Universiteit Leiden
The Netherlands

Opleiding Informatica

The impact of mid-circuit measurements on quantum reinforcement learning for the cartpole problem

Lars Brussee, S2528673

Supervisors:
Evert van Nieuwenburg and Felix Frohnert

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

dd/mm/yyyy

Abstract

With the importance of AI in the modern computing landscape, there is an interest in quantum machine learning, as it has advantages of faster convergence in some circumstances, due to a potential quantum advantage. Quantum machine learning replaces the neural network with a quantum circuit, with trainable parameters. When training such a quantum machine learning circuit, it can encounter a vanishing gradient problem, called the barren plateau problem, during the training process. As this slows down or stops the training process, investigating potential solutions to this problem is important. One possible solution to this problem is the mid-circuit measurement. This solution relies on forcing the quantum bit into a classical state. Some use of the mid-circuit measurement should remove the barren plateau, but overuse takes away the quantum advantage. Therefore, a balance exists for the amount of mid-circuit measurements. We investigate this balance in this paper, by training a simulated quantum reinforcement learning model with different levels of mid-circuit measurements, and investigating its performance in the cartpole environment. The performance is the cumulative reward over an episode. We found that the used quantum reinforcement circuit has a very variable performance, which causes the performance impact of the mid-circuit measurements to be limited if present. We theorize that this very limited impact may be due to the limited size of the quantum circuit.

Contents

1 Introduction

AI has had great impact in our society recently, being used in such diverse fields as management[MPS25], education[SVS24] and healthcare[ADADAAD25]. Therefore, interest into machine learning has greatly increased in recent times. These AI agents are usually created with neural networks, but can be made with any learning circuit. An interesting alternative learning circuit comes from the field of quantum computing, the *parameterized quantum circuit*. This type of circuit is made of quantum bits (qubits) and quantum gates. Some of those gates have tunable parameters that can be updated in a training process. Due to its quantum nature, quantum computing has some theoretical and practical advantages over classical computing, such as being able to get better time complexities for disjunctive normal forms, needing fewer examples for the learning process for some select problems, and being able to potentially access data much faster [ea18, chapter 4 & 5]. The performance of quantum computing for machine and reinforcement learning can be better than classical computing in certain circumstances, but worse in other circumstances. A sample of such results and corresponding circumstances can be seen in table 1.

Citation	Problem	Approach	Performance
[LS20]	Cartpole-V0	100 parameter NN	130 after 900 episodes
[LS20]	Cartpole-V0	DQN	135 after 250 episodes
[LS20]	Cartpole-V0	1000 parameter NN	125 after 650 episodes
[LS20]	Cartpole-V0	DDQN	125 after 350 episodes
[LS20]	Blackjack	100 parameter NN	-0.05 after 150 episodes
[LS20]	Blackjack	DQN	-0.05 after 410 episodes
[LS20]	Blackjack	10 parameter NN	-0.03 after 250 episodes
[LS20]	Blackjack	DDQN	-0.05 after 280 episodes
[JGM+21]	PQC gen SL	softmax PQC	18,5 after 1000 episodes
[JGM+21]	PQC gen SL	Deep NN	7,5 after 1000 episodes
[JGM+21]	PQC gen cliffwalk	softmax PQC	18,5 after 1000 episodes
[JGM+21]	PQC gen cliffwalk	Deep NN	2,5 after 1000 episodes

Table 1: Comparisons of quantum ((D)DQN and PQC) and classical (NNs) approaches. The approach and problem are shown, as well as the performance. Note that the Quantum circuits sometimes perform similarly, but outperform when the problem itself is quantum (denoted by PQC generated(gen))

However, specific problems in quantum computing for machine learning do arise. quantum hardware is fundamentally noisy at this moment and will remain noisy so long as the circuits are not made fault tolerant [ea18, chapter 10b]. It also suffers from another specific problem which this paper will tackle: The barren plateau. This problem consists of an exponentially vanishing gradient that usually occurs in circuits with a large depth. It also occurs in classical circuits, where it is known as the vanishing gradient problem. It is somewhat different in a classical context, as it occurs at larger depths than in quantum circuits[MBS+18, page 4]. The solution we investigate in this paper, which is mid-circuit measurements, relies on the quantum nature of the circuit, so the barren plateau problem will be tackled in the quantum computing context.

1.1 Machine and Reinforcement learning

We make use of the concepts of Machine and Reinforcement learning in this paper. These concepts will be introduced here. The field of Machine learning consists of teaching algorithms to performs tasks without being explicitly programmed how to do so. The algorithms usually learn by either being provided examples of inputs and their correct outputs (supervised learning) or by finding patterns in provided data(unsupervised learning). In Reinforcement learning (RL), an agent interacts with a dynamic environment, in order to maximize a reward value, or minimize a loss value, which is given by a reward or loss function. This agent observes the state of the environment and can take actions in the environment, in order to (potentially) change the state of the environment. An representation of this interaction can be seen in figure 4. The actions are taken in order to either learn what combinations of states and actions result in what reward(exploration), or to use the knowledge of the optimal action for the best known reward, to obtain this best reward(exploitation). A RL agent needs to balance between exploration and exploitation to gain the maximum cumulative reward. Reinforcement learning is further defined in subsection 2.1. The RL agent can be implemented using a variety of trainable circuits, such as neural networks.
A neural network consists of an input layer, hidden layers and an output layer. Each layer manipulates its input and sends the resulting output as the input for the next layer. The initial input is the observed state of the environment, in the case of Reinforcement learning. The layers manipulate the data based on learned weights, which are optimized with optimizers such as the Adam optimizer to take on values that result in a higher reward. If the neural network consists of multiple layers, it will be considered a deep RL agent.

1.2 Quantum circuits

We cover the concept of quantum circuits in this paper, which are used to implement Quantum Reinforcement Learning. A quantum circuit is a circuit that manipulates qubits with the use of quantum gates, in order to accomplish some task. These concepts are covered in detail in the definitions chapter 2.1, but we provide a simple overview here. A normal simple digital circuit consists of bits having their binary value manipulated by logic gates, such as AND, OR and NOT gates. We provide an example of a classical circuit in figure 1. A quantum circuit is essentially a quantum parallel to the classical circuit. The quantum nature of the circuit fundamentally changes how it works, however. The qubit, which is the quantum version of a bit [ea18, chapter 2], starts out like a bit, with a value of 1 or 0. It can be put into a superposition with certain gates however. This superposition is a state in which a measurement of the qubit will result in either a value of 1 or 0, depending on a probability distribution created by the state of the qubit [ea18, chapter 2]. Other quantum circuits can manipulate its state to change its probability, and qubits can be brought in a complex correlated state through quantum entanglement. These concepts are further covered in the definitions section 2.1. Do note that the measurement of the qubit forces it out of the superposition, which is important for the mid-circuit measurement.

