

# Policy Gradients using Variational Quantum Circuits

André Sequeira<sup>1,2,3\*</sup>, Luis Paulo Santos<sup>1,2,3†</sup> and Luis Soares Barbosa<sup>1,2,3†</sup>

<sup>1</sup>\*Department of Informatics, University of Minho, Braga, Portugal.

<sup>2</sup>HASLab, INESC TEC, Braga, Portugal.

<sup>3</sup>International Nanotechnology Laboratory (INL), Braga, Portugal.

\*Corresponding author(s). E-mail(s): [andre.sequeira@inl.int](mailto:andre.sequeira@inl.int);  
Contributing authors: [psantos@di.uminho.pt](mailto:psantos@di.uminho.pt); [lsb@di.uminho.pt](mailto:lsb@di.uminho.pt);

†These authors contributed equally to this work.

## Abstract

Variational Quantum Circuits are being used as versatile Quantum Machine Learning models. Some empirical results exhibit an advantage in supervised and generative learning tasks. However, when applied to Reinforcement Learning, less is known. In this work, we considered a Variational Quantum Circuit composed of a low-depth hardware-efficient ansatz as the parameterized policy of a Reinforcement Learning agent. We show that an  $\epsilon$ -approximation of the policy gradient can be obtained using a logarithmic number of samples concerning the total number of parameters. We empirically verify that such quantum models behave similarly to typical classical neural networks used in standard benchmarking environments and quantum control, using only a fraction of the parameters. Moreover, we study the Barren Plateau phenomenon in quantum policy gradients using the Fisher information matrix spectrum.

**Keywords:** Quantum Machine Learning, Reinforcement Learning, Policy Gradients, Quantum Control

# 1 Introduction

Reinforcement Learning (RL) is responsible for many relevant developments in Artificial Intelligence (AI). Successes such as beating the world champion of Go [1] and solving numerous complex games without any human intervention [2] were relevant milestones in AI, providing optimal planning without supervision. RL is paramount in complex real-world problems such as self-driving vehicles [3], automated trading [4, 5], recommender systems [6], quantum physics [7], among many others. Recent advancements in RL are strongly associated with advances in Deep Learning [8] since scaling to large state/action space environments is possible, as opposed to tabular RL [9].

Previous results suggest that RL agents obeying the rules of quantum mechanics can outperform classical RL agents [10–15]. However, these suffer from the same scaling problem as classical tabular RL: they do not scale easily to real-world problems with large state-action spaces. Additionally, the lack of fault-tolerant quantum computers [16] further compromises the ability to handle problems of significant size.

Variational Quantum Circuits (VQCs) are a viable alternative since state-action pairs can be parameterized, enabling, at least in theory, a reduction in the circuit’s complexity. Moreover, VQCs could enable shallow enough circuits to be confidently executed on current NISQ (*Noisy Intermediate Scale Quantum*) hardware [17] without resorting to typical brute force search over the state/action space as in the quantum tabular setting [10, 13]. Variational models are also referred to as approximately universal quantum neural networks [18, 19]. Nevertheless, fundamental questions on the expressivity and trainability of VQCs remain to be answered, especially from a perspective relevant to RL.

This paper proposes an RL agent’s policy resorting to a shallow VQC and studies its effectiveness when embedded in the Monte-Carlo-based policy gradient algorithm REINFORCE [20] throughout standard benchmarking environments. However, benchmarking variational algorithms for classical environments exhibit a trade of information between a quantum and a classical channel that incurs an overhead from encoding classical information into the quantum processor. Efficient encoding of real-world data constitutes a real bottleneck for NISQ devices, with the consequence of neglecting any potential quantum advantage [21]. In the case of a quantum agent-environment interface, the cost of data encoding can often be neglected, and there is room for potential quantum advantages from quantum data [22]. In optimal quantum control, gate fidelity is improved by exploiting the full knowledge of the system’s Hamiltonian [23]. However, such methods are only viable when the system’s dynamics are known. Thus, applying variational quantum methods may indeed be relevant [24]. In this setting, we considered a quantum RL agent that optimizes the gate fidelity in a model-free setting, learning directly from the interface with the noisy environment.

The main contributions of this paper are:

- Design of a variational softmax-policy using a shallow VQC similar to or outperforming long-term cumulative reward compared to a restricted class of classical neural networks used in a set of standard benchmarking environments and the problem of quantum state preparation, using a fraction of the number of trainable parameters.
- Demonstration of a logarithmic sample complexity concerning the number of parameters in gradient estimation.
- Empirical verification of different parameter initialization strategies for variational policy gradients.
- Study of the barren plateau phenomenon in quantum policy gradient optimization using the Fisher information matrix spectrum.

The rest of the paper is organized as follows. Section 2 reviews quantum variational RL's state-of-the-art. Section 3 summarizes the theory behind the classical policy gradient algorithm used in this work. Section 4 details each block of the proposed VQC and the associated quantum policy gradient algorithm. Section 4.5 explores trainability under gradient-based optimization using quantum hardware and its corresponding sample complexity. Section 5 presents the performance of the quantum variational algorithm in simulated benchmarking environments. Section 6 analyzes the number of parameters trained and the Fisher Information spectrum associated with the classical/quantum policy gradient. Section 7 closes the paper with some concluding remarks and suggestions for future work.

## 2 Related Work

Despite numerous publications focusing on Quantum Machine Learning (QML), the literature on variational methods applied to RL remains scarce. Most results to date focus on value-based function approximation rather than policy-based. Chen et al. [25] use VQCs as quantum value function approximators for discrete state spaces, and, in [26], the authors generalize the former result to continuous state spaces. Lockwood et al. [27] show that simple VQC-inspired Q-Networks (i.e., state-action value approximators) based on Double Deep Q-Learning are not adequate for the Atari games, Pong and Breakout. Sanches et al. [28] proposed a hybrid quantum-classical policy-based algorithm to solve real-world problems like vehicle routing. In [29], the authors proposed a variational actor-critic agent, which is the only work so far operating on the quantum-quantum context of QML [30], i.e., a quantum agent acting upon a quantum environment. The authors suggest that the variational method could solve quantum control problems. Jerbi et al. [31] propose a novel quantum variational policy-based algorithm achieving better performance than previous value-based methods in a set of standard benchmarking environments. Their architecture consists of repeated angle-encoding to increase the expressivity of the variational model, i.e., increasing the number of functions of the input state that the model can represent [19]. Compared with [31], our work shows that a simpler variational architecture composed of a shallow ansatz,

consisting of a two-qubit entangling gate and two single-qubit gates [32] with a single encoding layer can be considered for standard benchmarking environments. Variational policies can be devised with decreased depth and fewer trainable parameters. The type of functions our circuit can represent is substantially smaller when compared to [31]. However, simpler classes of policies may be beneficial in the language of generalization and overfitting. Furthermore, compared to [31], this work considers a more trivial set of observables for the measurement of the quantum circuit, leading to fewer shots necessary to estimate the agent's policy and respective policy gradient.

### 3 Policy Gradients

Policy Gradient methods try to learn a parameterized policy  $\pi(a|s, \theta) = \mathbb{P}\{a_t = a | s_t = s, \theta_t = \theta\}$ , where  $\theta \in \mathbb{R}^k$  is the parameter vector of size  $k$ ,  $s$  and  $a$  are the state and action, respectively, and  $t$  is the time instant, that can optimally select actions without resorting to a value function. These methods try to maximize a performance measure  $J(\theta)$ , performing gradient ascent on  $J(\theta)$

$$\theta_{i+1} = \theta_i + \eta \nabla_{\theta_i} J(\theta_i) \quad (1)$$

where  $\eta$  is the learning rate. Provided that the action space is discrete and relatively small, then the most prominent way of balancing exploration and exploitation is by sampling an action from a *Softmax-Policy*, also known as Neural Policy [33]:

$$\pi(a|s, \theta) = \frac{e^{h(s,a,\theta)}}{\sum_{b \in A} e^{h(s,b,\theta)}} \quad (2)$$

where  $h(s, a, \theta) \in \mathbb{R}$  is a numerical preference for each state-action pair and  $A$  is the action set. For legibility,  $A$  will be omitted whenever a policy similar to equation (2) is presented. The policy gradient theorem [34] states that the gradient of the objective function can be written as a function of the policy itself. In general, the Monte-Carlo policy gradient known as REINFORCE [20], computes the gradient of samples obtained from  $N$  trajectories of length  $T$ , also known as the *horizon* under the parameterized policy, as in Equation (3).

$$\nabla_{\theta} J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} G_t(\tau_i) \nabla_{\theta} \log \pi(a_{t_i} | s_{t_i}, \theta) \quad (3)$$

where  $G_t(\tau)$  is the  $\gamma$ -discounted cumulative reward per time step, known as the *return* (see Equation (5)) derived from trajectory's return  $G(\tau)$  (see Equation (4)).

