing agents, as shown in figure 8.29. This suggests that these agents may gradually improve their understanding of the environment, potentially leading to enhanced performance with prolonged training time.

The quantum agents with trainable entanglement possess a greater expressivity, which could be the reason for this slight improvement in performance over time. The expressive power of quantum circuits can be drastically increased by incorporating entanglement, allowing the model to learn more complex representations of the environment.

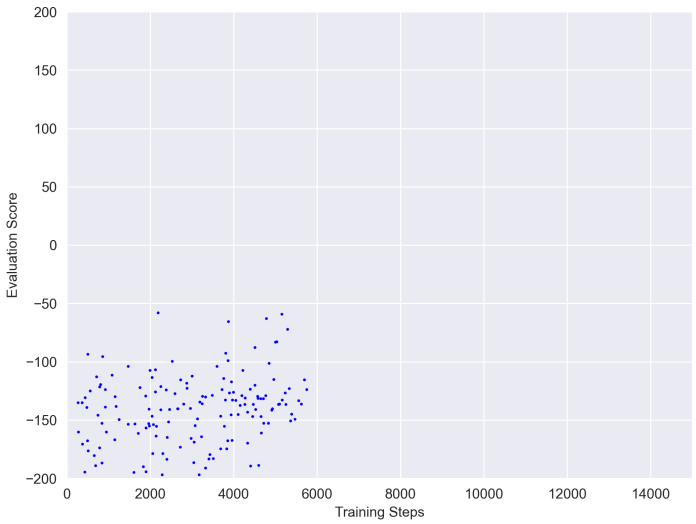


Figure 8.28: Performance over all episodes for quantum models in the LunarLander environment. The experimental setup can be found in Experiment 7.4.

Despite this, the overall performance of quantum agents with trainable entanglement still needs to be satisfactory, emphasizing the need for further experimentation with longer training time and larger models. The results underline the challenges that quantum reinforcement learning faces when tackling more complex environments, particularly those requiring more parameters for successful navigation.

Best Classical

A distinct trend becomes evident when evaluating the performance of the best classical models in the Lunar Lander environment. By shifting our perspective from individual runs to the average evaluation score, we notice a marked decrease in performance, as depicted in figure 8.30.

This phenomenon implies that while specific model configurations may occasionally produce high-scoring runs, these models' overall consistency and reliability are lacking. This highlights one of the critical challenges in reinforcement learning - achieving stable and repeatable performance across different runs and conditions.

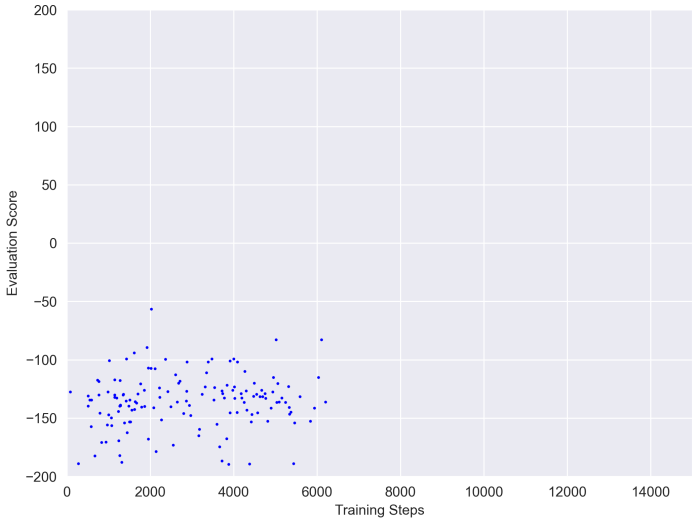


Figure 8.29: Performance over all episodes for quantum models with trainable entanglement in the LunarLander environment. The experimental setup can be found in Experiment 7.6.

It underscores the importance of not only identifying configurations that can produce peak performances but also those that can maintain a high level of performance consistently. In other words, the robustness of a model across multiple runs and varying conditions is as crucial as its ability to achieve high scores in individual instances.

In the case of the Lunar Lander environment, despite the occasional high-scoring runs, the classical models, on average, need to provide a satisfactory solution. This paints a clear picture of the complexity of this environment and the challenges it poses to classical and quantum reinforcement learning techniques. It emphasizes the need for more robust and consistent models tackling such complex tasks.

Best Quantum

In assessing the performance of the best quantum models within the Lunar Lander environment, we are met with consistent underperformance, as shown in figure 8.31. The quantum agents continue to perform poorly despite focusing on the

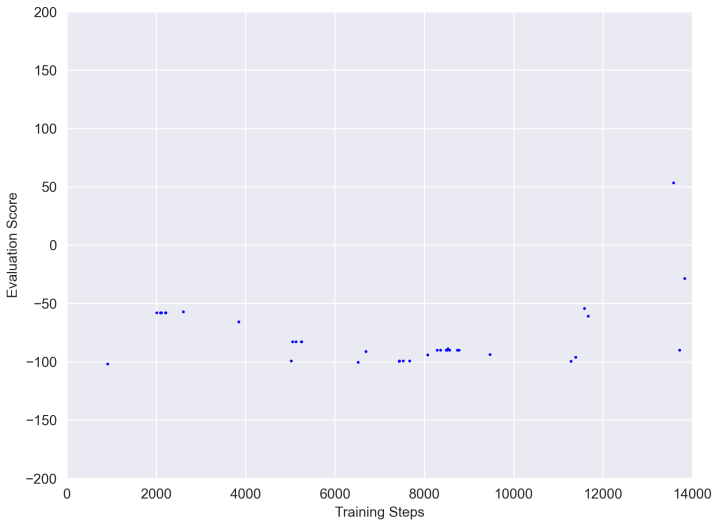


Figure 8.30: Performance for the best episodes for classical models in the LunarLander environment. The experimental setup can be found in Experiment 7.2.

best-performing instances.