1.3 Quantum Machine and Reinforcement learning

Quantum Machine learning and Quantum reinforcement learning use the quantum circuit for their respective form of learning. In order to do this, the specific form of a parameterized quantum

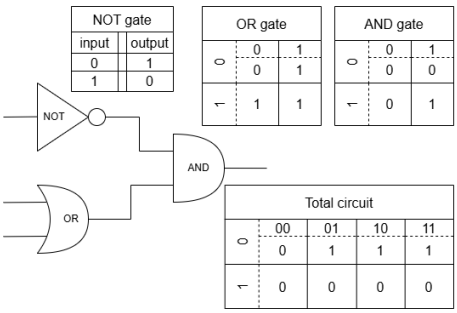

Figure 1: example of a classical circuit, with tables of the inputs and their resulting outputs for the gates and the whole circuit.

circuit needs to be used. It is parameterized, because it uses weights as parameters for some of the quantum gates and these weights can also be trained to allow the quantum circuit to learn. The input is encoded onto the qubits with the use of an encoder circuit and the data is manipulated with a variational circuit. The output consists of the qubits, which can be repeatedly sampled to obtain their probability distribution, or which are directly a probability distribution in the case of simulations on classical hardware. More detail can be found in the definition chapter 2.1 under PQC (parameterized quantum circuit)

1.4 The barren plateau and mid-circuit measurements

As previously mentioned, barren plateau is the term for exponentially vanishing gradients in quantum circuits, which makes machine learning hard when they occur. This is due to the fact that many machine learning models work by finding the gradient of the loss or reward function, so that the agent may be improved into a better direction. If there is no gradient that can be used to improve, no improvement can be made in these cases. As mentioned earlier in the introduction, the same problem can be found in deep classical neural networks[CSV+21]. According to Zhao[ZG21, introduction], the barren plateau is caused by varying circumstances. It can occur in both deep and shallow quantum circuits according to [CSV+21, page 2]. More details can be read in section2.1.
In order to tackle these barren plateaus in the context of quantum machine learning, a varied number of techniques can be used. M Cerezo et al[CSV+21] mentions methods for tackling the barren plateau issue. These are the handling of parameter initialization and parameterized quantum circuit creation. These techniques are investigated by Grant et al[GWOB19] and Verdon et al [VML+19] respectively. However, we investigate another less investigated technique for handling the barren plateau problem: *Mid-circuit measurements*.
Mid-circuit measurements are measurements of a qubit within the quantum circuit. These measurements force the quantum state of the qubit into a 0 or a 1, based on the probability distribution created by the quantum state of its qubit2.1. When pondering on solving the barren plateau problem with mid-circuit measurements, it is helpful to consider the two extremes of applying the mid-circuit measurements. On one side, the expressive power from a quantum circuit comes from its entangled qubits, but not applying any mid-circuit measurements leaves the quantum circuit with the barren plateau problem. On the other hand, applying mid-circuit measurements after every quantum gate removes the entanglement and therefore the expressive power. Therefore, there

is most likely a balance between the amount of mid-circuit measurements: enough to reduce the barren plateau, but no more than necessary for that purpose. We will test this hypothesis of the existence of this balance in a narrow context, in order to provide insight into this solution to the barren plateau problem.
The specific research question of this paper is as follows: "*For the created quantum circuit and its hyper parameters and for the Reinforcement learning problem of the cartpole problem, is there an optimal amount of mid-circuit measurements? If so, which amount of mid-circuit measurements is optimal for the learning process?*" The results of this paper can inform future research in what direction this topic can be taken; Should we find an optimal amount of mid-circuit measurements in our context, future quantum RL models can be initially trained with the found optimal amount of mid-circuit measurements, as that amount of mid-circuit measurements is optimal in at least one quantum RL context.

1.5 Thesis overview

This paper is structured as follows: This chapter contains the introduction; Section 2 discusses related work, what the previous research shows for the causes of barren plateaus, uses for mid-circuit measurements and the use of quantum computing for machine and reinforcement learning; Section 2.1 includes the definitions for the varied terms and concepts mentioned in this paper; Section 3 describes the experimental setup and therefore the specific quantum circuit and Reinforcement learning problem; Section 4 shows the data gathered during the experiment; Section 5 analyses the result, concludes what they may imply, shows limitations of the research, and suggests further research into the topic based on the finding of this paper.

2 Related Work

In this section, we will cover works related to mid-circuit measurements, barren plateaus and quantum reinforcement learning. [BGK+22] covers the use of mid-circuit measurements for error mitigation on quantum hardware. This application of mid-circuit measurements implies that mid-circuit measurement application can serve as both barren plateau mitigation and error mitigation. However, [ea25] created a benchmark to measure the error created by mid-circuit measurements. This implies that mid-circuit measurements can also introduce more error. Combined, this implies that, for the purposes of error mitigation, there must also be a balance in the frequency of mid-circuit measurement application. [HLK+25] uses mid-circuit measurements for yet another purpose: To facilitate classical communication between sections of the quantum circuit: if a certain measurement is measured with the mid-circuit measurement, apply a certain quantum gate. They show that this can improve the quantum circuit for binary classification, almost as much as quantum communication (entanglement) between sections.
As seen in the introduction, [MBS+18] covers the causes behind the barren plateau. The paper also puts a scaling to the vanishing gradient, as it points out that it scales exponentially with the number of qubits. The number of layers also cause the barren plateau, up to a certain point, depending on the number of qubits[MBS+18, figure 4]. This occurs at tens to hundreds of layers however, so it is irrelevant for most PQCs used in Quantum RL that we have seen in other papers. As we mentioned in the introduction, [GWOB19] tackles the barren plateau problem with techniques

different from mid-circuit measurements in the form of initialization strategies. [VML+19] introduces a new quantum circuit type, the Quantum Graph Neural Networks. They created this QGNN for distributed quantum systems, but according to [CSV+21, page 2] it is a form of PQC that can work around the barren plateau problem.
Quantum Machine Learning and Quantum Reinforcement learning are covered by a variety of papers, such as the previously mentioned [VML+19] which introduces the QGNN. They use it for certain graph problems (clustering and classification), the creation of quantum sensor networks and quantum Hamiltonian dynamics. These problems seem to make use of the nature of the neural networks, as these problem consist of both quantum systems and graphs. This reinforces the idea that quantum computers are suitable for tackling quantum problems, which can also be seen in the results from table 1. [FN18] trains a quantum neural network for a classical classification problem however: The classification of letters. They had to limit the problem to distinguishing between two digits, due to the limitations of the input size with the limited number of qubits. They used 9 qubits in total, Their results are a 97% accuracy after a 1000 episodes, though no direct comparison with performance of a classical circuit was made. [ea23] uses a PQC to apply Bayesian learning for usage in a NP-complete optimization problem (weighted max cut), modeling of quantum magnetism and as a generative model for stamps from the Hidalgo dataset. It uses Laplace prior regularization, which they claim may help with the barren plateau problem, alongside correlating trainable parameters. [JTPN+21] uses quantum algorithms in order to improve certain RL models (energy based models). These RL models suffer from a problem in their sampling complexity, that the quantum algorithms can more efficiently solve.
The most interesting works for this paper are those with a similar setup. These are [LS20] and [JGM+21], as they both apply QRL to the cartpole problem. Their setup can be found in tables 2, 3 and 4, with their results in table 5. These will be compared with the setup of this paper in chapter 3, and their results with ours in chapter 5.