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \quad (4) \quad G_t(\tau) = \sum_{t'=0}^{T-t-1} \gamma^{t'} r_{t'+t} \quad (5)$$

A known limitation of the REINFORCE algorithm is due to Monte Carlo estimates. Stochastically sampling the trajectories results in gradient estimators with high variance, which deteriorate the performance as the environment's complexity increases [35]. The REINFORCE estimator can be improved by

leveraging a control variate known as *baseline*  $b(s_t)$ , without increasing the number of samples  $N$ . Baselines are subtracted from the return such that the optimization landscape becomes smooth. The REINFORCE with baseline gradient estimator is represented in Equation (6), and the complete algorithm is presented in Algorithm 1.

$$\nabla_{\theta} J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} (G_t(\tau_i) - b(s_{t_i})) \nabla_{\theta} \log \pi(a_{t_i} | s_{t_i}, \theta) \quad (6)$$

For the benchmarking environments in Section 5, the average return was used as a baseline, calculated as in equation (7).

$$b(s_t) = \frac{1}{N} \sum_{i=0}^{N-1} G_t(\tau_i) \quad (7)$$

---

**Algorithm 1** REINFORCE with baseline
 

---

**Require:**  $\theta \in \mathbb{R}^k$ , learning rate  $\eta$ , horizon  $T$

**while** True **do**

**for**  $i = 0 \dots N - 1$  **do**

    Following  $\pi_{\theta}$ , generate trajectory of the form

$$\tau_i = \{(s_0, a_0, r_0), \dots, (s_{T-1}, a_{T-1}, r_{T-1})\}$$

**end for**

  Compute gradient with baseline as in Equation (6)

  update parameters via gradient ascent  $\theta = \theta + \eta \nabla_{\theta} J(\theta)$

**end while**

---

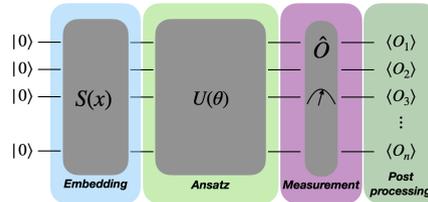
## 4 Quantum Policy Gradients

This section details the proposed VQC-based policy gradient. Numerical preferences  $h(s, a, \theta) \in \mathbb{R}$  are the output of measurements in a given parameterized quantum circuit. The result can be represented as the expectation value of a given observable or the probability of measuring a basis state. We resort to the former since it allows for more compact representations of objective functions [36]. Additionally, the type of ansatz used by the proposed VQC implies that  $\theta \in \mathbb{R}^k$  is a high dimensional vector corresponding to the angles of arbitrary single-qubit rotations.

VQCs are composed of four main building blocks, as represented in Figure 1. Initially, a state preparation routine or *embedding*,  $S$ , encodes data points into the quantum system. Next, a unitary  $U(\theta)$  maps the data into higher dimensions of the Hilbert space. Such a parameterized model corresponds to linear methods in quantum feature spaces. Expectation values returned from a measurement scheme are finally post-processed into the quantum neural policy.

## 6 Policy Gradients using Variational Quantum Circuits

A careful analysis of each block of Figure 1 follows. Moreover, the sample complexity of estimating the quantum policy gradient is analyzed in Section 4.5.



**Fig. 1** Building blocks of Variational Quantum Circuits.

## 4.1 Embedding

Unlike classical algorithms, the state-preparation routine is a crucial step for any variational quantum algorithm. There are numerous ways of encoding classical data into a quantum processor [37]. Angle encoding [21] is used to allow continuous-state spaces. Arbitrary Pauli rotations  $\sigma \in \{\sigma_x, \sigma_y, \sigma_z\}$  can encode a single feature per qubit. Hereby, given an agent's state  $s$  with  $n$  features,  $s = \{s_0, s_2, \dots, s_{n-1}\}$ ,  $\sigma_x$  rotations are used, requiring  $n$  qubits to encode  $|s\rangle$ , as indicated by Equation (8).

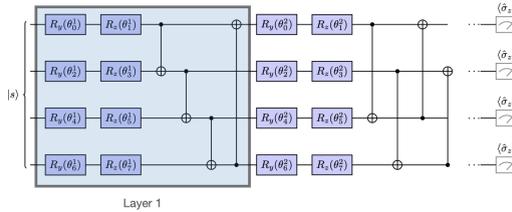
$$|s\rangle = \bigotimes_{i=0}^{n-1} e^{-j\sigma_x s_i} |b_i\rangle \quad (8)$$

where  $|b_i\rangle$  refers to the  $i^{\text{th}}$  qubit of an  $n$ -qubit register initially in state  $|0^n\rangle$  (represented w.l.g as  $|0\rangle$  from now on). Each feature needs to be normalized such that  $s_i \in [-\pi, \pi]$ . Since the range of each feature is usually unknown, this work resorts to normalization based on the  $L_\infty$  norm. The main advantage of angle encoding lies in the simplicity of generating the encoding, given the composition of solely  $n$  single-qubit gates, thus giving rise to a circuit of depth 1. In contrast, the main disadvantage is the linear dependence between the number of qubits and the number of features characterizing the agent's state and the poor representational power, at least in principle [38].

## 4.2 Parameterized model

To the best of the authors' knowledge, no *problem-inspired* ansatz exploiting the physics behind the problem is known in RL applications. This can be explained by the difficulty of expressing and training RL agent's policies as Hamiltonian-based evolution models [36]. Moreover, since the goal is to design a NISQ ansatz to capture the agent's optimal policy in different environments, this work uses a parameterized model from the family commonly referred to as *hardware-efficient* ansatz [36]. Such models behave similarly to a classical

feed-forward neural network. The main advantage of this family of ansatz is its versatility, accommodating encoding symmetries and bringing correlated qubits closer for depth reduction [39]. The ansatz consists of an alternating-layered architecture composed of single-qubit gates followed by a cascade of entangling gates as pictured in Figure 2.



**Fig. 2** Hardware-efficient ansatz for RL based on single-qubit  $R_y, R_z$  rotation gates.

A single layer is composed of two single-qubit  $\sigma_y, \sigma_z$  rotation gates per qubit, followed by a cascade of entangling gates, such that features are correlated in a highly entangled state. The ansatz includes  $2n$  single-qubit rotation gates per layer, each gate parameterized by a given angle. Therefore, there are  $2nL$  trainable parameters for  $L$  layers. The entangling gates follow a pattern that changes over the number of layers, inspired by the circuit-centric classifier design [19]. The pattern follows a modular arithmetic CNOT[ $i, (i + l) \bmod n$ ] where  $i \in [1, \dots, n]$  and  $l \in [1, \dots, L]$  indexes the layers. Increasing the number of layers increases the correlation between features and expressivity.

### 4.3 Measurement

An arbitrary state  $|\psi\rangle \in \mathbb{C}^{2^n}$  is represented by an arbitrary superposition over the basis states, as in Equation (9).

