ELSEVIER

Uncovering instabilities in variational-quantum deep Q-networks

Maja Franz[a,1,*], Lucas Wolf[a,1], Maniraman Periyasamy[b,1],
Christian Utrecht[b], Daniel D. Scherer[b], Axel Plinge[b],
Christopher Mutschler[b], Wolfgang Mauerer[a,c]

[a] *Technical University of Applied Sciences Regensburg, Germany*
[b] *Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits IIS, Division Positioning and Networks, Nuremberg,*
Germany
[c] *Siemens AG, Corporate Research, Munich, Germany*

Received 30 November 2021; received in revised form 7 July 2022; accepted 17 August 2022
Available online xxx

Abstract

Deep Reinforcement Learning (RL) has considerably advanced over the past decade. At the same time, state-of-the-art RL algorithms require a large computational budget in terms of training time to converge. Recent work has started to approach this problem through the lens of quantum computing, which promises theoretical speed-ups for several traditionally hard tasks. In this work, we examine a class of hybrid quantum-classical RL algorithms that we collectively refer to as variational quantum deep Q-networks (VQ-DQN). We show that VQ-DQN approaches are subject to instabilities that cause the learned policy to diverge, study the extent to which this afflicts reproducibility of established results based on classical simulation, and perform systematic experiments to identify potential explanations for the observed instabilities. Additionally, and in contrast to most existing work on quantum reinforcement learning, we execute RL algorithms on an actual quantum processing unit (an IBM Quantum Device) and investigate differences in behaviour between simulated and physical quantum systems that suffer from implementation deficiencies. Our experiments show that, contrary to opposite claims in the literature, it cannot be conclusively decided if known quantum approaches, even if simulated without physical imperfections, can provide an advantage as compared to classical approaches. Finally, we provide a robust, universal and well-tested implementation of VQ-DQN as a reproducible testbed for future experiments.
© 2022 The Author(s). Published by Elsevier Ltd on behalf of The Franklin Institute.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

1. Introduction

Techniques for reinforcement learning (RL) have seen considerable progress during the past decade. Driven by both, algorithmic advances and the emergence of deep learning [1–3], RL has emerged from a conceptual approach to successfully tackling tasks previously deemed infeasible. This includes aspects of robotic manipulation [4–7], autonomous driving [8–10], and mastering combinatorially-hard board games [11–14]. At the same time, state-of-the-art deep RL methods require an exorbitant computational budget to match or exceed human performance on seemingly simple tasks, such as playing arcade video games. As an example, Badia et al. [15] report training times of roughly 53000 h, distributed over 256 machines, to achieve superhuman performance on all 57 Atari games of the Arcade Learning Environment benchmark [16]. Also, the learning dynamics of these approaches, both in terms of stability and optimality, are not yet fully understood and remain a subject of current research [17–20].

Concurrent to these developments, quantum computing [21] has started to receive increasing interest in real-life applications. It promises computational speedups, especially selected weakly-structured search problems like integer factoring [22], or exploration of unstructured search spaces [23,24] by exploiting fundamental phenomena of quantum mechanics (see Section 2.2). Reinforcement learning can be regarded as a search problem (in terms of seeking an optimal policy, as we outline in Section 2.1). Consequently, it is natural to ask whether a quantum speedup is realisable in this domain.

Limitations on achievable speedups have been studied in detail [25], and lower bounds are known for several important fundamental problems [26]. Despite numerous technological challenges rooted in, amongst others, noise and imperfections of near-term intermediate scale quantum devices [27], sufficient margins for industrially relevant improvements remain [28,29], but necessitate a more precise understanding and a critical evaluation of the performance of quantum approaches on currently available hardware designs. Since RL, like other machine learning approaches, relies on stochastic components that may amplify variations in algorithmic performance (and, more generally, challenge replication efforts), this is another aspect that requires careful consideration.