id	Citation	Problem	algorithm	gradient method
DQNCart	[LS20]	Cartpole-V0	DQN	Backpropagation
DDQNCart	[LS20]	Cartpole-V0	DDQN	Backpropagation
DQNBlack	[LS20]	Blackjack	DQN	Backpropagation
DDQNBlack	[LS20]	Blackjack	DDQN	Backpropagation
RRawCart	[JGM+21]	Cartpole-V1	REINFORCE Raw	Parameter shift rule
RSftCart	[JGM+21]	Cartpole-V1	REINFORCE Softmax	Parameter shift rule
RRawMnt	[JGM+21]	Mountaincar-V0	REINFORCE Raw	Parameter shift rule
RSftMnt	[JGM+21]	Mountaincar-V0	REINFORCE Softmax	Parameter shift rule
RRawBot	[JGM+21]	Acrobot-V1	REINFORCE Raw	Parameter shift rule
RSftBot	[JGM+21]	Acrobot-V1	REINFORCE Softmax	Parameter shift rule

Table 2: Overview of Quantum RL setups from 2 papers. For comparisons with the setup in this paper. These are given ids for comparison in different tables. We investigated the QRL setup from [CYQ+20], but it fails to disclose some of the details of their setup, such as γ, which they do use.

id	layers	encoding layer & scheme	variational layer	qubits
All from [LS20]	1 enc, 3 var	Rx,Rz. See caption	CNOT +x,y,z Rot	4
R(Raw/Sft)Cart	$1 \times$ (var+enc)	Ry, Rz learnable λ to scale input	Rz, Ry, CZ	4
R(RawSft)Mnt	$4 \times$ (var+enc)	Ry, Rz learnable λ to scale input	Rz, Ry, CZ	2
R(Raw/Sft)Bot	$2 \times$ (var+enc)	Ry, Rz learnable λ to scale input	Rz, Ry, CZ	6

Table 3: The structure and encoding of the used PQCs in the setups from table 2. The encoding scheme from [LS20] usually scales the input variables between 0 and 2π, scaling based on their minimum and maximum values (which come from the environment). For the velocity in the cartpole problem, this was impossible, as these have infinite maxima and minima, so those are set to π if positive, 0 otherwise.

id	learning rate	rl decay	γ
All from [LS20]	1. 0.01 minimum	$\epsilon = 0.9\epsilon$	0.95
RRaw(Cart/Bot)	[0.01,0,0.1]	-	1
RSft(Cart/Bot)	[0.01,0.1,0.1]	-	1
RRawMnt	[0.01,0,0.01]	-	1
RSftMnt	[0.01,0.1,0.01]	-	1

Table 4: Hyperparameters from the setups from table 2. The RRaq/Sft setups have 3 learning rates, as it has 3 different types of learnable parameters: rotation weights (the normal weights), observable weights, and λ for the input weights.

id	Best result	its episode	best possible result
DQNCart	≈ 135	≈ 250	300
DDQNCart	≈ 125	≈ 350	300
DQNBlack	≈ -0.05	≈ 410	1
DDQNBlack	≈ -0.05	≈ 280	1
RRawCart	≈ 400	≈ 1800	500
RSftCart	≈ 450	≈ 1050	500
RRawMnt	≈ -110	≈ 1600	0
RSftMnt	≈ -80	≈ 1100	0
RRawBot	≈ -250	≈ 1800	-100
RSftBot	≈ -150	≈ 500	-100

Table 5: The results of the setups. The best results and their episodes are approximate, because these have been taken from reading of graphs, which lacks exact precision.

2.1 Definitions

This section covers definitions and explanations of terms and concepts used in this paper. We use these concepts in different sections of this paper, so this section covers concepts already used earlier in the paper, as well as after this section. Therefore, this section should be read when necessary, when a concept is unknown to the reader. The terms and concepts are in bold text, so that the correct paragraph can be quickly found.
As mentioned in chapter 1.2 **qubits** are the information unit for quantum circuits. Differently from bits, the qubits can consist of a superposition of the possible values of a bit. The qubit is usually denoted $|\psi\rangle$, with the equation

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

where α and β are so called "probability amplitudes" that are made up of complex numbers, to which quantum operations can be performed. $|\psi\rangle = |0\rangle$ and $|\psi\rangle = |1\rangle$ denote the states where the qubit is purely a 0 or 1 respectively. quantum operations (by use of quantum gates) manipulate α and β to change the probability distribution for the qubit to be in the 0 or 1 state. The probability of measuring 0 is $|\alpha|^2$ and 1 $|\beta|^2$, with the total probability of the two values adding up to 1. In figure 2, this interpretation can be seen in the form of a Bloch sphere. The states that are not purely in the 0 or 1 state (not $\alpha = 1, \beta = 0$ or $\alpha = 0, \beta = 1$) means that the qubit is in a **superposition**. When a quantum superposition is **measured**, it's superposition collapses, so that it's state becomes either 0 or 1, depending on the probability amplitudes. It is then in a classical state, which can be manipulated into a quantum state again with quantum circuits.
Information can be encoded into the superposition using rotation, as can also be seen in figure 2. These can be used for quantum circuits to encode classical input data and apply weights to these inputs. The quantum gates that apply these rotation are called x-, y- and z-**rotation gates**. The weights and inputs are treated in the same manner, as they are both the angles used in rotation gates. The qubits must also be entangled with each other, so that qubits can influence each other. This requires two-qubit gates, the most common of which is the **controlled NOT gate**(CNOT), which applies a NOT gate to the first qubit, if and only if the second qubit is a 1. The entanglement is necessary, as the quantum entanglement is what differentiates the quantum circuit from classical computation[ea18, chapter 2]. The **Quantum entanglement** fundamentally means that the state of one entangled quantum state cannot be described without considering the state it was entangled with. Their states become correlated with one another. In the context of quantum computing, this means that when entangled, the state of one qubit influences the state of the other qubit.
Another angle of looking at quantum gates and entanglement is that of matrix multiplication. The