This indicates that the quantum agents, even in their most favorable configurations, cannot match the performance of classical models in this complex environment. It highlights the limitations of shallow quantum agents, particularly when handling tasks of higher complexity with relatively few parameters.

Best Quantum with Trainable Entanglement

As illustrated in figure 8.32, the quantum agents with trainable entanglement also critically fail the Lunar Lander environment.

Although including trainable entanglement offers a greater degree of expressiveness to the quantum network, it needs to overcome the inherent limitations of the quantum agents in this high-complexity task. These observations highlight the ongoing challenges quantum reinforcement learning techniques face in tackling complex environments. They also underscore the importance of continued research and development in this field, as these techniques may hold the potential to excel in certain problem domains, given the right improvements and advancements.

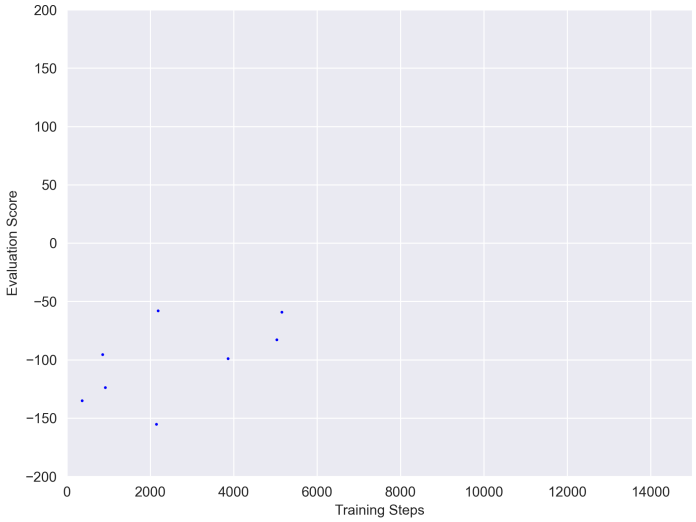


Figure 8.31: Performance for the best episodes for quantum models in the LunarLander environment. The experimental setup can be found in Experiment 7.4.

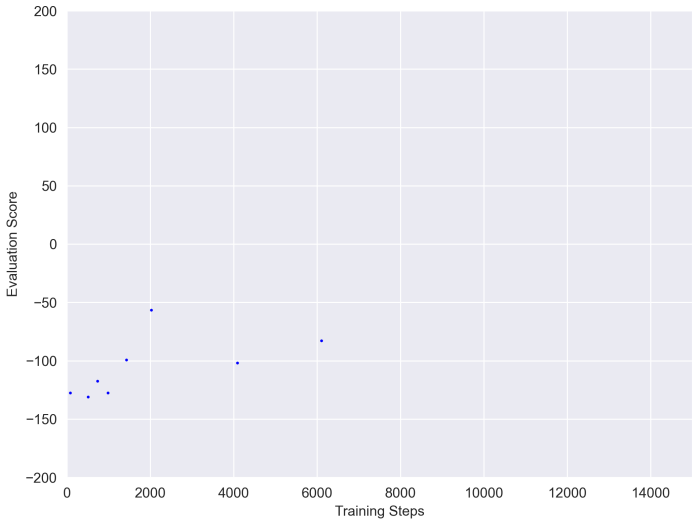


Figure 8.32: Performance for the best episodes for quantum models with trainable entanglement in the LunarLander environment. The experimental setup can be found in Experiment 7.6.

Comparison

When we juxtapose classical, quantum, and quantum performances with trainable entanglement models within the Lunar Lander environment, we observe a sobering reality - all three models critically fail to solve the task, as illustrated in figure 8.33.

Despite the classical model's inherent advantage in terms of parameter count, it does not demonstrate any significant performance improvement. This brings into sharp focus the complexity and difficulty level of the Lunar Lander environment.

The quantum models, both with and without trainable entanglement, also need help to make a dent in the task. This observation underlines the limitations of current quantum reinforcement learning techniques in handling high-complexity environments, as it is challenging to train them effectively for an extended time. The introduction of trainable entanglement, while theoretically providing a boost in the expressiveness of the quantum network, fails to translate into any practical improvement in performance, as the models still need to be larger.

These findings drive home the point that despite the advancements in quantum and classical reinforcement learning, much work is needed to equip these agents better to handle complex tasks. Future work in this field should focus on longer-running experiments with larger models.

As little insight can be drawn from the rest of the lunar lander results, the remainder is in the appendix 10.1.

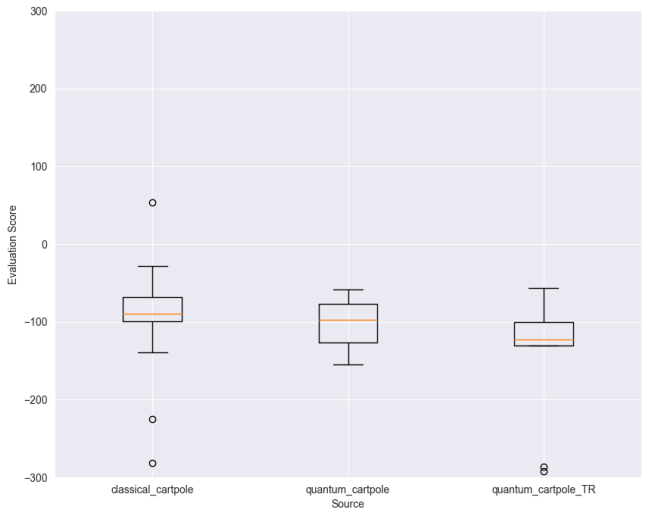


Figure 8.33: Distribution of best scores achieved by classical DQL, quantum DQL, and quantum DQL with trainable entanglement in the LunarLander environment, illustrated as boxplots. For a detailed description of the boxplot elements, refer to the introduction of this section. The experimental setups can be found in Experiments 7.1, 7.3, and 7.5, respectively.