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |\psi_i\rangle \quad (9)$$

Measuring the state  $|\psi\rangle$  in the computational basis ( $\sigma_z$  basis) collapses the superposition into one of the basis states  $|\psi_i\rangle$  with probability  $|c_i|^2$ , as given by the Born rule [40]. In general, the expectation value of some observable  $\hat{O}$ , is given by the summation of each possible outcome, i.e., the eigenvalue  $\lambda_i$  weighted by its respective probability  $p_i = |c_i|^2$  as in Equation (10).

$$\langle \hat{O} \rangle = \langle \psi | \hat{O} | \psi \rangle = \sum_{i=0}^{2^n-1} \lambda_i p_i \quad (10)$$

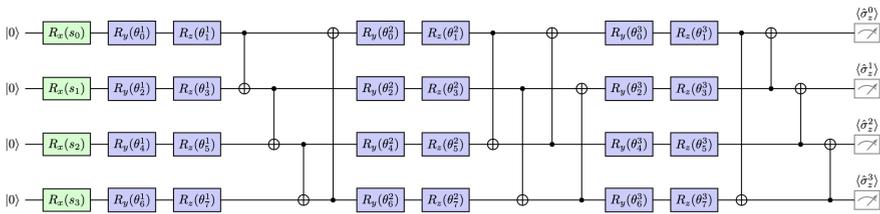
Let  $\hat{O}$  be the single-qubit  $\sigma_z^i$  measurement, applied to the  $i^{\text{th}}$ -qubit. Given that the  $\sigma_z$  eigenvalues are  $\{-1, 1\}$ , the expectation value  $\langle \sigma_z^i \rangle$  can be obtained by the probability  $p_0$  of the qubit being in the state  $|0\rangle$  as  $\langle \sigma_z^i \rangle = 2p_0 - 1$ . Notice

## 8 Policy Gradients using Variational Quantum Circuits

that in practice,  $p_0$  needs to be estimated from several circuit repetitions to obtain an accurate estimate of the expectation value. Let the state  $|\psi\rangle$  be the quantum state obtained from the encoding of an agent's state via  $S(s)$ , and the parameterized block  $U(\theta)$ , as in Sections 4.1 and 4.2 respectively. Let  $\langle\sigma_z^i\rangle$  be the quantum analogue of the numerical preference for action  $i$ , which we represent by  $\langle a_i\rangle$  for clarity. Its expectation can be formally described by Equation (11).

$$\langle a_i\rangle_\theta = \langle 0|S(s)^\dagger U(\theta)^\dagger \sigma_z^i U(\theta) S(s)|0\rangle \quad (11)$$

For a policy with  $|A|$  possible actions, each  $\sigma_z$  measurement corresponds to the numerical preference of each action. Thus,  $|A|$  single-qubit estimated expectation values are needed. If the number of features in the agent's state is larger than the number of actions, the single-qubit measurements occur only on a subset of qubits. Such measurement scheme is qubit-efficient [37]. Figure 3 represents the full VQC for an environment with four feature states and four actions with three parameterized layers.



**Fig. 3** Variational Quantum Circuit for Policy-based RL with three parameterized layers.

#### 4.4 Classical Post-processing

Measurement outcomes representing numerical preferences  $h(s, a, \theta) = \langle a\rangle_\theta$  are classically post-processed to convert the estimated expectation values to the final quantum neural policy, as given by Equation (12).

$$\pi(a | s, \theta) = \frac{e^{\langle a\rangle_\theta}}{\sum_b e^{\langle b\rangle_\theta}} \quad (12)$$

Equation (12) imposes an upper bound on the greediness of  $\pi$ . It will always allow for exploratory behavior, which can negatively impact the performance of RL agents, especially in deterministic environments. As an example, consider a 2-action environment with

$$\pi = [\pi(a_0 | s, \theta), \pi(a_1 | s, \theta)]$$

The entries of  $\pi$  are given by Equation (12) and the actions' estimated expectation values  $[\langle a_0\rangle_\theta, \langle a_1\rangle_\theta]$ . As these are bounded as  $\langle\sigma_z\rangle \in [-1, 1]$ , the maximum

difference between action preferences occurs when the estimated vector is  $[\langle a_0 \rangle_\theta = -1, \langle a_1 \rangle_\theta = 1]$ . The corresponding softmax normalized vector is:

$$\pi_a = [\pi(a_0 | s, \theta), \pi(a_1 | s, \theta)] = [0.88, 0.12]$$

In this case, the policy always has a  $\sim 0.1$  probability of selecting the worst action; the same *rationale* applies to larger action sets. Thus, a trainable parameter  $\beta$  is added to the quantum neural policy as in Equation (13):

$$\pi(a|s, \theta) = \frac{e^{\beta \langle a \rangle_\theta}}{\sum_b e^{\beta \langle b \rangle_\theta}} \quad (13)$$

$\beta$  has the effect of scaling the output values from the quantum circuit measurements, resembling an energy-based model. Instead of decreasing  $\beta$  over time, we treat it as a hyperparameter to be tuned along with  $\theta$ . The optimization sets  $\beta$ , assuring convergence towards the optimal policy.

## 4.5 Gradient Estimation

This section develops upper bounds on both the number of samples and the number of circuit evaluations necessary to obtain an  $\epsilon$ -approximation of the policy gradient, as given by Equation (3), restated here for completion:

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} G_t(\tau_i) \nabla_\theta \log \pi(a_{t_i} | s_{t_i}, \theta)$$

The gradient  $\nabla_\theta J(\theta)$  can be estimated using the same quantum device that computes expectations  $\langle a_i \rangle_\theta$ , via parameter-shift rules [41]. These rules require the policy gradient to be framed as a function of gradients of observables, as given by Equation (14).

$$\nabla_\theta \log \pi(a|s, \theta) = \beta \left( \nabla_\theta \langle a \rangle_\theta - \sum_b \pi(b|s, \theta) \nabla_\theta \langle b \rangle_\theta \right) \quad (14)$$

By combining equations (3) and (14), the quantum policy gradient estimator is given by Equation (15):

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} G_t(\tau_i) \beta \left( \nabla_\theta \langle a_{t_i} \rangle_\theta - \sum_{b_{t_i}} \pi(b_{t_i} | s_{t_i}, \theta) \nabla_\theta \langle b_{t_i} \rangle_\theta \right) \quad (15)$$

The number of samples associated with Equation (15) is defined as the number of visited states. Since there are  $N$  trajectories (sequences of actions,  $\tau_i$ ), each visiting  $T$  states, the total number of samples is  $\mathcal{O}(NT)$ . Lemma 4.1 provides an upper bound for  $N$  such that the policy gradient is  $\epsilon_\nabla$ -approximated with probability  $1 - \delta_\nabla$ .

**Lemma 4.1** ( $\epsilon_{\nabla}$ -approximation of the policy-gradient). *Let  $\theta \in \mathbb{R}^k$ ,  $k$  being the number of parameters,  $R_{max}$  be the maximum possible reward in any time step,  $T$  the horizon, and  $\nabla_{\theta}J(\theta)$  the expected policy gradient. The policy gradient,  $\hat{\nabla}_{\theta}J(\theta)$ , can be  $\epsilon_{\nabla}$ -approximated, with probability  $1 - \delta_{\nabla}$*

$$|\hat{\nabla}_{\theta}J(\theta) - \nabla_{\theta}J(\theta)| \leq \epsilon_{\nabla} \quad (16)$$

using a number of samples given by

$$NT \approx \mathcal{O} \left( \frac{8\beta^2 R_{max}^2 T^3}{\epsilon_{\nabla}^2 (\gamma - 1)^4} \log \left( \frac{2k}{\delta_{\nabla}} \right) \right) \quad (17)$$

The most relevant insight drawn from Lemma 4.1 is that it establishes that for obtaining an  $\epsilon_{\nabla}$ -approximated policy gradient, the algorithm needs a number of samples that grows logarithmically with the total number of parameters. The proof of Lemma 4.1 is presented in detail in Appendix A.1.

Gradient-based optimization can be performed using the same quantum device that computes expectations  $\langle a_i \rangle_{\theta}$ , via parameter-shift rules [41, 42], which compute the gradient of an observable w.r.t a single variational parameter concerning rotation angles of quantum gates. Parameter-shift rules are given by Equation (18):

$$\nabla_{\theta_i} \langle a_i \rangle_{\theta} = \frac{1}{2} [\langle a_i \rangle_{\theta + \frac{\pi}{2}} - \langle a_i \rangle_{\theta - \frac{\pi}{2}}] \quad (18)$$

The gradient's accuracy depends on the expectation values,  $\langle a \rangle_{\theta}$ . These are estimated for each sample and action using several repetitions of the quantum circuit or shots. Lemma 4.2 establishes an upper bound on the total number of shots required to reach an  $\epsilon_{\diamond}$ -approximated policy gradient, with probability  $1 - \delta_{\diamond}$ .