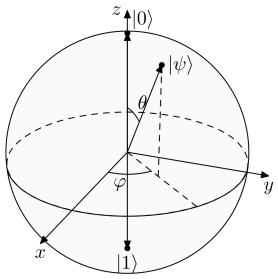

Figure 2: Bloch sphere representation of a qubit. The qubit is represented as a point on the shell of the sphere, because of the constraint $|\alpha|^2 + |\beta|^2 = 1$. This constraint also allows for the encoding of α and β as angles on this sphere, with the formula $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$. θ is the angle on the $z-y$ plane and ϕ the $z-x$ plane. Note the $|0\rangle$ and $|1\rangle$ poles, which correspond to a qubit that always reads out 0 or 1. Rotation gates rotate the qubit point around the x, y or z axis, rotating by a provided angle in π radians. This can be used to encode weights into a quantum circuit, as the rotation usually changes the quantum state of the qubit. Image by Notezik - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=179712998.

matrices for a qubit and the quantum gates used in this paper are as follows:

$$
\begin{aligned}
|\psi\rangle &= \begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \alpha|0\rangle + \beta|1\rangle \\
\text{CNOT} &= \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix} \\
\text{X-Rot}(\phi) &= \begin{bmatrix}\cos(\phi/2) & -i\sin(\phi/2)\\ -i\sin(\phi/2) & \cos(\phi/2)\end{bmatrix} \\
\text{Y-Rot}(\phi) &= \begin{bmatrix}\cos(\phi/2) & -\sin(\phi/2)\\ \sin(\phi/2) & \cos(\phi/2)\end{bmatrix} \\
\text{Z-Rot}(\phi) &= \begin{bmatrix}e^{-i\phi/2} & 0\\ 0 & e^{i\phi/2}\end{bmatrix}
\end{aligned}
\tag{1}
$$

These matrices illustrate the way the input and weights are applied to the qubits, and how the qubits are manipulated.
The quantum gates can be applied in succession, which form a quantum circuit, to manipulate qubits. When using learnable parameters as inputs for this manipulation, a **parameterized quantum circuit** is formed. An example of such a circuit can be seen in figure 6. According to Benedetti[BLSF19], parameterized quantum circuits, in this paper referred to as **PQCs**, are quantum circuits consisting of fixed gates and adjustable gates. The adjustable gates are used to apply parameters (the weights and the input) and are made using rotation gates. The fixed gates are used to combine qubit input and consist of CNOT gates. These different types of gates are combined to create different circuits that make up the PQC. There is an encoder circuit, which is used to encode the input data, and a variational circuit, which corresponds to a neural network.

The qubit data may also need to be pooled so that the right number of qubits are measured for the output. The output of each qubit is one 0 or 1 when measured. This means that, in order to obtain a probability distribution of the qubits, the qubits have to be measured many times, requiring running the circuit that many times too. The structure of a PQC is present in figure 3. The actual implementation of the circuit can be found in the methods section 3, in figure 6.

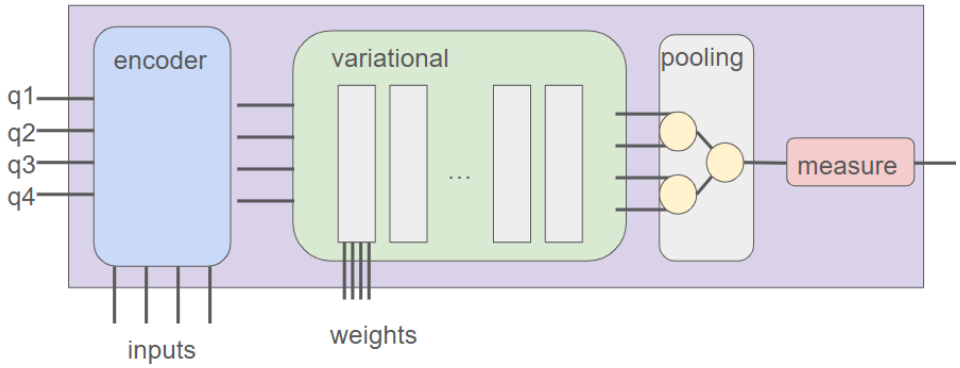

Figure 3: An overview of the global structure of a PQC. qubits are manipulated from left to right. The embedding circuit is applied, followed by a variational circuit, which consists of layers. The qubits are pooled to be combined for the last actually used measurement for the output.

A **mid-circuit measurement** is the measuring of a qubit within the quantum circuit. As any measure of a qubit results in the collapse of the super position of the qubit, the quantum nature of the qubit is temporarily removed after the mid-circuit measurement, until a new superposition is created by the application of a quantum gate. As mentioned in the introduction, the mid-circuit measurement is used in this context for the reduction of the barren plateau problem.
The **barren plateau problem** is the occurrence of vanishing gradients that removes the usable gradients used in the learning process, therefore removing the ability for the RL agent to learn. It occurs due to sections of the circuit generating similar output, while having different parameters and being statistically independent[CJ23, page 5]. According to Zhao[ZG21, introduction], the problem occurs on quantum circuits when a variety of elements are present: Noise, too much entanglement or specific structures of the circuit. It can occur in both deep and shallow quantum circuits when the initial state is randomly initiated[CSV+21, page 2].
Reinforcement learning (RL) is mathematically modeled as a Markov decision process, which consists of the state space(S), the action space(A), the probabilities of the transitions($P_a(s, s')$) between states s and s' given a certain action a and the rewards provided by those transitions $R_a(s, s')$. The RL agent optimizes the policy $\pi : S \times A$ so that the probabilities for selecting an action given a state provide maximum reward or minimal loss during all steps of an episode. An episode consists of all steps taken from an initial state to an end state. The balance between this learned optimal policy π for exploitation and exploration depends on the used learning algorithm, in this paper we use the REINFORCE algorithm (see section 3). A representation of the RL agent interaction with the environment is given in figure 4.
The **cartpole** environment from the gymnasium library is the reinforcement learning environment that we chose in this paper. This environment consists of a cart that can move without friction on a horizontal plane. The cart also has a freely rotating pole attached to it. A snapshot of this

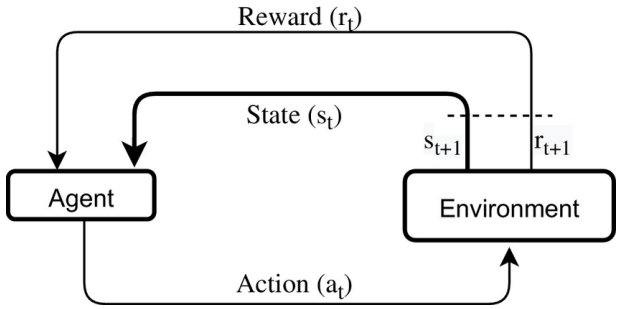

Figure 4: The interaction between the RL agent and its environment. Image from [JK22]

environment can be seen in figure 5. The reward that the agent wishes to maximize consists of a value of 1 for every step taken by the agent in the environment. The sum of all rewards from all steps is the **cumulative reward**. The agent wishes to maximize this cumulative reward. Therefore, the goal of the agent is to avoid termination. The environment terminates when the pole has an angle outside of a certain range, when the cart itself moved outside of a range of positions, and when the maximum number of steps is taken.For this environment, the observation space consists of 4 variables, and the action space of 1 binary variable, with the value 0 moving the cart to the left and the value 1 moving the cart to the right. These values, the observation space, and the action space are defined in table 6