9 Conclusion

This thesis examined classical and quantum implementations of deep Q-learning, emphasizing hyperparameters' sensitivity and trainable entanglement's role in quantum reinforcement learning. The aim was to investigate whether quantum models could offer an efficient alternative to classical ones in solving the Cart Pole and Lunar Lander environments.

Our exploration revealed that quantum agents could perform comparably to classical counterparts despite exhibiting less stability, using significantly fewer parameters. This highlights quantum computing's potential efficiency in reinforcement learning, although careful optimization and judicious hyperparameter selection are crucial for the success of these models.

While the scalability of quantum agents with more parameters remains uncertain, future research could benefit from examining the implications of more intensive training and larger models. Despite the failure of our quantum agents in the more challenging Lunar Lander environment, given sufficient training, wider and deeper quantum models or expanded networks could solve this environment effectively.

Interestingly, other studies claiming the successful application of quantum models to the Lunar Lander problem incorporate substantial classical components. In contrast, our approach prioritizes a primarily quantum composition, pointing to the unexplored potential of quantum computing in reinforcement learning.

While using actual quantum computers addresses the bottleneck encountered when simulating quantum computing on classical machines, today's quantum technology presents challenges. Noise accumulation in large and deep quantum circuits can distort data over time, rendering outputs useless. Thus, reducing noise in quantum computers is a significant hurdle to practically implementing quantum computing.

The findings of this thesis extend beyond the specific scenarios explored, holding promise for broader applications. With their ability to estimate exponentially larger search spaces than classical techniques, quantum circuits could revolutionize the

representation and optimization of latent spaces in classical neural networks.

In conclusion, this thesis reveals both the promise and challenges of quantum computing in reinforcement learning. It highlights the potential efficiency of quantum computing while shedding light on stability and scalability issues. Future research should delve deeper into these areas, addressing the associated challenges and further exploring the potential contributions of quantum computing to reinforcement learning.

References

Scott Aaronson and Alex Arkhipov. The boson sampling problem. *Theory of Computing*, 9(1):143–252, 2013.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: A system for large-scale machine learning. *Operating Systems Review*, 49(1):73–77, 2016.

Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

Ville Bergholm, Josh Izaac, Maria Schuld, Ilya Sinayskiy, Mark Walters, Daniel Willsch, Leonid Wossnig, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

Chris M Bishop. Neural networks and their applications. *Review of scientific instruments*, 65(6):1803–1832, 1994.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Max Born. Quantenmechanik der stoßvorgänge. *Zeitschrift für physik*, 38(11-12): 803–827, 1926a.

Max Born. Quantum mechanics of collision phenomena. *Zeitschrift für Physik*, 38 (11-12):803–827, 1926b.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9): 625–644, 2021.

François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 192–204, 2015.

Isaac L. Chuang and Michael A. Nielsen. *Quantum Computation and Quantum Information*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0521635039. doi: 10.1017/CBO9780511976667.

Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloe. *Quantum mechanics*. John Wiley & Sons, 1977.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

Marcus Cramer and Martin B Plenio. An efficient quantum algorithm for finding hidden structures. *Nature Communications*, 1(1):149, 2010.

David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

Michel H Devoret and Robert J Schoelkopf. Superconducting circuits for quantum information: an outlook. *Science*, 339(6124):1169–1174, 2013.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47 (10):777–780, 1935.

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014a.

Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Hartmut Neven. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1412.6062*, 2014b.

Richard P Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.

Soham Gadgil, Yunfeng Xin, and Chengzhe Xu. Solving the lunar lander problem under uncertainty using reinforcement learning. In *2020 SoutheastCon*, volume 2, pages 1–8. IEEE, 2020.

Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman, 1979.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323. PMLR, 2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. *Nature*, 521 (7553):436–444, 2016.

Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, James Lockhart, Vid Stojevic, Andrew G Green, and Simone Severini. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3(1-2):214, 2019.

David J. Griffiths. *Introduction to quantum mechanics*. Cambridge University Press, 2018.

Lov K Grover. A fast quantum mechanical algorithm for database search. *STOC*, 28(2):212–219, 1996.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):1–42, 2018.

Werner Heisenberg. Uber den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift fur Physik*, 43(3-4):172–198, 1927.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Zoe Holmes, Marco Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *Quantum*, 5:391, 2021.

Ryszard Horodecki, Pawel Horodecki, Michal Horodecki, and Karol Horodecki. Quantum entanglement. *Reviews of Modern Physics*, 81(2):865, 2009.

Sofiene Jerbi, Casper Gyurik, Simon Marshall, Hans Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28362–28375, 2021.

Swarnadeep Khatri, Ryan LaRose, Marco Pistoia, Stuart Hadfield, Lukasz Cincio, and Patrick J Coles. Quantum-assisted quantum compiling. *Quantum Science and Technology*, 5(3):034008, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Yunseok Kwak, Won Joon Yun, Soyi Jung, Jong-Kook Kim, and Joongheon Kim. Introduction to quantum reinforcement learning: Theory and pennylane-based

implementation. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 416–420. IEEE, 2021.

Yann LeCun, L'eon Bottou, Genevieve B Orr, and Klaus-Robert M"uller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 9–48, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

Long-Ji Lin. Reinforcement learning for robots using neural networks. In *Advances in neural information processing systems*, pages 1008–1014, 1993.

Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):4812, 2018.

Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2010.

N David Mermin. *Quantum computer science: An introduction*. Cambridge University Press, 2007.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. *ICML*, 27:807–814, 2010.

Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2000.

Travis E Oliphant. A guide to numpy. *Trelgol Publishing USA*, 2006.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

Matteo G A Paris and Jaroslav Rehacek. Quantum state tomography. *International Journal of Quantum Information*, 2(04):125–137, 2004.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

J. J. Sakurai and Jim Napolitano. *Modern quantum mechanics*. Addison-Wesley, 2014.

Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.

Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, 2021.

Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5): 1484–1509, 1999.

Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *Quantum*, 6:720, 2022.