**Lemma 4.2** (Total number of quantum circuit evaluations). *Let  $\theta \in \mathbb{R}^k$ ,  $\mathcal{O}(NT)$  be the sample complexity given by Lemma 4.1, and  $|A|$  the number of available actions. With probability  $1 - \delta_{\diamond}$  and approximation error  $\epsilon_{\diamond}$ , the quantum policy gradient algorithm requires a number of shots given by*

$$\mathcal{O} \left( \frac{|A|NT}{\epsilon_{\diamond}^2} \log \left( \frac{2k}{\delta_{\diamond}} \right) \right) \quad (19)$$

Similarly to Lemma 4.1, it is shown that the accuracy of the policy gradient, as a function on the total number of shots, grows logarithmically with the total number of parameters. The proof of Lemma 4.2 is presented in detail in Appendix A.2.

## 5 Performance in simulated environments

This section examines the performance of the proposed quantum policy gradient through standard benchmarking environments from the OpenAI Gym library [43]. Moreover, the quantum policy gradient was also tested in a hand-crafted quantum control environment. In this setting, a quantum agent was designed to learn to prepare the state  $|1\rangle$  with high fidelity, starting from the ground state  $|0\rangle$ . The empirical reward over the number of episodes was used to discern the performance of both classical and quantum models. The best-performing classical neural network was selected from a restrictive set of networks composed of at most two hidden linear layers. All quantum circuits were built using the PennyLane library [44] and trained using the PyTorch automatic differentiation backend [45] to be directly compared with classical models built with the same library. All training instances used the most common classical optimizer, ADAM [46].

### 5.1 Numerical Experiments

The CartPole-v0 and Acrobot-v1 environments were selected as classic benchmarks. They have a continuous state space with a relatively small feature space (2 to 6 features) and discrete action space (2 to 3 possible actions). The reward function is similar to each environment. In Cartpole, the agent receives a reward of +1 at every time step. The more time the agent keeps the pole from falling, the more reward it gets. In Acrobot, the agent receives a  $-1$  reward at every time step and reward 0 once it gets to the goal state. Thus, Acrobot will be harder to master since, for the Cartpole, every action has an immediate effect as opposed to Acrobot.

In the quantum control environment of state preparation, which we refer to as QControl on this point onward, for simplicity, the mapping  $|0\rangle \mapsto |1\rangle$  can be characterized by a time-dependent Hamiltonian  $H(t)$  of the form of Equation (20) describing the quantum environment as in [47].

$$H(t) = 4J(t)\sigma_z + h\sigma_x \quad (20)$$

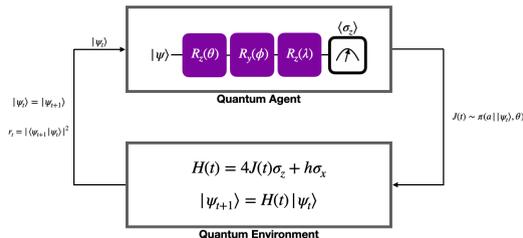
Where  $h$  represents the single-qubit energy gap between tunable control fields, considered a constant energy unit.  $J(t)$  represents the dynamical pulses controlled by the RL quantum agent in a model-free setting. The learning procedure defines a fixed number of steps  $N = 10$ , from which the RL agent must be able to create the desired quantum state. The quantum environment prepares the state associated with the time step  $t + 1$ , given the gate-based Hamiltonian at time step  $t$ ,  $U(t)$ :

$$|\psi_{t+1}\rangle = U(t)|\psi\rangle \quad (21)$$

The reward function is naturally represented as the fidelity between the target state  $|\psi_T\rangle = |1\rangle$  and the prepared state  $|\psi_t\rangle$  naturally serves as the reward  $r_t$  for the agent at time step  $t$ , as in Equation (22).

$$r_t = |\langle \psi_t | \psi_T \rangle|^2 \quad (22)$$

Using the policy gradient algorithm of Section 4, the goal is to learn how to maximize fidelity. Figure 4 depicts the agent-environment interface.



**Fig. 4** Agent-Environment interface for quantum control.

Each sequence of  $N$  pulses corresponds to an episode. The quantum agent should learn the optimal pulse sequence that maps to the state with maximum fidelity as the number of episodes increases. The quantum variational architecture selected was the same as described in Section 4. In this setting, the main difference is the lack of encoding. The quantum agent receives the quantum state from the corresponding time-step Hamiltonian applied at each time step. However, since the environment is simulated, the qubit is prepared in the state of time step  $t$  and then fed to the variational quantum policy. In this setting, it is considered the binary action-space  $A = [0, 1]$  (apply pulse  $A = 1$  or not,  $A = 0$ ). A sequence of  $N$  actions corresponds to  $N$  pulses. A performance comparison is made relative to classical policy gradients. In this case, the corresponding state vector associated with the qubit was explicitly encoded at each time step, considering both real and imaginary components. All environment specifications are presented in Table 1.

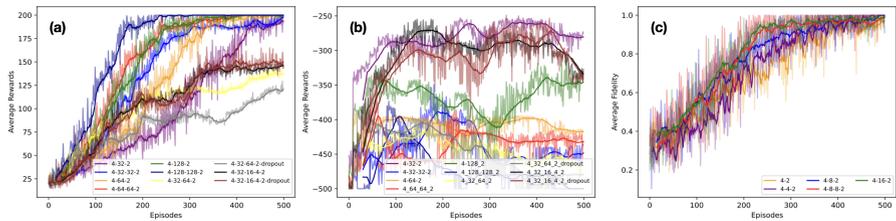
Environment	#F	#A	Reward (per step)	Max #s (per episode)	Terminal states
CartPole-v0	4	2	1	200	Out of bounds or reward 200 or below horizontal line
Acrobot-v1	6	3	-1	500	500 steps
QControl	4	2	$ \langle \psi_t   \psi \rangle ^2$	10	$ \langle \psi_t   \psi \rangle ^2 \leq 10^{-4}$ or 10 steps

**Table 1** Description of the environments (#F: number of features; #A: number of actions; Max #s: maximum steps).

Several neural network architectures were tested for the CartPole-v0 and Acrobot-v1 environments. However, the structure is the same. Every neural network is composed of fully connected layers using a rectified linear unit

(ReLU) activation function in every neuron. The output layer is the only layer that does not have ReLU activation. The depth, the total number of trainable parameters, and the existence of dropout differs from network to network. All the networks using dropout have a probability equal to 0.2. Every network was trained with an ADAM optimizer with an experimentally fine-tuned learning rate of 0.01. Figures 5(a) and 5(b) illustrate the average reward for different classical network configurations for the benchmarking environments. The results show that a fully connected neural network with a single layer of 128 and 32 neurons performs reasonably better than similar architectures for the CartPole-v0 and Acrobot-v1 environments, respectively.

In the QControl environment, eight different neural networks were tested with a single hidden layer. Since the optimal neural network for this problem is still an open question, to the best of the author’s knowledge, it was decided to successively increase the size of the network until it solves the task of comparing the minimum viable network with the VQC. For this set of classical architectures, the neural network with a single layer of 16 neurons was chosen since it achieves the best average fidelity as the minimum viable network solving the problem, as illustrated in Figure 5(c).



**Fig. 5** Different classical neural network architectures used in the three simulated environments. Panels (a), (b), and (c) represent different architectures for the cartpole, Acrobot, and QControl environments, respectively. Each label indicates the respective network structure and if it uses dropout. Each label represents the total number of neurons in each input, hidden, and output layer. E.g., 4 – 4 – 4 has input, hidden, and output layers with four neurons each.