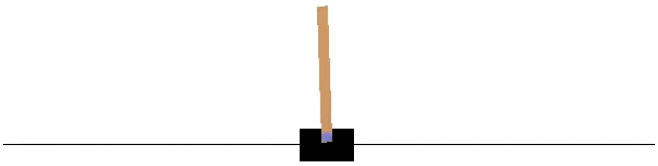

Figure 5: Render of the cartpole environment. The cart moves along the horizontal line and its pole can rotate. By Condordellanebbia - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=151762281

ranges	direction	Cart Position	Cart Velocity	Pole Angle in rad	Pole Angular Velocity
possible	0 or 1	$(-4.8, 4.8)$	$(-\text{inf},\text{inf})$	(-0.418,0.418)	$(-\text{inf},\text{inf})$
non-termination	-	$(-2.4, 2.4)$	$(-\text{inf},\text{inf})$	(-0.2095,0.2095)	$(-\text{inf},\text{inf})$
initial state	-	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)

Table 6: The cartpole-v1 environment, its action and state spaces and the allowed ranges of values that are needed to avoid termination.

3 Methods

For this paper, we need to implement the quantum circuit and RL problem. For the implementation, we use the Python programming language[Fou26a]. We selected a number of libraries for creating the experiment. These can be found in table 7.

Library	gymnasium	pytorch	numpy	Pennylane
Version	1.2.3	2.10.0	2.4.2	0.44.0
usage	problem env	tensors, optimizer	computations,stochastics	quantum components
reference	[FF26]	[Fou26b]	[tea26]	[Xan26]

Table 7: The used libraries, their versions, what it is mainly used for and references.

We set up the experiment to optimize the policy for obtaining a maximum reward from the Cartpole-v1 environment 2.1. The hyperparameters for the trainingscontext can be seen in table 8. We fix the hyperparameters to the values for which the PQC starts to learn. We set the number of layers and episodes to a value that keeps the experiment within a reasonable timeframe, while allowing the circuit to learn. Because the goal of the experiment is to show a difference between levels of mid-circuit measurement, the actual performance of the PQC does not have to be optimal; It only needs to be able to learn enough to show a difference between mid-circuit measurement amounts. This means that we can set the hyperparameters as soon as it is able to learn, which in turn means that the hyperparameters can be changed around empirically until it learns.
The measurement probability is the only hyperparameter that is variable. This measurement probability is the probability of applying a mid-circuit measurement on each qubit after a variational circuit layer. variational circuit layers can be seen in figure 6. This probability allows us to change the amount of mid-circuit measurement applied in the experiment, so that the optimal amount of mid-circuit measurement can be found. The learning algorithm we used is REINFORCE.The

hyperparameter	learning rate	# episodes	Gamma	# layers	measurement probability
value	0.01	300	0.99	2	varies

Table 8: The hyperparameters used for the experiment.

REINFORCE algorithm as used for this paper follows the pseudo-code seen in algorithm 1. We adapted the REINFORCE algorithm as present in an online available jupiter notebook [vN25]. We use this algorithm because it works with a probability distribution as the policy, where the simulated quantum circuit is also able to output a probability distribution. Note that this learning algorithm cannot be used by real quantum hardware without excessive sampling to sample the underlying probability distribution, which would increase overhead costs.
For the learning process, we use the backpropagation algorithm. Although the standard in classical RL, backpropagation is not actually possible in true quantum hardware. This is due to the nature of quantum systems. Backpropagation requires information on the impact of each neuron on the output. This would require reading the quantum state within a network. Reading a quantum state collapses the superposition of the quantum state, therefore influencing the output. This is

Algorithm 1 Pseudocode of the training loop for the experiment, with a REINFORCE algorithm. Note that the probability of actions to take corresponds to the probability distribution that the PQC outputs in the simulation.

environment = cartpole-v1
policy $\pi_\theta \leftarrow$ random weights
for each episode **do**
    state = environment.reset()
    Done = False
    initialize/reset lists of actions, states, rewards and action-log-probabilities to empty lists
    while Done== False **do**
        action-distribution = π_θ(state)
        action = action-distribution.sample()
        log probability = action-distribution.log-prob(action)
        next-state, reward and Done = environment.step(action)
        actions.append(action), states.append(state), rewards.append(reward),
        action-log-probabilities.append(log probability)
        state = next-state
    end while
    returns = [], sum-returns = 0
    for r in reversed(rewards) **do**
        sum-return = r + sum-return $\times$ Gamma
        return.insert(0,sum-return)
    end for
    calculate the loss = $-\sum$action-log-probabilities$_t \times$returns$_t$
    apply this loss with backwards propagation to improve the agent
    print the sum of the rewards
end for

undesirable, as the network would lose its quantum state, becoming classical and losing its quantum advantage.
To work around this issue for backpropagation, we set up the experiment to make use of a simulated quantum circuit, of which the internal state is not actually a real quantum state, making it readable without collapse. This makes it so that we can use backpropagation, making the learning process much easier. This means that we can use a normal RL optimizer. For the experiment, we opted for using the Adam optimizer from pytorch. This is done for two reasons: 1) It works on a wide range of learning problems, such as with noisy or sparse gradients and 2) it usually does not require extensive hyperparameter tuning, which saves time in developing the experiment[KB17].
As mentioned at the start of this section, we used Pennylane for the implementation of the PQC itself. The pennylane library uses a features argument for the layers it constructs, which we used for applying inputs and weights. See figure 6 for the implemented circuit.
As touched on in the definitions subsection, the PQC needs an encoder circuit to encode 2.1 the input data, and a variational circuit for the actual learning process. The encoder circuit we chose consists of one Pennylane angle embedding layer, from a pennylane template, with the input as the features. This embedding layer consists of x-rotation gates. We selected this template, as it can take in floating point numbers, and is simple.
The variational circuit section consists of two repeating custom created layers. This custom layer we made consists of x,y and z rotation gates for each qubit which take in weights for each of the rotations. The rotation gates are followed by CNOT gates combining the qubits. This gate combination comes from the strongly entangling layer template from Pennylane. Each of these is followed by a probability based mid-circuit measurement possibilities for each qubit after each variational circuit layer.
The variational circuit can be considered as consisting of fully connected layers, with mid-circuit measurements following each layer. The variational circuit is followed by a pooling layer, where the wires are entangled with CNOT into one wire. This is so that one qubit can be read for the output of the circuit.
The number of learnable parameters of the PQC that is created this way is 12 per strongly entangling layer. Each wire has 3 weights, for a x, y and z rotation. With 2 layers, this brings the total number of weights to 24.
The experimental setup we created consisted of running the code with 10 different seeds, with values for the mid-circuit measurement probabilities from 0.0 to 0.9 in steps of 0.1 being applied to each of these runs, adding up to a 100 runs. We ran the code itself using the compute resources from the Academic Leiden Interdisciplinary Cluster Environment (ALICE) provided by Leiden University. We chose to run 10 seeds in parallel, and apply the mid-circuit measurement probabilities consecutively. We did all of this using a SLURM file, which is used to set up jobs on clusters like ALICE[Sch26]. It ran on the cpu-zen4 partition with 100 M memory allocated per cpu.
The code we made for the experiment has a number of flaws, which will be discussed in the 5.1 subsection. This is to be kept in mind for reproducibility reasons. The code itself can be found in this github repository[Bru26].
The output of the experiment is text recording the cumulative reward of an episode during the training process. Due to the experiment taking up more time than we allocated for the ALICE cluster, the seeds have to be reinitialized during the experiment. Table 9 shows at which points the seeds were reset to the provided seed, and table 10 what seeds were provided. We provide these seeds for facilitating an exact reproduction of the experiment. The duration of each run of the