Andrew Skolik, Jules Meyer, Victor Bhaskara, Marco Cerezo, and Patrick J Coles. Layerwise learning for quantum neural networks. *Quantum*, 5:441, 2021.

Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Barbara M Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2):307, 2015.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *Coursera: Neural networks for machine learning*, volume 4, pages 26–31, 2012.

Stéfan Van Der Walt, S Chris Colbert, and Gaël Varoquaux. Numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

Guido van Rossum. Python tutorial, 1995.

Zhikang T Wang, Yuto Ashida, and Masahito Ueda. Deep reinforcement learning control of quantum cartpoles. *Physical Review Letters*, 125(10):100401, 2020.

Nathan Wiebe, Dominic W Berry, and Peter Høyer. Quantum simulation of fermionic hamiltonians with a polynomial number of oracle queries. *Journal of Physics A: Mathematical and Theoretical*, 45(44):445308, 2012.

Jens Wienke. *Quantum computing: a short course from theory to experiment*. Wiley-VCH, 2007.

Wikipedia. Bloch sphere, 2023. URL https://en.wikipedia.org/wiki/Bloch_sphere. Accessed: 2023-01-06.

Kristian Wold. Parameterized quantum circuits for machine learning. Master's thesis, University of Oslo, 2021.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.

Wojciech Hubert Zurek. Decoherence, einselection, and the quantum origins of the classical. *Reviews of Modern Physics*, 75(3):715, 2003.

10 Appendix

10.1 Lunar Lander Results

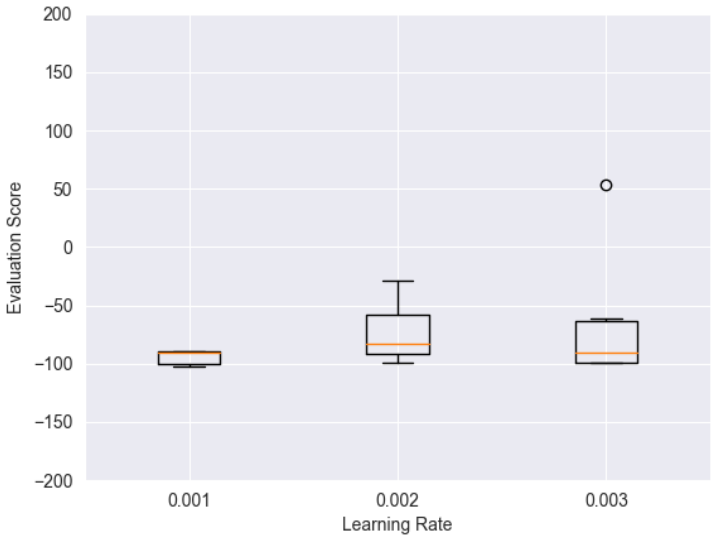


Figure 10.1: Sensitivity of classical models in the LunarLander environment to different learning rates, illustrated as boxplots. For a detailed description of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.1.

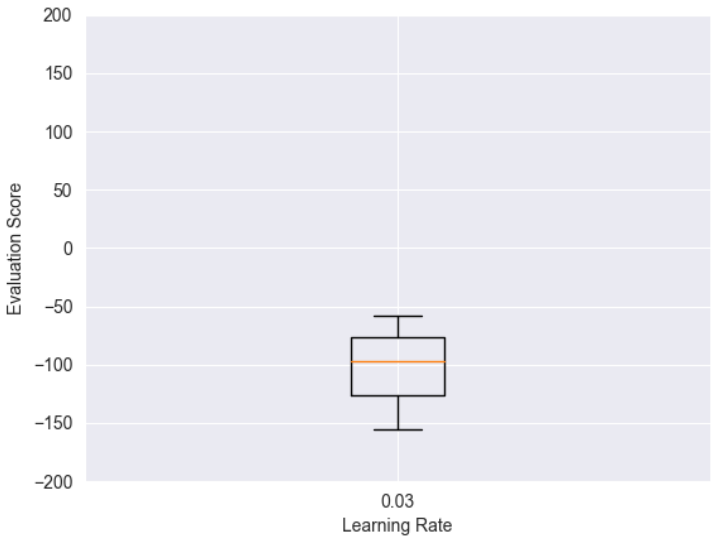


Figure 10.2: Sensitivity of quantum models in the LunarLander environment to different learning rates, illustrated as boxplots. For a detailed description of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.3.

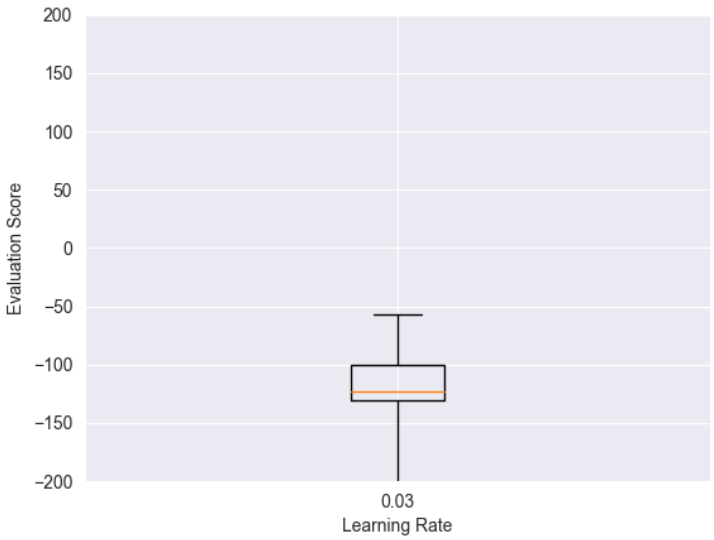


Figure 10.3: Sensitivity of quantum models with trainable entanglement in the LunarLander environment to different learning rates, illustrated as boxplots. For a detailed description of the boxplot elements, refer to the introduction of this section. The experimental setup can be found in Experiment 7.5.