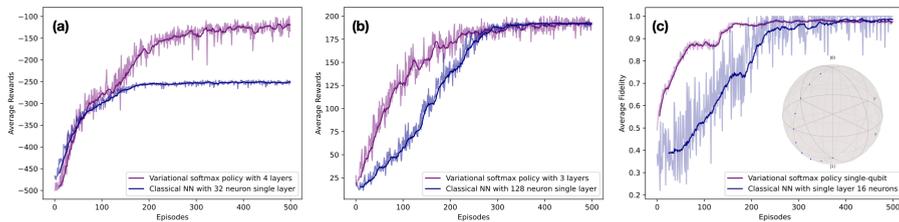
The second step compares the performance of the quantum neural policy of Section 4 against the aforementioned classical architecture. Increasing the number of layers in the parameterized quantum model would perhaps increase the expressivity of the model [38]. At the same time, increasing the number of layers leads to more complex optimization tasks, given that more parameters need to be optimized. For some variational architectures, there is a threshold for expressivity in terms of the number of layers [48]. We encountered precisely this in practice. For Cartpole, the expressivity of the quantum neural policy saturates after three layers, and for the Acrobot, after four layers. From there on, the agent’s performance deteriorated rather than improved. For the QControl environment, the classical NN was compared with a simplified version of the variational softmax policy. In this case, it was considered a VQC with

the most general gate with three parameters that can approximately prepare every single-qubit state. The observables for the numerical action preference are the opposite sign computational basis measurement, i.e.,  $[\langle\sigma_z\rangle, -\langle\sigma_z\rangle]$ . In every environment, the model's learning rate was fine-tuned by trial and error as opposed to  $\beta$ , which was randomly initialized. The optimal configuration for the learning rate, number of layers, and batch size used to compare are presented in table 2.

Environment	Policy	Learning rate	#Layers	Batch size
CartPole-v0	Quantum	0.1	3	10
CartPole-v0	Classical	0.01	-	10
Acrobot-v1	Quantum	0.1	4	10
Acrobot-v1	Classical	0.01	-	10
QControl	Quantum	0.01	1	10
QControl	Classical	0.01	-	10

**Table 2** Specification for hyperparameter, number of layers, and batch size used for the classical and quantum neural policies in the three simulated environments.

Figures 6(a), 6(b) and 6(c) compare the average cumulative reward through several episodes for quantum and classical neural policies for the Cartpole, Acrobot, and QControl environments, respectively. A running mean was plotted to smooth the reward curves since the policy and environments are noisy. Figure 6(c) also plots the respective control trajectory obtained by the variational quantum policy.



**Fig. 6** Average cumulative reward. Comparison between the variational softmax policy and the respective classical NN. Panels (a), (b), and (c) represent the average reward comparison for the Cartpole, Acrobot, and QControl environments, respectively.

One can conclude that the quantum and classical neural policies perform similarly in every environment. In the QControl environment, the classical policy achieves a slightly greater cumulative reward. Nonetheless, there is clear evidence that the quantum-inspired policy needs fewer interactions with the environment to converge to near-optimal behavior. Moreover, the total number of trainable parameters for the quantum and classical models is summarized in the Table 3. The input layer of a classical neural network is related

to the number of qubits in a quantum circuit. Furthermore, we take the number of layers in the VQC as the number of hidden layers in a classical neural network. Given that the quantum circuit is unitary, the number of neurons in a quantum neural network is constant, i.e., equal to the system's number of qubits. Thus, one can conclude that the quantum policy has similar or even outperforming behavior compared to the classical policy with an extremely reduced total number of trainable parameters.

Env	Policy	I	O	#N	#R	$\beta$	#P
CartPole-v0	Quantum	4	2	—	2	Yes	25
CartPole-v0	Classical	4	2	128	—	No	768
Acrobot-v1	Quantum	6	3	—	2	Yes	33
Acrobot-v1	Classical	6	3	32	—	No	288
QControl	Quantum	1	1	—	3	Yes	3
QControl	Classical	4	2	16	—	No	96

**Table 3** Number of parameters trained for both environments (**Env**: environment; **I**: Input layer; **O**: Output layer; **#N**: neurons; **#R**: rotations per qubit; **#P**: parameters).

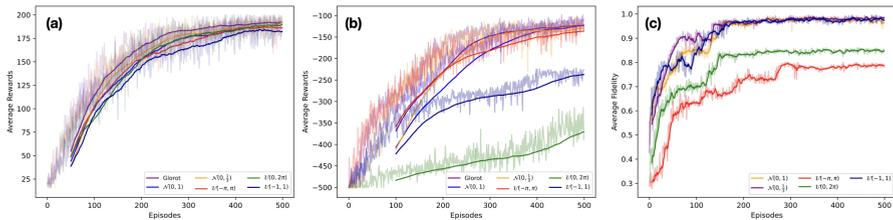
## 5.2 The effect of initialization

The parameters' initialization strategy can dramatically improve the convergence of a machine learning algorithm. Random initialization is often used to break the symmetry between different neurons [8]. However, if the parameters are arbitrarily large, the activation function may saturate, diffculting the learning task. Therefore, parameters are often drawn from specific distributions. For instance, the Glorot [49] initialization strategy is among the most commonly used to balance initialization and regularization [8]. In quantum machine learning models, the problem persists. However, it was verified experimentally that the Glorot initialization has a slight advantage compared to other strategies. The empirical results reported in Section 5.1 were obtained using such a strategy. The Glorot strategy samples the parameters of the network from a normal distribution  $\mathcal{N}(0, std^2)$  with standard deviation given by Equation (23):

$$std = gain * \sqrt{\frac{6}{n_{in} + n_{out}}} \quad (23)$$

where *gain* is a constant multiplicative factor.  $n_{in}$  and  $n_{out}$  are the number of inputs and outputs of a layer, respectively. It was devised to initialize all layers with approximately the same activation and gradient variance, assuming that the neural network does not have nonlinear activations, being thus reducible to a chain of matrix multiplications. The latter assumption motivates this strategy in quantum learning models since they are composed of unitary layers without nonlinearities. The only nonlinearity is introduced by

the measurement [40]. Figures 7(a), 7(b) and 7(c) plot the average reward obtained by the quantum agent in the CartPole, Acrobot and QControl environments, respectively, following the most common initialization strategies. Glorot initialization has a slightly better performance and stability. Moreover, it is verified empirically that for policy gradients, initialization from normal distributions generates better results for the classic environments compared to uniform distributions, as reported in [50] for standard machine learning cost functions. However, in the QControl task was not observed the same behavior since uniform sampling  $U(-1, 1)$  achieves similar performance than  $N(0, 1)$ .



**Fig. 7** Normal and Uniform distributions used to initialize the parameters of the variational softmax policy. Panels (a), (b), and (c) represent the average reward comparison for the Cartpole, Acrobot, and QControl environments, respectively.

## 6 Quantum enhancements

In this section, further steps are taken toward studying the possible advantages of quantum RL agents following two different strategies:

- **Parameter Count** - Comparison between quantum and classical agents regarding the number of parameters trained. It is unclear whether this is a robust approach to quantify advantage, given that the number of parameters alone can be misleading. For example, the function  $\sin(\theta)$  has a single parameter and is more complex than polynomial  $ax^3 + bx^2 + cx + d$ . However, having smaller networks could enable solutions for more significant problems at a smaller cost. Even though only parameter-shift rules are allowed on real quantum hardware, it enables a lower cost on memory than backpropagation. Perhaps the training difference may be negligible from a tradeoff between memory and time consumption for large enough problems. As reported in Table 3, a massive reduction in the number of parameters in the quantum neural network compared with the classical counterpart for all three simulated environments.
- **Fisher Information** - The Fisher Information matrix spectrum is related to the effect of barren plateaus in the optimization surface itself. Studying the properties of the matrix eigenvalues should help to explain the hardness of training.