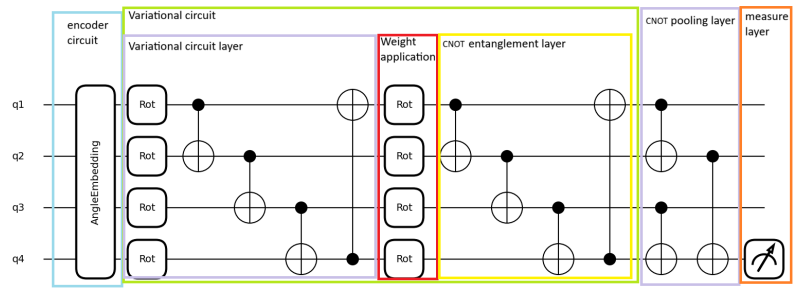

Figure 6: An overview of the structure of the PQC. The qubits/wires q1 to q4 start on the left. The angle embedding circuit is applied, followed by a variational circuit. The variational circuit consists of the rotations and entanglement with each other (the crosses in circles). Each variational circuit layer may be followed by a measurement for a qubit, depending on the mid-circuit measurement probability. Two variational circuit layers are applied, and then the qubits are pooled to be combined for the last actually used measurement for the output, as seen in the bottom right.

experiment on the cluster can be seen in table 11.
As mentioned in chapter 2, [LS20] and [JGM+21] use a setup similar to the setup in this paper for the cartpole environment. This makes comparisons between those two setups and our setup useful, as the limited differences may be able to explain differences in results, giving insight into the results when these are investigated. Their setups can be found in table 2, with their PQC structures in table 3 and their hyperparameters in table 4. The results of these similar setups can be found in table 5. How the results compare will be covered in chapter 5.
We will now discus differences and similarities in their setups for the cartpole environment. Firstly, [LS20] uses (D)DQN instead of REINFORCE, but [JGM+21] uses the parameter shift rule instead of backpropagation. So their algorithms both share similarities with the setup in our experiment, but they also both differ for our setup. Secondly, The number of qubits is the same as our setup. Thirdly, [LS20] uses a decaying learning rate for the rotation gate parameters, that decays to the same learning rate as in our setup and the setup from [JGM+21]. [JGM+21] does use learning rates for weights for the inputs and observables. Lastly, [LS20] uses one extra variational layer, whilst [JGM+21] uses only one variational layer.
The differences in setup seem to indicate that a more complex PQC was used by [LS20] and [JGM+21]. As we make a PQC for finding the effects of mid-circuit measurements, we do not need the PQC to be as complex as those in [LS20] and [JGM+21]. Their setups would allow us to conclude that problem in performance are caused by an insufficiently deep PQC, as their setups with similar algorithms do obtain okay or good performance. According to [JGM+21], the setup in the research of [LS20] has not been able to solve classical benchmarking tasks as per the specifications of the Gym library. Based on the remarks of [JGM+21], and the fact that that [JGM+21] does manage to train the QRL agents up to the maximum reward for some of the environments, attaining a maximum reward for an environment may be the specification for solving the benchmark. This means that that the setup from [JGM+21] is more optimal, although it did have to train for more than a 1000 episodes instead of 250 episodes for such performance.

seed evolution	set	-	-	reset	-	-	-	-	reset for 2,3,5,6,7	-
measurement probability	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

Table 9: At which points in the running process the seeds where set or reset.

number of thread	0	1	2	3	4	5	6	7	8	9
seed	5322	3817	2283	6033	4326	5197	1982	7394	8761	9703

Table 10: The threads that ran in parallel and their seeds.

Allocated time	2 days	7 days	3 days
Average real time taken	2-00:00:14	6-12:53:21	2-09:17:13
range of measurement probabilities	0.0 until 0.2	0.3 until 0.7	0.8 and 0.9

Table 11: Each run due to the time limit, the time taken, the memory allocated and the seeds it managed to run through.

4 Results

The evolution of the cumulative rewards over the 300 episodes can be seen in the figures 7,8,9 and 10. These figures show the episode number plotted against the cumulative reward. As mentioned in the definition of the Cartpole environment2.1, the cumulative reward of an episode is the number of steps the agent manages to take before the termination step, as each steps gives a reward of 1. Each line is for each mid-circuit measurement probability or the average cumulative reward of all mid-circuit measurement probabilities. This is to show the impact of the mid-circuit measurements on the cumulative reward during the training process. It can be seen in its raw form in figure 7. As this data is rather noisy, we decided to create a version of the graph with a rolling average of 10 episodes in figure 8. In order to gain insight into the noise from the different threads, the standard deviation can be seen in figure 9 and can be seen with a rolling average per 10 episodes in figure 10. The rolling average is created with the following definition:

$$\text{Rolling average in episode i} : \sum_{k=i-4}^{i+5} \frac{k}{10}$$

With the first and last 5 episodes taking the sum of the cumulative rewards of the first and last 10 episodes respectively, and dividing those by 10. This causes these first and last episodes to all have the same value, which can be seen as a flat section of their curves.