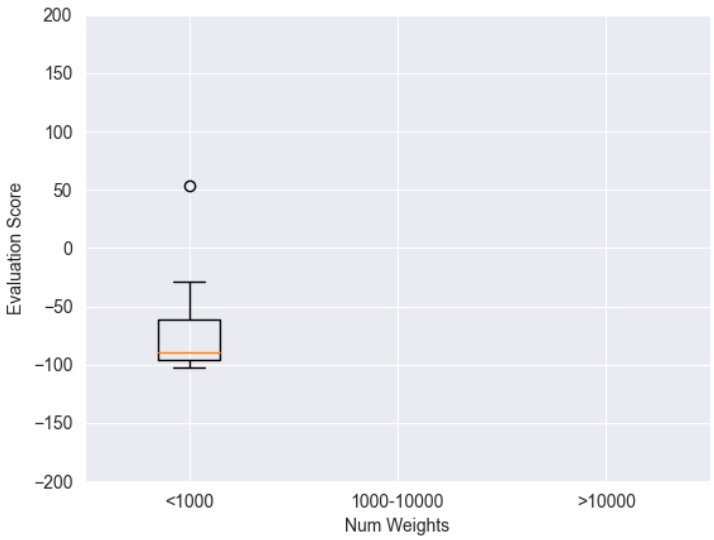


Figure 10.4: Sensitivity of classical models in the LunarLander environment to the number of weights. The x-axis denotes the number of weights in the neural network, and the y-axis denotes the corresponding score achieved by the model. The box plots indicate the distribution of scores for different numbers of weights.

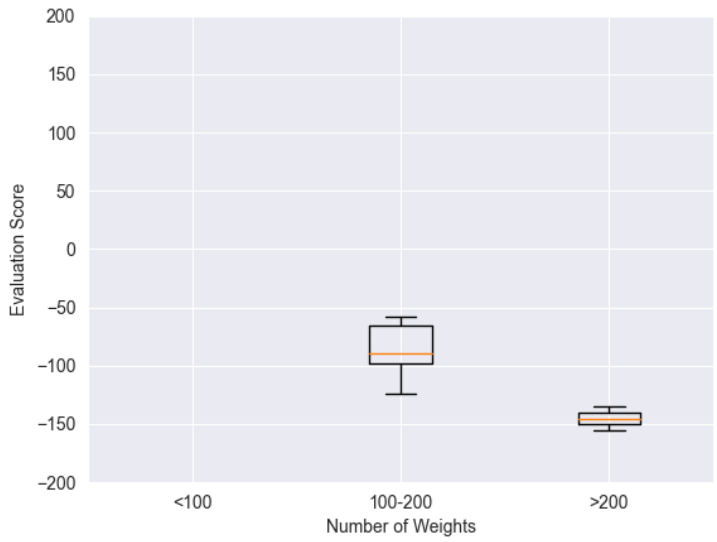


Figure 10.5: Sensitivity of quantum models in the LunarLander environment to the number of weights. The x-axis denotes the number of weights in the quantum neural network, and the y-axis denotes the corresponding score achieved by the model. The box plots indicate the distribution of scores for different numbers of weights.

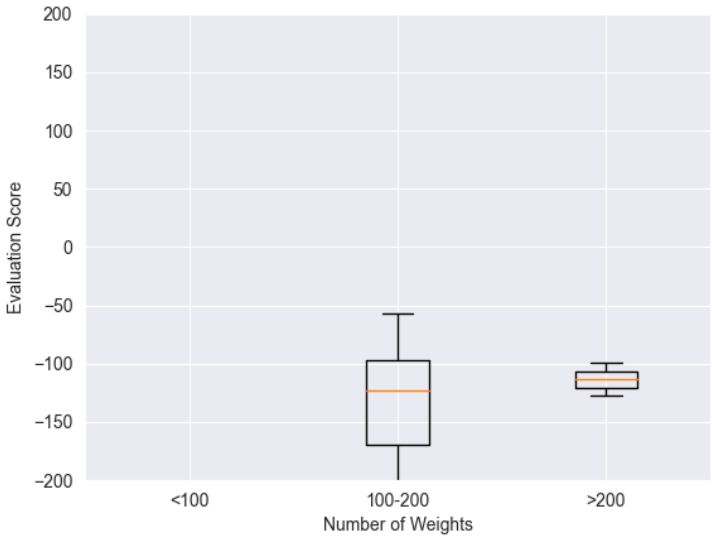


Figure 10.6: Sensitivity of quantum models with trainable entanglement in the LunarLander environment to the number of weights. The x-axis denotes the number of weights in the quantum neural network, and the y-axis denotes the corresponding score achieved by the model. The box plots indicate the distribution of scores for different numbers of weights.

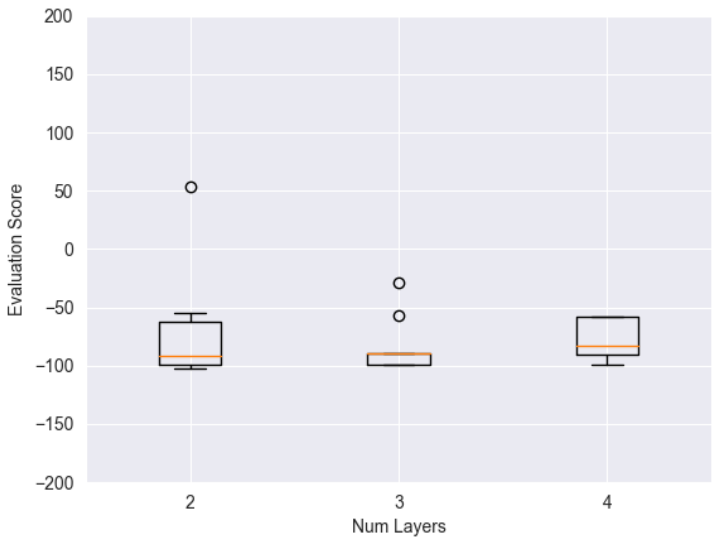


Figure 10.7: Sensitivity of classical models in the LunarLander environment to different numbers of layers. The x-axis denotes the number of layers in the neural network, and the y-axis denotes the corresponding score achieved by the model. The box plots indicate the distribution of scores for different numbers of layers. The experimental setup can be found in Experiment 7.2.