The Fisher Information [51] is crucial both in computation and statistics as a measure of the amount of information in a random variable  $X$  in a statistical model parameterized by  $\theta$ . Its most general form amounts to the negative Hessian of the log-likelihood. Suppose a datapoint  $x$  sampled i.i.d from  $p(x|\theta)$  where  $\theta \in \mathbb{R}^k$ . Since the Hessian reveals information about the curvature of a function, the Fisher Information Matrix (see Equation (24)) captures the sensitivity concerning changes in the parameter space, i.e., changes in the curvature of the loss function.

$$F(\theta) = \mathbb{E}_{x \sim p} [\nabla_{\theta} \log p(x|\theta) \nabla_{\theta} \log p(x|\theta)^{\top}] \in \mathbb{R}^{k \times k} \quad (24)$$

The Fisher Information matrix is computationally demanding to obtain. Thus, the empirical Fisher information matrix is usually used in practice and can be computed as in Equation (25):

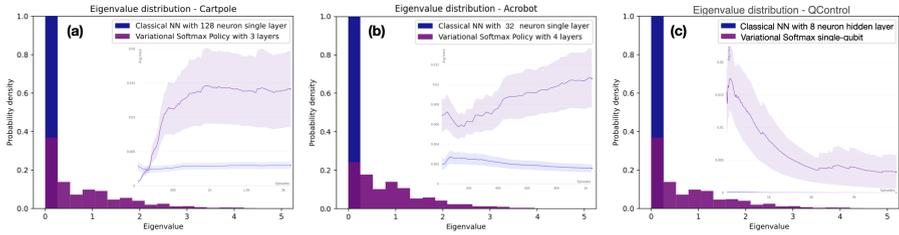
$$F(\theta) = \frac{1}{T} \sum_{i=1}^T \nabla_{\theta} \log p(x_i|\theta) \nabla_{\theta} \log p(x_i|\theta)^{\top} \quad (25)$$

Equation (25) captures the curvature of the score function at all parameter combinations. That is, it can be used as a measure for studying barren plateaus in maximum likelihood estimators [52], given that all the matrix entries will approach zero with the flatness of the model's landscape. This effect is captured by looking at the spectrum of the matrix. If the model is in a barren plateau, then the eigenvalues of the matrix will approach zero [53]. In the context of policy gradients, the empirical Fisher information matrix [54] is obtained by multiplying the vector resultant of the gradient of the log-policy with its transpose as in Equation (26):

$$F(\theta) = \frac{1}{T} \sum_{t=1}^T \nabla_{\theta} \log \pi(a_t|s_t, \theta) \nabla_{\theta} \log \pi(a_t|s_t, \theta)^{\top} \quad (26)$$

Inspecting the spectrum of the matrix in Equation (26) reveals the flatness of the loss landscape. Thus, it can harness the hardness of the model's trainability for both RL agents based on classical neural networks and VQCs [53]. This work considers the trace and the eigenvalues' probability density of the Fisher Information matrix. The trace will approach zero if the model is closer to a barren plateau and the eigenvalues' probability density unveils the magnitude of the associated eigenvalues.

Figures 8(a), 8(b) and 8(c) plot the average Fisher information matrix eigenvalue distribution for training episodes during the entire training for the CartPole, Acrobot and QControl environments, respectively. Subpanels in every plot indicate the associated information matrix trace. On average, the Fisher information matrix of the quantum model exhibits significantly larger density in eigenvalues different from zero compared to the classical model during the entire training. The same behavior is observed for every environment, explaining the improvement of the training performance for quantum agents (section 5) compared to classical ones. Although it is not visible from the eigenvalue distribution, the classical model has larger eigenvalues than the quantum



**Fig. 8** Probability density for the Fisher information matrix eigenvalues and average trace. Panels (a), (b), and (c) represent the eigenvalue distribution and trace of the Fisher information matrix for the Cartpole, Acrobot, and QControl environments, respectively.

model. However, their density is extremely small, thus making it negligible in a distribution plot. Further analysis is required to understand the behavior of both classical and quantum agents thoroughly.

## 7 Conclusion

In this work, a VQC was embedded into the decision-making process of an RL agent, following the policy gradient algorithm, solving a set of standard benchmarking environments efficiently. Empirical results demonstrate that such variational quantum models behave similarly or even outperform several typically used classical neural networks. The quantum-inspired policy needs fewer interactions to converge to an optimal behavior, benefiting from a reduction in the total number of trainable parameters.

Parameter-shift rules were used to perform gradient-based optimization resorting to the same quantum model used to compute the policy. It was proved that the sample complexity for gradient estimation via parameter-shift rules grows only logarithmically with the number of parameters.

The Fisher Information spectrum was used to study the effect of barren plateaus in quantum policy gradients. The spectrum indicates that the quantum model comprises larger eigenvalues than its classical counterpart, suggesting that the optimization surface is less prone to plateaus.

Finally, it was verified that the quantum model could prepare a single-qubit state with high fidelity in fewer episodes than the classical counterpart with a single layer.

Concerning future work, it would be interesting to apply such RL-based variational quantum models to quantum control problems of larger dimensions. Specifically, their application to noisy environments would be of general interest. Moreover, studying the expectation value of policy gradients given a specific initialization strategy to support empirical claims is crucial. At last, the quantum Fisher Information [55] should be addressed to analyze the information behind quantum states. Moreover, it would be interesting to embed the Quantum Fisher Information in a Natural Gradient optimization [56] to derive Quantum Natural Policy Gradients. Advanced RL models such as

Actor-Critic or Deep Deterministic Policy Gradients (DDPG) could benefit from quantum-aware optimization.

## Acknowledgements

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within grant LA/P/0063/2020, and project IBEX, with reference PTDC/CC1-COM/4280/2021.

## Declarations

### Conflict of interests

The authors declare no competing interests.

### Data availability

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

## References

- [1] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T.P., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016)
- [2] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. *Nature* **588**(7839), 604–609 (2020). <https://doi.org/10.1038/s41586-020-03051-4>
- [3] Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P.: Deep Reinforcement Learning for Autonomous Driving: A Survey (2021)
- [4] Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., Wang, C.D.: FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance (2020)
- [5] Mosavi, A., Ghamisi, P., Faghan, Y., Duan, P., Shamshirband, S.: Comprehensive review of deep reinforcement learning methods and applications in economics (2020). <https://doi.org/10.20944/preprints202003.0309.v1>

- [6] Afsar, M.M., Crump, T., Far, B.: Reinforcement learning based recommender systems: A survey (2021)
- [7] Dalgaard, M., Motzoi, F., Sørensen, J.J., Sherson, J.: Global optimization of quantum dynamics with alphazero deep exploration. *npj Quantum Information* **6**(1) (2020). <https://doi.org/10.1038/s41534-019-0241-0>
- [8] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, ??? (2016)
- [9] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA (2018)
- [10] Dunjko, V., Taylor, J.M., Briegel, H.J.: Quantum-Enhanced Machine Learning. *Physical Review Letters* **117**(13), 1–19 (2016) [arXiv:1610.08251](https://arxiv.org/abs/1610.08251). <https://doi.org/10.1103/PhysRevLett.117.130501>
- [11] Dunjko, V., Liu, Y.-K., Wu, X., Taylor, J.M.: Exponential improvements for quantum-accessible reinforcement learning (2017) [arXiv:1710.11160](https://arxiv.org/abs/1710.11160)
- [12] Paparo, G.D., Dunjko, V., Makmal, A., Martin-Delgado, M.A., Briegel, H.J.: Quantum speedup for active learning agents. *Physical Review X* **4**(3), 1–14 (2014) [arXiv:1401.4997](https://arxiv.org/abs/1401.4997). <https://doi.org/10.1103/PhysRevX.4.031002>
- [13] Sequeira, A., Santos, L.P., Barbosa, L.S.: Quantum tree-based planning. *IEEE Access* **9**, 125416–125427 (2021). <https://doi.org/10.1109/ACCESS.2021.3110652>
- [14] Dunjko, V., Briegel, H.J.: Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics* **81**(7), 074001 (2018). <https://doi.org/10.1088/1361-6633/aab406>
- [15] Saggio, V., Asenbeck, B.E., Hamann, A., Strömberg, T., Schiansky, P., Dunjko, V., Friis, N., Harris, N.C., Hochberg, M., Englund, D., et al.: Experimental quantum speed-up in reinforcement learning agents. *Nature* **591**(7849), 229–233 (2021). <https://doi.org/10.1038/s41586-021-03242-7>
- [16] Preskill, J.: Fault-tolerant quantum computation. *arXiv: Quantum Physics* (1997)
- [17] Preskill, J.: Quantum computing in the nisy era and beyond. *Quantum* **2**, 79 (2018). <https://doi.org/10.22331/q-2018-08-06-79>
- [18] Farhi, E., Neven, H.: Classification with Quantum Neural Networks on Near Term Processors (2018)