5 Conclusions and Further Research

Firstly, the data as gathered directly from the experiment has a high amount of noise, as can be seen in figure 7. Combined with figure 9, we can observe that the PQC is highly inconsistent in its performance with each run and episode. However, trends can be observed. The standard deviation

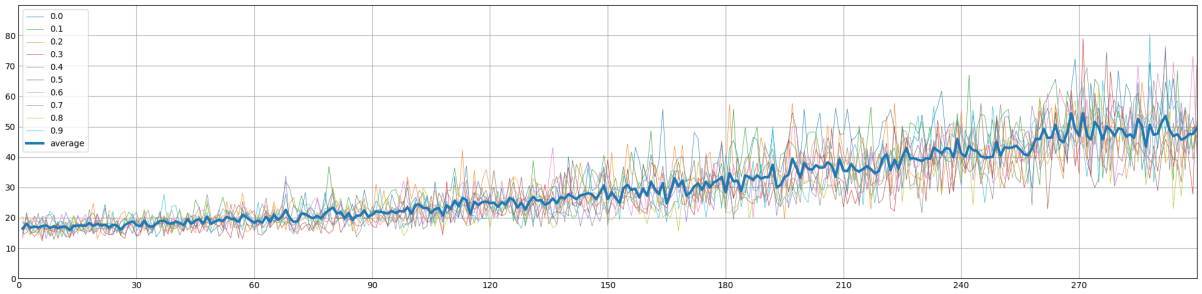

Figure 7: Cumulative reward of each episode, averaged over the 10 threads

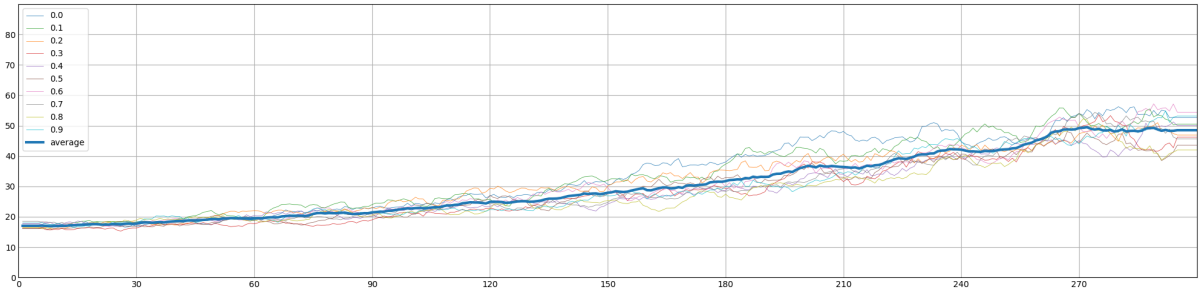

Figure 8: Cumulative reward of each episode, averaged over the 10 threads, and rolling average per 10 episodes. The first and last 5 episodes where averaged for their 10 nearest episodes, and are therefore 'flat".

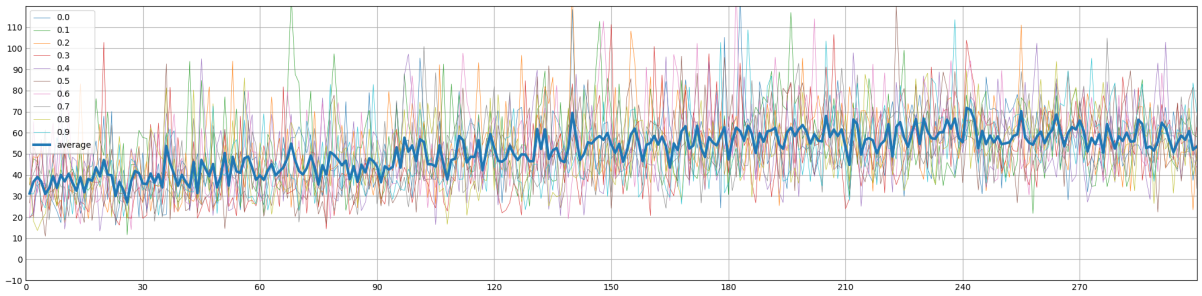

Figure 9: Standard deviation as a percentage of the average value of the cumulative reward of each episode, averaged over the 10 threads. Illustrative of the noise in the data.

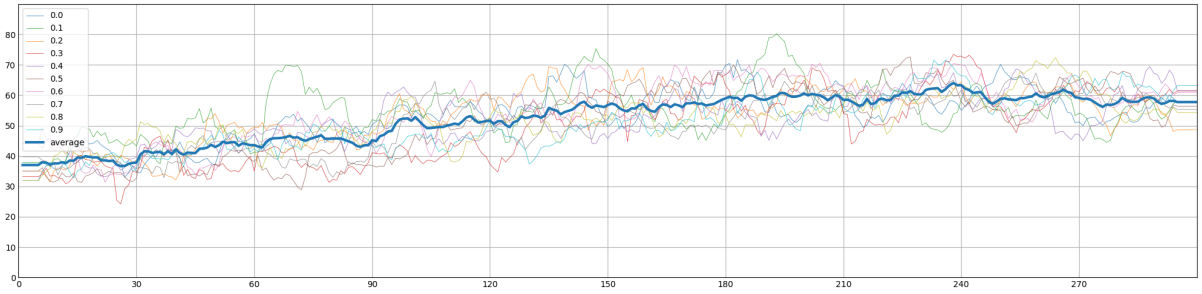

Figure 10: Standard deviation as a percentage of the average value of the cumulative reward of each episode, averaged over the 10 threads and over an 10 episode window. To show trends in the standard deviation of the threads.

increases proportionally to the average value as it learns, after which it seems to plateau at around the 180 episode mark. Figure 10 shows these phases for the standard deviation more clearly. These phases in the standard deviation show that, for the experiment, the noise in the data does not increase after 180 episodes in the training process.
Secondly, we might observe by looking at figures 7 and 8, that the PQC does gain an increased cumulative reward over time. The circuit therefore does optimize the policy for the cartpole environment. It is impossible to observe an optimal probability for the mid-circuit measurements in figure 7, as the data is too noisy. A trend for the values of 0.0 and 0.1 for the optimal probability of mid-circuit measurements may be observed in figure 8, although this is hard to discern. We expected the high mid-circuit measurement probabilities to underperform significantly, as it should lose its quantum advantage when the mid-circuit measurement is applied often. This suggests that the circuit itself is too simple for the quantum advantage to appear.
Thirdly, when disregarding the last 5 datapoints due to the rolling average, there appears to no longer be an increase in the cumulative reward after around the 270 episode mark, which indicated that the quantum RL agent was no longer improving with the learning process. This differs between the different mid-circuit measurement probabilities, but does occur on average. The difference between probabilities could be the result of a high amount of noise, which is indicated by the high standard deviation that is present in the data as can be seen in figure 10. Or it could be the result of actual differences between probabilities, which would imply that some probabilities for mid-circuit measurements may be more optimal when training with more episodes. The actual lack of convergence after the 270 episodes mark seems to indicate that the quantum circuit was not expressive enough.
Fourthly, when comparing the performance of our experiment with that of the setups from table 5, for the cartpole environment, it can be noted that our performance is much worse than the performance of [LS20] and [JGM+21]. All of their setups for the cartpole environment perform above 125, whilst our best result in, on average, around 55. This shows that our PQC is much less powerful, which adds support to the idea that the PQC was not complex enough, as their PQCs are more complex. The addition of one or more layers will most likely improve performance, which is usefull for further research. Another thing to note in comparison with the other setups, is that the problem lies in the PQC, as [LS20] gains better results with one variational layer extra, whilst also training for 300 episodes. Do note that gaining better performance does not guaranty

different performances for different mid-circuit measurement rates, but more circuit depth should increase the barren plateau problem [MBS+18], so making the PQC more complex should make the mid-circuit measurements more relevant too.