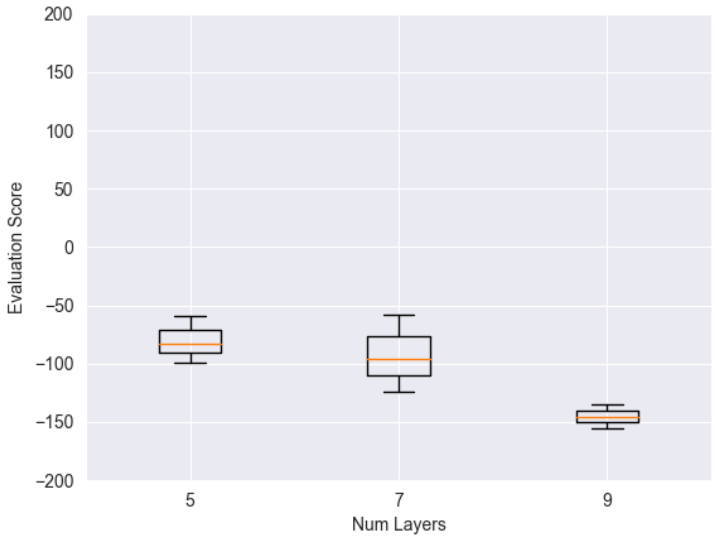


Figure 10.8: Sensitivity of quantum models in the LunarLander environment to different numbers of layers. The x-axis denotes the number of layers in the quantum neural network, and the y-axis denotes the corresponding score achieved by the model. The box plots indicate the distribution of scores for different numbers of layers. The experimental setup can be found in Experiment 7.4.

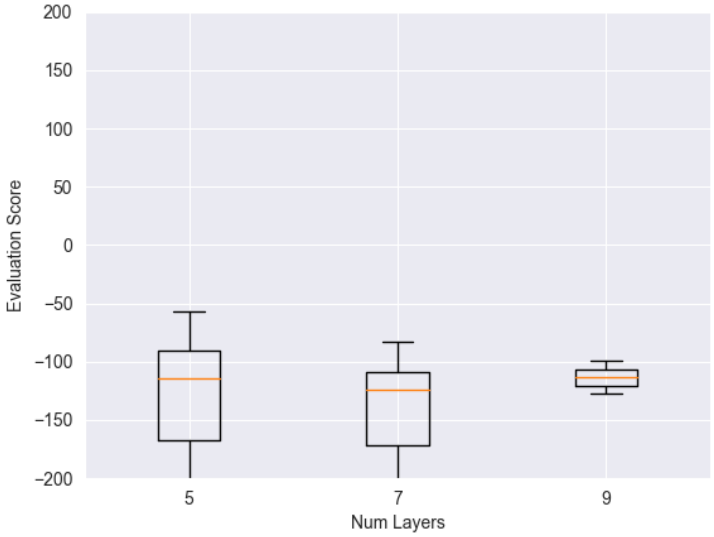


Figure 10.9: Sensitivity of quantum models with trainable entanglement in the LunarLander environment to different numbers of layers. The x-axis denotes the number of layers in the quantum neural network, and the y-axis denotes the corresponding score achieved by the model. The box plots indicate the distribution of scores for different numbers of layers. The experimental setup can be found in Experiment 7.6.

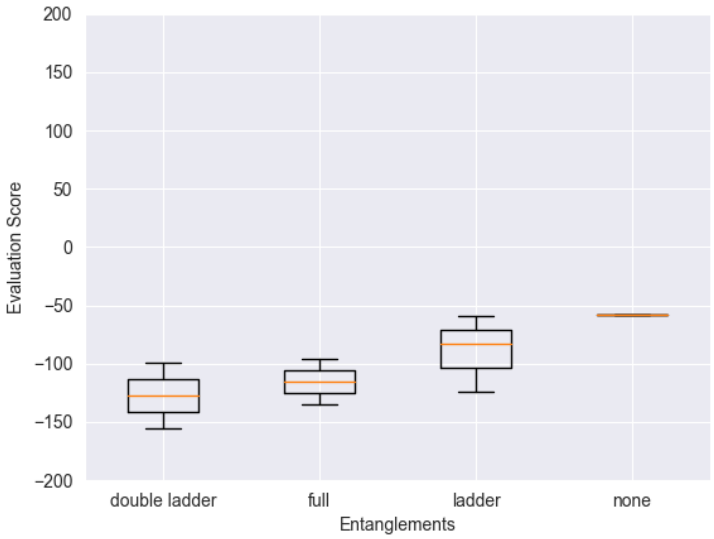


Figure 10.10: Sensitivity analysis of quantum models in the LunarLander environment on different entanglement schemes. The experimental setup can be found in Experiment 7.4.

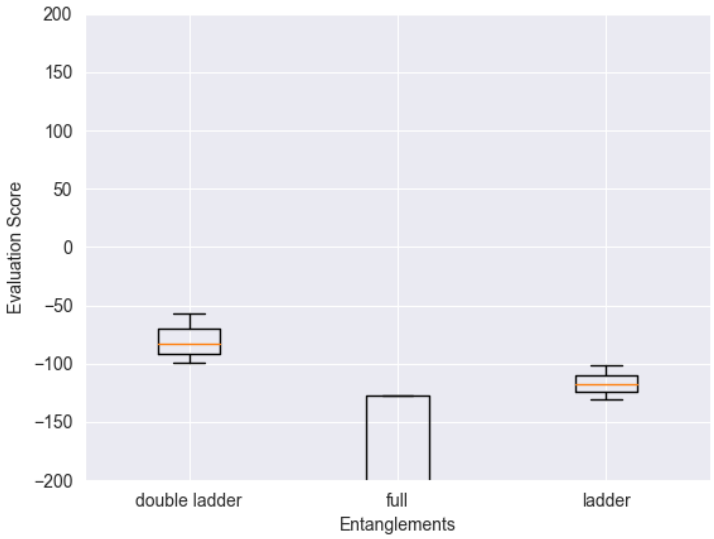


Figure 10.11: Sensitivity analysis of quantum models with trainable entanglement in the LunarLander environment on different entanglement schemes. The experimental setup can be found in Experiment 7.6.