- [19] Schuld, M., Sweke, R., Meyer, J.J.: Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A* **103**(3) (2021). <https://doi.org/10.1103/physreva.103.032430>
- [20] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8**, 229–256 (2004)
- [21] LaRose, R., Coyle, B.: Robust data encodings for quantum classifiers. *ArXiv* **abs/2003.01695** (2020)
- [22] Huang, H.-Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., Mcclean, J.: Power of data in quantum machine learning. *Nature Communications* **12** (2021). <https://doi.org/10.1038/s41467-021-22539-9>
- [23] James, M.R.: Optimal quantum control theory. *Annual Review of Control, Robotics, and Autonomous Systems* **4**(1), 343–367 (2021) <https://doi.org/10.1146/annurev-control-061520-010444>. <https://doi.org/10.1146/annurev-control-061520-010444>
- [24] Martín-Guerrero, J., Lamata, L.: Reinforcement learning and physics. *Applied Sciences* **11**, 8589 (2021). <https://doi.org/10.3390/app11188589>
- [25] Chen, S.Y.C., Yang, C.H.H., Qi, J., Chen, P.Y., Ma, X., Goan, H.S.: Variational Quantum Circuits for Deep Reinforcement Learning. *IEEE Access* (2020) [arXiv:1907.00397](https://arxiv.org/abs/1907.00397). <https://doi.org/10.1109/ACCESS.2020.3010470>
- [26] Lockwood, O., Si, M.: Reinforcement learning with quantum variational circuits. In: *Proceedings of the 16th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2020* (2020)
- [27] Lockwood, O., Si, M.: Playing atari with hybrid quantum-classical reinforcement learning. (2021)
- [28] Sanches, F., Weinberg, S., Ide, T., Kamiya, K.: Short Quantum Circuits in Reinforcement Learning Policies for the Vehicle Routing Problem (2021)
- [29] Wu, S., Jin, S., Wen, D., Wang, X.: Quantum reinforcement learning in continuous action space (2021)
- [30] Aïmeur, E., Brassard, G., Gambs, S.: Machine learning in a quantum world. In: Lamontagne, L., Marchand, M. (eds.) *Advances in Artificial Intelligence*, pp. 431–442. Springer, Berlin, Heidelberg (2006)
- [31] Jerbi, S., Gyurik, C., Marshall, S., Briegel, H.J., Dunjko, V.: Variational quantum policies for reinforcement learning (2021)

- [32] Bharti, K., Cervera-Lierta, A., Kyaw, T.H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J.S., Menke, T., Mok, W.-K., Sim, S., Kwek, L.-C., Aspuru-Guzik, A.: Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **94**, 015004 (2022). <https://doi.org/10.1103/RevModPhys.94.015004>
- [33] Agarwal, A., Jiang, N., Kakade, S.: Reinforcement learning: Theory and algorithms. (2019)
- [34] Sutton, R., McAllester, D.A., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS (1999)
- [35] Greensmith, E., Bartlett, P.L., Baxter, J.: Variance reduction techniques for gradient estimates in reinforcement learning. *J. Mach. Learn. Res.* **5**, 1471–1530 (2004)
- [36] Bharti, K., Cervera-Lierta, A., Kyaw, T.H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J.S., Menke, T., Mok, W.-K., Sim, S., Kwek, L.-C., Aspuru-Guzik, A.: Noisy intermediate-scale quantum (NISQ) algorithms (2021)
- [37] Schuld, M., Petruccione, F.: Supervised Learning with Quantum Computers, 1st edn. Springer, ??? (2018)
- [38] Schuld, M.: Quantum machine learning models are kernel methods. (2021)
- [39] Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., McClean, J.R., Mitarai, K., Yuan, X., Cincio, L., Coles, P.J.: Variational quantum algorithms. *Nature Reviews Physics* **3**(9), 625–644 (2021). <https://doi.org/10.1038/s42254-021-00348-9>
- [40] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition, 10th edn. Cambridge University Press, USA (2011)
- [41] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J.A., Killoran, N.: Evaluating analytic gradients on quantum hardware. *Physical Review A* **99**, 032331 (2019)
- [42] Sweke, R., Wilde, F., Meyer, J., Schuld, M., Faehrmann, P.K., Meynard-Piganeau, B., Eisert, J.: Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum* **4**, 314 (2020). <https://doi.org/10.22331/q-2020-08-31-314>
- [43] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J.,

- Tang, J., Zaremba, W.: OpenAI Gym (2016)
- [44] Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Alam, M.S., Ahmed, S., Arrazola, J.M., Blank, C., Delgado, A., Jahangiri, S., McKiernan, K., Meyer, J.J., Niu, Z., Száva, A., Killoran, N.: PennyLane: Automatic differentiation of hybrid quantum-classical computations (2020)
- [45] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
- [46] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2017)
- [47] Zhang, X.-M., Wei, Z., Asad, R., Yang, X.-C., Wang, X.: When does reinforcement learning stand out in quantum control? a comparative study on state preparation. *npj Quantum Information* **5**, 1–7 (2019). <https://doi.org/10.1038/s41534-019-0201-8>
- [48] Sim, S., Johnson, P.D., Aspuru-Guzik, A.: Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies* **2**(12), 1900070 (2019). <https://doi.org/10.1002/qute.201900070>
- [49] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010)
- [50] Zhang, K., Hsieh, M.-H., Liu, L., Tao, D.: Gaussian initializations help deep variational quantum circuits escape from the barren plateau. arXiv (2022). <https://doi.org/10.48550/ARXIV.2203.09376>. <https://arxiv.org/abs/2203.09376>
- [51] Ly, A., Marsman, M., Verhagen, J., Grasman, R., Wagenmakers, E.-J.: A Tutorial on Fisher Information (2017)
- [52] Karakida, R., Akaho, S., Amari, S.-i.: Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach (2019)
- [53] Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., Woerner, S.: The power of quantum neural networks. *Nature Computational Science* **1**(6), 403–409 (2021). <https://doi.org/10.1038/s43588-021-00084-1>
- [54] Kakade, S.: A natural policy gradient. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. NIPS'01, pp. 1531–1538. MIT Press, Cambridge, MA, USA (2001)

- [55] Meyer, J.J.: Fisher information in noisy intermediate-scale quantum applications. *Quantum* **5**, 539 (2021). <https://doi.org/10.22331/q-2021-09-09-539>
- [56] Stokes, J., Izaac, J., Killoran, N., Carleo, G.: Quantum natural gradient. *Quantum* **4**, 269 (2020). <https://doi.org/10.22331/q-2020-05-25-269>
- [57] Hoeffding, W.: Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* **58**(301), 13–30 (1963) <https://www.tandfonline.com/doi/pdf/10.1080/01621459.1963.10500830>. <https://doi.org/10.1080/01621459.1963.10500830>

## A Upper bounds on gradient estimation

This appendix develops the proofs for Lemmas 4.1 and 4.2, as presented in section 4.5.

### A.1 $\epsilon_{\nabla}$ -approximation of the policy-gradient

Lemma 4.1 establishes an upper bound on the number of samples required to  $\epsilon_{\nabla}$ -estimate the policy gradient  $\hat{\nabla}_{\theta}J(\theta)$ .

**Lemma 4.1** ( $\epsilon_{\nabla}$ -approximation of the policy-gradient). *Let  $\theta \in \mathbb{R}^k$ ,  $k$  being the number of parameters,  $R_{max}$  be the maximum possible reward in any time step,  $T$  the horizon, and  $\nabla_{\theta}J(\theta)$  the expected policy gradient. The policy gradient,  $\hat{\nabla}_{\theta}J(\theta)$ , can be  $\epsilon_{\nabla}$ -approximated, with probability  $1 - \delta_{\nabla}$*

$$|\hat{\nabla}_{\theta}J(\theta) - \nabla_{\theta}J(\theta)| \leq \epsilon_{\nabla} \quad (16)$$

using a number of samples given by

$$NT \approx \mathcal{O} \left( \frac{8\beta^2 R_{max}^2 T^3}{\epsilon_{\nabla}^2 (\gamma - 1)^4} \log \left( \frac{2k}{\delta_{\nabla}} \right) \right) \quad (17)$$

*Proof* The policy gradient is estimated by resorting to Monte Carlo techniques, as described by Equation (15), restated here for completion.

$$\nabla_{\theta}J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} G_t(\tau_i)\beta \left( \nabla_{\theta}\langle a_{t_i} \rangle_{\theta} - \sum_{b_{t_i}} \pi(b_{t_i}|s_{t_i}, \theta) \nabla_{\theta}\langle b_{t_i} \rangle_{\theta} \right)$$

Recall that the number of samples is defined as the number of visited states. Since there are  $N$  trajectories (sequences of actions,  $\tau_i$ ), each visiting  $T$  states, the total number of samples is equal to  $NT$ .