5.1 Limitations

There are a number of limitations of the performed research. Firstly, the simulation of the quantum circuit was very slow. This was due to the mid-circuit measurements in combination with the backpropagation algorithm. The experiment needed around an hour for a mid-circuit measurement probability of 0.0, whereas any other probability made the training process take more than a day. This limited the complexity of the PQC and the number of layers that could be applied. This limited the expressive capability of the PQC, which could explain its inconsistent performance.
Secondly, the code that was used for the experiment was not cleaned up properly. When wishing for guaranteed reproducibility, the original flawed code should be used if the results are to be exactly replicated.
Thirdly, the impact of the mid-circuit measurements seems to be low to nonexistent. Although this is a valid finding, the cause for the lack of impact can have a number of causes. As mentioned in section 1.4, the Barren plateau issue can be handled with correct parameter initialization and the structure of the parameterized quantum circuit. This makes it possible that the barren plateau problem was already handled by the structure of the used PQC or the chosen hyperparameters. As the PQC is rather simple due to the long running time for the experiment, its simple structure might have made mid-circuit measurements unnecessary. In order to investigate this, future research should first construct a PQC that suffers from the barren plateau problem and then apply the different probabilities for the mid-circuit measurements.
Fourthly, the use of simulation for the PQC and the use of backpropagation may provide unrepresentative results that do not represent actual behavior of real PQCs. This requires research with actual quantum hardware to investigate.

5.2 Further Research

Based on the limitations of this research, more research is needed to investigate the impact of mid-circuit measurements on PQCs. This future research could look at creating a deeper PQC that is confirmed to suffer from the barren plateau problem, and then apply similar methodology to investigate the impact. Another possibility for future research is to investigate the impact of mid-circuit measurements on actual quantum hardware, instead of simulated hardware. Broader research into other RL problems or other ML problems is also recommended, although not as concrete in its necessity. Instead, this research would provide data for a more generalized analysis of the subject in different conditions. Further research could also have more episodes for the training process, to be able to observe more difference in the amount of mid-circuit measurements.

5.3 Conclusion

In conclusion, the impact of mid-circuit measurements on a parameterized quantum circuit for tackling the barren plateau problem is limited or non-existent for the Cartpole-v1 problem and simulated parameterized quantum circuit as covered in this paper. Future research should focus on

changing the circuit to ensure the presence of the barren plateau problem, and may be performed on real quantum hardware. It should also use a larger PQC so that the differences between the mid-circuit measurement rates are more pronounced.

References

[ADADAAD25] Rand Al-Dmour, Hani Al-Dmour, Eatedal Basheer Amin, and Ahmed Al-Dmour. Impact of ai and big data analytics on healthcare outcomes: An empirical study in jordanian healthcare institutions. *DIGITAL HEALTH*, 11:20552076241311051, 2025.

[BGK+22] Ludmila Botelho, Adam Glos, Akash Kundu, Jarosław Adam Miszczak, Özlem Salehi, and Zoltán Zimborás. Error mitigation for variational quantum algorithms through mid-circuit measurements. *Physical Review A*, 105(2), February 2022.

[BLSF19] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, November 2019.

[Bru26] Lars Brussee. Mid-circuit-measurement-impact-thesis-2026-experiment, 2026.

[CJ23] Nguyen T. Cybulski J.L. Impact of barren plateaus countermeasures on the quantum neural network capacity to learn. *Quantum Inf Process*, 22(442), 2023.

[CSV+21] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1), March 2021.

[CYQ+20] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning, 2020.

[ea18] Ciliberto Carlo et al. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A*, 2018.

[ea23] Samuel Duffield et al. Bayesian learning of parameterised quantum circuits. *Mach. Learn.: Sci. Technol*, 2023.

[ea25] Daniel Hothem et al. Measuring error rates of mid-circuit measurements. *Nature communications*, 2025.

[FF26] Farama-Foundation. gymnasium 1.2.3, 2026.

[FN18] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors, 2018.

[Fou26a] Python Software Foundation. Python3 language, 2026.

[Fou26b] The Linux Foundation. pytorch 2.10.0, 2026.

[GWOB19] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, December 2019.

[HLK+25] Kiwmann Hwang, Hyang-Tag Lim, Yong-Su Kim, Daniel K. Park, and Yosep Kim. Distributed quantum machine learning via classical communication. *QUANTUM SCIENCE AND TECHNOLOGY*, 10(1), JAN 1 2025.

[JGM+21] Sofiene Jerbi, Casper Gyurik, Simon C. Marshall, Hans J. Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning, 2021.

[JK22] Vibha Jain and Bijendra Kumar. Blockchain enabled trusted task offloading scheme for fog computing: A deep reinforcement learning approach. *Transactions on Emerging Telecommunications Technologies*, 33, 06 2022.

[JTPN+21] Sofiene Jerbi, Lea M. Trenkwalder, Hendrik Poulsen Nautrup, Hans J. Briegel, and Vedran Dunjko. Quantum enhancements for deep reinforcement learning in large spaces. *PRX Quantum*, 2(1), February 2021.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[LS20] Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuits, 2020.

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

[MPS25] Irshadullah Asim Mohammed, Prashant Pandey, and Sruthi S. The impact of ai on strategic decision making in modern management. *European Economics Letters*, 15:3770–3782, 11 2025.

[Sch26] SchedMD. Slurm, 2026.

[SVS24] Ekamdeep Singh, Prihana Vasishta, and Anju Singla. Ai-enhanced education: exploring the impact of ai literacy on generation z's academic performance in northern india. *Quality Assurance in Education*, 33(2):185–202, 08 2024.

[tea26] NumPy team. numpy 2.4.2, 2026.

[VML+19] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks, 2019.

[vN25] Evert van Nieuwenburg. Nordita-rl-4.ipynb, 2025.

[Xan26] Xanadu. Pennylane 0.44.0, 2026.

[ZG21] Chen Zhao and Xiao-Shan Gao. Analyzing the barren plateau phenomenon in training quantum neural networks with the zx-calculus. *Quantum*, 5:466, 2021.