In this article, we examine and extend a class of recent hybrid quantum-classical approaches to reinforcement learning that we collectively refer to as *Variational-Quantum Deep Q-Networks* (VQ-DQN). Originally proposed by Chen et al. [30] and later refined by Lockwood and Si [31], VQ-DQN builds upon the deep Q-networks (DQN) algorithm [32,33], which replaces the core neural network component with a quantum machine learning model, namely, a variational quantum circuit (VQC) [34]. Although the results published in [32,33] promise interesting properties, we show that VQ-DQN approaches are subject to instabilities that ultimately cause the learned policy to diverge. Policy divergence is obviously detrimental to the practical utility of the approach, especially if it already happens in perfect simulations of quantum systems. Quantum computers that can be manufactured under the constraints of current technological limitations additionally suffer from noise, imperfections, and very strongly

---

∗ Corresponding author.

E-mail addresses: maja.franz@othr.de (M. Franz), lucas.wolf@st.othr.de (L. Wolf), maniraman.periyasamy@iis.fraunhofer.de (M. Periyasamy), christian.ufrecht@iis.fraunhofer.de (C. Ufrecht), daniel.scherer2@iis.fraunhofer.de (D.D. Scherer), axel.plinge@iis.fraunhofer.de (A. Plinge), christopher.mutschler@iis.fraunhofer.de (C. Mutschler), wolfgang.mauerer@oth-regensburg.de (W. Mauerer).

[1] These authors contributed equally (name order randomised).

limited amounts of available quantum bits. They are referred to as *noisy, intermediate scale quantum computers* (NISQ). To understand the additional degradation caused by these imperfections on the performance of RL approaches, we perform comparative experiments on actual quantum hardware—a gate-based IBMQ device (Falcon r4) operated in Ehningen, Germany.

In general, our investigation is part analysis and part reproduction study, and we provide a reproduction package with a well-tested implementation[2] of VQ-DQN. To make best use of available libraries and to provide an open testbed for future experiments, our implementation is written in two separate quantum frameworks, which are each coupled to a machine-learning framework: Tensorflow-Quantum [35](TFQ)/Tensorflow [36] and Qiskit [37]/Pytorch [38].

The paper is structured as follows: Section 2 provides a concise introduction to DQN (2.1), VQCs (2.2), and the VQ-DQN algorithm (2.3). Section 3 reviews related work. Section 4 describes our methodological approach towards finding and characterising instabilities. Section 5 summarizes our experiments. Section 6 explains the validation experiment on real quantum device. Further, we proceed to compare the DQN with variational quantum circuit against a DQN with classical neural network in Section 7. Finally, we conclude in Section 8.

2. Background

To introduce the concepts used in this study, the following paragraph discusses notation and basic principles of both, machine learning and quantum computation.

2.1. Deep Q-learning

Most formulations of RL center around the notion of a *Markov decision process* (MDP) [39], where an *agent* interacts with an *environment* at discrete time steps t. In each time step, the current configuration of the environment is summarised by the *state* $S_t \in \mathcal{S}$. Based on this information, the agent selects an *action* $A_t \in \mathcal{A}$ according to a *policy* $\pi(s, a) = \mathbb{P}[A_t = a | S_t = s]$. Executing the selected action causes a transition of the environment to a next state S_{t+1}; simultaneously, the agent receives a scalar *reward* R_{t+1} that quantifies the contribution of the selected action towards solving the task. The agent's goal is to maximize the return, i.e., the discounted sum of rewards, $G_t = \sum_{t'=t}^{T} \gamma^{t'} R_{t'}$ until a terminal state S_T is reached. In that, the discount factor γ controls how much the agent favors immediate over future rewards. Both S_{t+1} and R_{t+1} are assumed to obey the Markov property (i.e., conditional independence of previous states and actions given S_t, A_t). However, the MDP's *dynamics*, $\mathbb{P}[S_{t+1}, R_{t+1} | S_t, A_t]$, are typically unknown to the agent, which necessitates learning a policy by trial-and-error.

The fundamental idea of *Deep Q-Learning* (also referred to as *deep Q networks*, DQN) [32,33] is to learn the *optimal state-action value function* $Q_*(s, a) = \max_\pi \mathbb{E}[G_t | S_t = s, A_t = a, \pi]$ – that is, the return expected when taking action a in state s, and then following an optimal policy in all future states. Once $Q_*(s, a)$ is known, an optimal policy can be easily recovered by selecting actions greedily, that is $\pi_*(s) = \mathrm{argmax}_a Q_*(s, a)$. This is achieved by training a neural network to satisfy the well-known *Bellman Optimality Equation* (BOE) that relates the values of a state-action pair to the value of the next state:

$$Q_*(s, a) = \mathbb{E}\left[R_t + \gamma \max_{a'} Q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right] \tag{1}$$

---

[2] See https://doi.org/10.5281/zenodo.7030069.