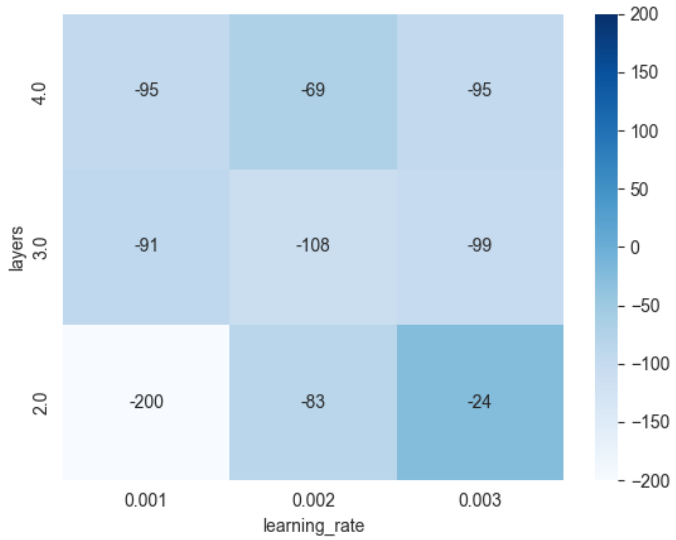


Figure 10.12: Heatmap of performance for classical models in the LunarLander environment across different numbers of layers and learning rates. The experimental setup can be found in Experiment 7.2.

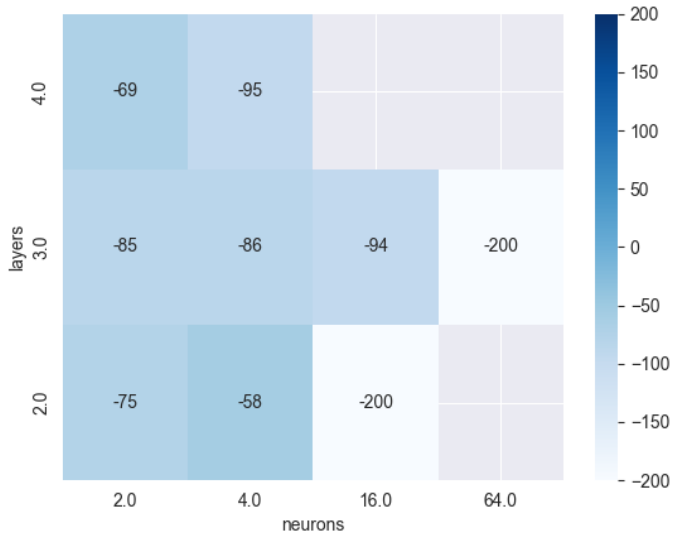


Figure 10.13: Heatmap of performance for classical models in the LunarLander environment across different numbers of layers and neurons. The experimental setup can be found in Experiment 7.2.

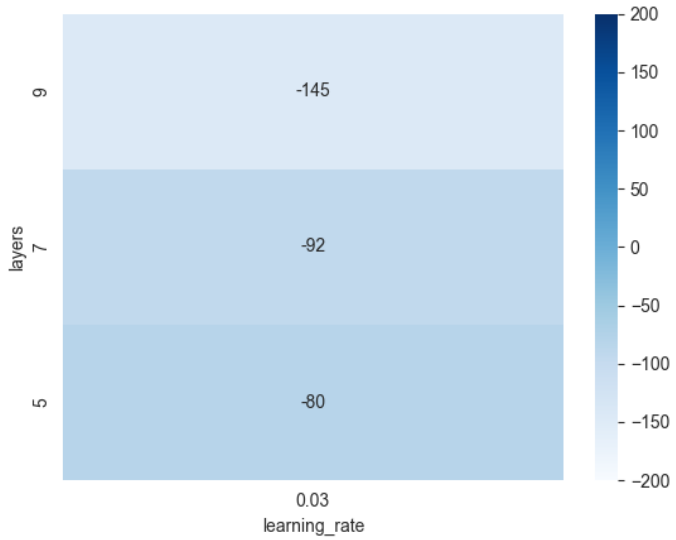


Figure 10.14: Heatmap of performance for quantum models in the LunarLander environment across different numbers of layers and learning rates. The experimental setup can be found in Experiment 7.4.

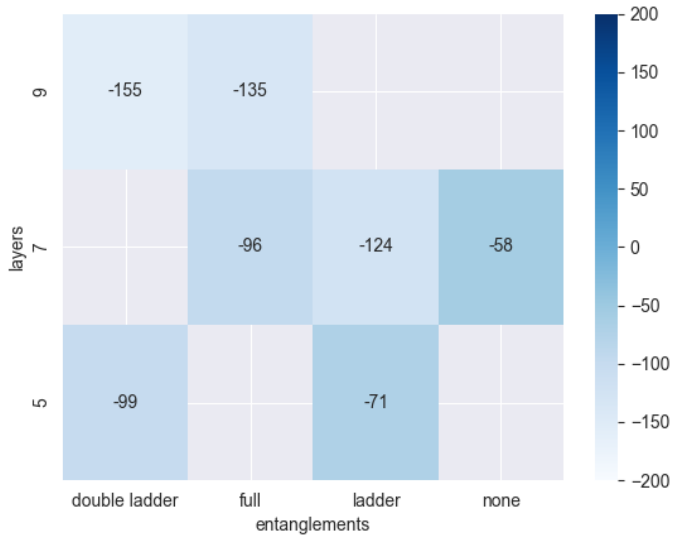


Figure 10.15: Heatmap of performance for quantum models in the LunarLander environment across different numbers of layers and entanglements. The experimental setup can be found in Experiment 7.4.