Since the expectation value of a single qubit observable is bounded as  $\langle \sigma_z \rangle \in [-1, 1]$

and since the gradient of an action's expected value is given by equation (18), then  $\nabla_{\theta}\langle a \rangle_{\theta} \in [-1, 1]$ . Therefore, the following holds:

$$\beta \left( \nabla_{\theta}\langle a_{t_i} \rangle_{\theta} - \sum_{b_{t_i}} \pi(b_{t_i} | s_{t_i}, \theta) \nabla_{\theta}\langle b_{t_i} \rangle_{\theta} \right) \in [-2\beta, 2\beta] \quad (27)$$

By defining  $R_{max}$  as the maximum possible reward at any time step and by recalling Equation (4), then

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \leq R_{max} \sum_{t=0}^{T-1} \gamma^t = R_{max} \frac{\gamma^T - 1}{\gamma - 1} \quad (28)$$

where the expression for the sum of  $T$  terms of a geometric progression was used. Using this upper bound on  $G(\tau)$  enables the following result

$$\sum_{t=0}^{T-1} G_t(\tau) \leq R_{max} \sum_{t=0}^{T-1} \frac{\gamma^{T-t} - 1}{(\gamma - 1)} \leq R_{max} \frac{T}{(\gamma - 1)^2} \quad (29)$$

where the last inequality can be obtained by algebraic development and by resorting again to the sum of terms of a geometric progression. Combining results (27) and (29), gives

$$\sum_{t=0}^{T-1} G_t(\tau_i) \beta \left( \nabla_{\theta}\langle a_{t_i} \rangle_{\theta} - \sum_{b_{t_i}} \pi(b_{t_i} | s_{t_i}, \theta) \nabla_{\theta}\langle b_{t_i} \rangle_{\theta} \right) \in \left[ -\frac{2\beta R_{max} T}{(\gamma - 1)^2}, \frac{2\beta R_{max} T}{(\gamma - 1)^2} \right] \quad (30)$$

From Hoeffding's inequality [57], the probability of the average over  $N$  estimates of the policy gradient random variable being  $\epsilon_{\nabla}$ -inaccurate is given by

$$\mathbb{P} [ |\nabla_{\theta}^* J(\theta) - \nabla_{\theta} J(\theta)| \geq \epsilon_{\nabla} ] \leq 2 \exp \left( -\frac{2N\epsilon_{\nabla}^2(\gamma - 1)^4}{16\beta^2 R_{max}^2 T^2} \right) \quad (31)$$

From the union bound, for all  $k$  parameters, the probability is less than

$$\mathbb{P} \left[ \bigcup_k 2 \exp \left( -\frac{N\epsilon_{\nabla}^2(\gamma - 1)^4}{8\beta^2 R_{max}^2 T^2} \right) \right] \leq 2k \exp \left( -\frac{N\epsilon_{\nabla}^2(\gamma - 1)^4}{8\beta^2 R_{max}^2 T^2} \right) \quad (32)$$

Let  $\delta_{\nabla} = \mathbb{P} [ |\nabla_{\theta}^* J(\theta) - \nabla_{\theta} J(\theta)| \geq \epsilon_{\nabla} ]$ . Then

$$\begin{aligned} 1 - \delta_{\nabla} &= \mathbb{P} [ |\nabla_{\theta}^* J(\theta) - \nabla_{\theta} J(\theta)| \leq \epsilon_{\nabla} ] \\ 1 - \delta_{\nabla} &\geq 1 - 2k \exp \left( -\frac{N\epsilon_{\nabla}^2(\gamma - 1)^4}{8\beta^2 R_{max}^2 T^2} \right) \\ \delta_{\nabla} &\leq 2k \exp \left( -\frac{N\epsilon_{\nabla}^2(\gamma - 1)^4}{8\beta^2 R_{max}^2 T^2} \right) \end{aligned} \quad (33)$$

Thus, an upper bound on  $N$  can be obtained

$$N \leq \frac{8\beta^2 R_{max}^2 T^2}{\epsilon_{\nabla}^2(\gamma - 1)^4} \log \left( \frac{2k}{\delta_{\nabla}} \right) \quad (34)$$

Considering  $NT$  samples completes the proof.  $\square$

## A.2 Total number of quantum circuit evaluations

Lemma 4.2 establishes an upper bound on the number of quantum circuit evaluations (or shots) required to  $\epsilon_{\langle \cdot \rangle}$ -estimate the policy gradient  $\hat{\nabla}_{\theta} J(\theta)$  with probability  $1 - \delta_{\langle \cdot \rangle}$ . This result builds on Lemma 4.1 and the same approach is used to demonstrate it.

**Lemma 4.2** (Total number of quantum circuit evaluations). *Let  $\theta \in \mathbb{R}^k$ ,  $\mathcal{O}(NT)$  be the sample complexity given by Lemma 4.1, and  $|A|$  the number of available actions. With probability  $1 - \delta_{\langle \cdot \rangle}$  and approximation error  $\epsilon_{\langle \cdot \rangle}$ , the quantum policy gradient algorithm requires a number of shots given by*

$$\mathcal{O} \left( \frac{|A|NT}{\epsilon_{\langle \cdot \rangle}^2} \log \left( \frac{2k}{\delta_{\langle \cdot \rangle}} \right) \right) \quad (19)$$

*Proof* An action preference observable  $\langle a \rangle_{\theta}$  is given by a single-qubit observable  $\langle \sigma_z \rangle$ , as described in Section 4.3. The number of shots,  $n'$ , required to estimate the observable expectation with additive error  $\epsilon_{\langle \cdot \rangle}$  with probability  $1 - \delta_{\langle \cdot \rangle}$  is akin to the estimation of the probability of a Bernoulli distribution using Hoeffding inequality. Since  $\langle a \rangle_{\theta} \in [-1, 1]$ , then, by resorting to Hoeffding inequality and the union bound, we have

$$\mathbb{P} \left[ |\langle a \rangle_{\theta}^* - \langle a \rangle_{\theta}| \geq \epsilon_{\langle \cdot \rangle} \right] \leq 2k \exp \left( -\frac{n' \epsilon_{\langle \cdot \rangle}^2}{2} \right) \quad (35)$$

Following the same reasoning as described in the proof of Lemma 4.1,  $n'$  is given by

$$n' \leq \frac{2}{\epsilon_{\langle \cdot \rangle}^2} \log \left( \frac{2k}{\delta_{\langle \cdot \rangle}} \right) \quad (36)$$

Since the observable's gradient  $\nabla_{\theta} \langle a \rangle_{\theta}$  is estimated via parameter shift rules, as stated in Equation (18), it requires the estimation of each action preference observable twice, i.e. both  $\langle a \rangle_{\theta + \frac{\pi}{2}}$  and  $\langle a \rangle_{\theta - \frac{\pi}{2}}$ . Therefore, the number of shots,  $n$ , required to estimate  $\nabla_{\theta} \langle a \rangle_{\theta}$  is given by

$$n = 2n' \leq \frac{4}{\epsilon_{\langle \cdot \rangle}^2} \log \left( \frac{2k}{\delta_{\langle \cdot \rangle}} \right) \approx \mathcal{O} \left( \frac{1}{\epsilon_{\langle \cdot \rangle}^2} \log \left( \frac{2k}{\delta_{\langle \cdot \rangle}} \right) \right) \quad (37)$$

Recalling that  $\mathcal{O}(NT)$  samples are needed as in Lemma 4.1 and that each sample incurs  $|A|$  estimates, completes the proof.  $\square$