More concretely, the deep Q-network is trained to minimize the difference between the left- and right-hand side of this equation (also known as the *temporal difference error* or *TD-error*), under some loss function (e.g., L_2), evaluated on mini-batches of transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ sampled by the agent. These transitions are sampled using an *off-policy* approach – instead of applying the current greedy policy (also called *target policy*), an ϵ-greedy *behavior policy* that selects a random action with probability ϵ is chosen. Decaying ϵ over the course of training allows the agent to explore the environment, while guaranteeing that the behavior policy and target policy (and hence, the underlying data distributions) converge eventually.

As Mnih et al. [40] point out, learning Q_* with a high-capacity function approximator leads to convergence problems. To this end, DQN makes use of (1) a *target network*, which is a copy of the deep Q-network with temporarily fixed weights to evaluate the right-hand side of Eq. (1), and (2) an *experience replay buffer* [41] from which experienced transitions are re-sampled for mini-batch gradient descent. For a detailed discussion of these specifics, we refer the interested reader to [32,33].

2.2. Variational quantum circuits

Quantum computation uses the *qubit* as the fundamental unit of information. In contrast to classical bits, a set of n qubits can not only assume the 2^n classical basis states (i.e., $0, 1, \ldots, 2^n - 1$), but also any superposition of these basis states. Note that superimposable quantum states reside in an infinite state space than their classical counterparts, which is often seen as an indication of increased computational capabilities, although the exact reason for possible quantum speeds remains elusive [42].

The variational quantum circuit is a machine learning model based on quantum circuits [34]. Similar to neural networks, VQCs consist of sequential *layers* that represent parameterised transformations on the VQC's quantum state. In particular, VQC layers apply e.g. learnt single-qubit rotations (in X-, Y-, and Z direction using the corresponding Pauli operators [21]) to each qubit of the circuit. Entanglement can be generated by applying a series of CNOT-gates [21] to pairs of qubits. The specific single-qubit rotation parameters are learned via gradient-descent on an error signal, computed over the expected measurements in Z direction of one or more output qubits.

2.3. VQ-DQN

Variational quantum deep Q-networks (VQ-DQN) [30,31,43] replace the deep neural network in DQN with a VQC.

2.3.1. Q-value extraction

For a given input MDP state, Q-values are predicted for all $|\mathcal{A}|$ actions simultaneously by taking the expectation value of a measurement (in Z direction) of a corresponding number of output qubits. The resulting measurements lie within $[-1; 1]$; obtaining valid action values thus requires further processing, for instance by scaling the measured results by a learnt multiplicative factor.

2.3.2. Input encoding

To input a (classical) MDP state $s \in \mathcal{S}$ to the VQC, that state needs to be represented as a quantum state $|\Psi(s)\rangle$ using the available qubits. Chen et al. [30] address this problem

by only considering MDPs with discrete state spaces and associating each MDP state with one of the 2^N quantum basis states. Lockwood and Si [31] and Skolik et al. [43] extend this method to MDP states with continuous components with a simple encoding scheme, with which the authors report results on the "Blackjack" and "CartPole-v0" environments (see Ref. [44] for implementation details). In particular, each component of the input state s is encoded by applying parameterised Pauli rotation gates [21] to one respective qubit in the circuit (initialised to $|0\rangle$). Lockwood and Si [31] propose two encoding schemes: **Scaled (S)** encoding, which determines a rotation angle by scaling finite-domain input components to $[0, 2\pi]$, and **Directional (D)** encoding, which encodes infinite-domain inputs by rotating the qubit by π if the input is greater than 0. Skolik et al. [43] additionally present **Continuous (C)** encoding, which computes rotation angles as the arctan of the respective input component.