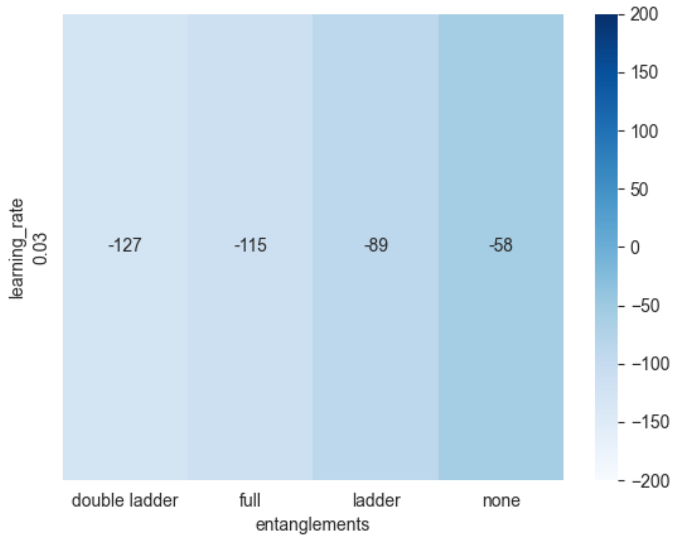


Figure 10.16: Heatmap of performance for quantum models in the LunarLander environment across different learning rates and entanglements. The experimental setup can be found in Experiment 7.4.

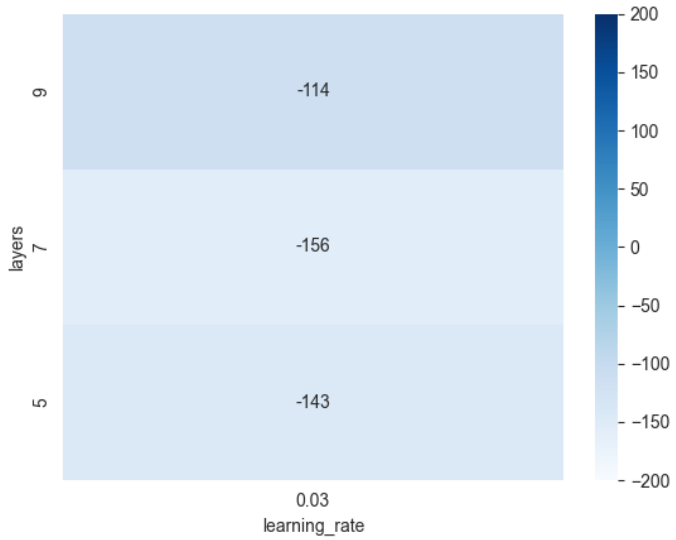


Figure 10.17: Heatmap of performance for quantum models with trainable entanglement in the LunarLander environment across different numbers of layers and learning rates. The experimental setup can be found in Experiment 7.6.

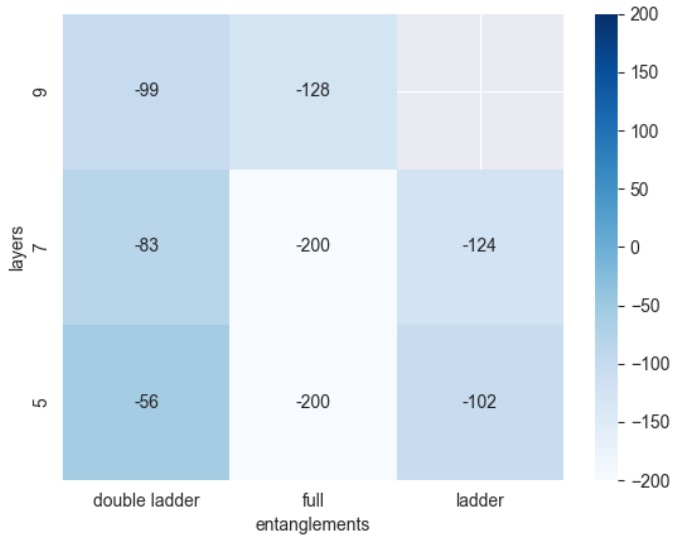


Figure 10.18: Heatmap of performance for quantum models with trainable entanglement in the LunarLander environment across different numbers of layers and entanglements. The experimental setup can be found in Experiment 7.6.

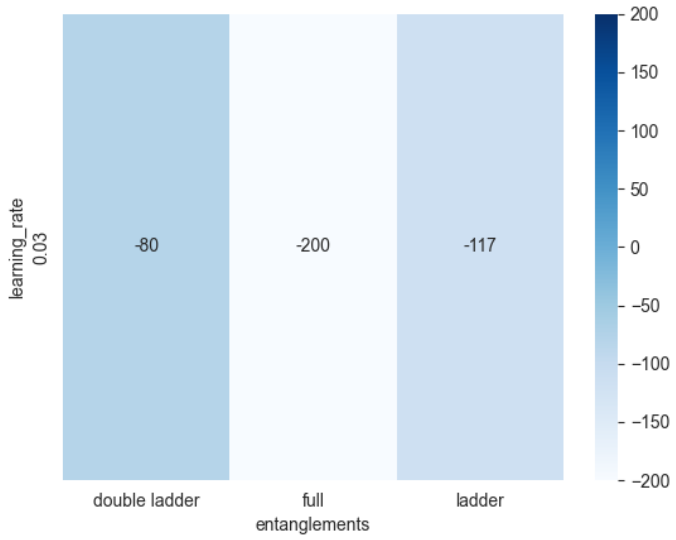


Figure 10.19: Heatmap of performance for quantum models with trainable entanglement in the LunarLander environment across different learning rates and entanglements. The experimental setup can be found in Experiment 7.6.