3. Related work

3.1. Deep Q-Learning and its instabilities

The DQN approach dates back to Watkin's Q-Learning [45] and has seen a lot of interest over the years due to its immense potential in learning capabilities. Deep Q-Learning is itself an active field of research because of its versatility in end applications. Nevertheless, as versatile as the end applications are, the algorithm possesses space for improvements in its stability and speed of convergence to a solution [15,17,46–51]. In particular, the Q-learning approaches, i.e., off-policy learning with function approximation and bootstrapping, are known to diverge in certain scenarios. This divergence occurs more often when the Q-value is approximated using a non-linear function approximator such as a deep neural network. However, the root causes are still unknown [18,52–54].

3.2. Quantum reinforcement learning

Over the past few years, there have been several attempts to improve the performance of reinforcement learning algorithms via possible 'quantum advantage' using quantum computing. Like in the classical realm, no one method has emerged as the superior approach in performance or generality. The first quantum reinforcement learning (QRL) algorithm (to our knowledge) has been proposed by Dong et al. [55], which uses a modified version of Grover's algorithm [23] to learn a state-value function. As in the classical reinforcement learning family, whose members vary in algorithm and methodology, various algorithms for QRL have been studied [56–59]. The VQ-DQN algorithm was originally proposed by Chen et al. [30] where the authors have used variational quantum circuits to solve two different discrete environments, namely, 'cognitive radio' and 'frozen lake'. Both these environments are discrete environments where the state space is finite. The next study on VQ-DQN algorithm was conducted by Lockwood and Si [31], where the authors used a VQC to solve both continuous and discrete environments. Another study that analyses the learning performance and behavior of VQ-DQN was conducted by Skolik et al. [43]. Here the authors explore the effects of having a VQC as a Q-value approximator along with techniques like data re-uploading and a hybrid quantum-classical model.
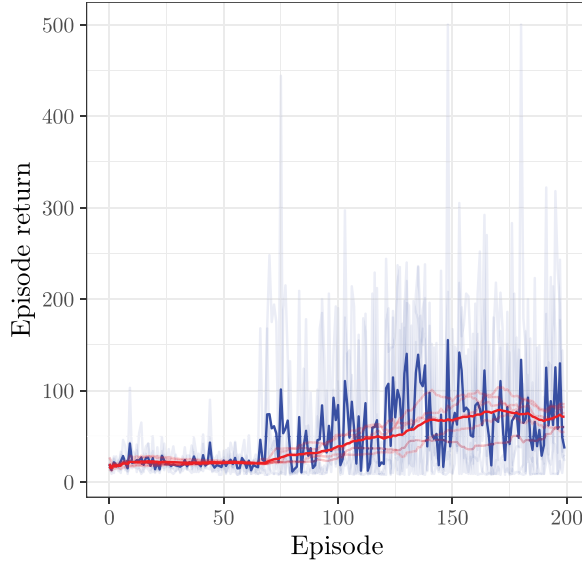
Fig. 1. Results from Ref. [31], reproduced using the published source code. The light blue lines indicate the total reward collected in an episode, using the greedy policy for each agent. The light red lines represent a moving average of the (up to) 20 previous episode returns. Results are averaged over five experiments, which is represented by the strong red and strong blue lines. The experiments are based on the `CartPole-v1` environment, where the maximum achievable return value is 500. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4. Reproduction study

To gauge the learning capability of VQ-DQN, we first reproduce the results published by Lockwood and Si [31] on the `CartPole-v1` task (cf. Section 2.3). We train five VQ-DQN agents and evaluate their performance during training using the source code[3] published by the authors. The results are visualised in Fig. 1. The blue line indicates episode returns. The red line represents a moving average of the (up to) 20 previous returns.[4] While our measurements reproduce the computational outcome of the published results, we identify two notable methodological aspects that require careful consideration and interpretation:

Training frequency—A step of mini-batch gradient descent is carried out only once per episode (namely, after its termination). This differs substantially not only from the original DQN algorithm, but also from the pseudo-code provided by Lockwood and Si [31], were training is executed in regular intervals after a set number of trajectories has been sampled by the agent. We are not aware of other approaches in the literature that pursue or analyse this approach, and conjecture that it might have a detrimental effect on learning, since the distribution of transitions in the replay buffer grows faster than the amount of data that the agent perceives. The adaptation also complicates the comparison between independent runs of the algorithm, depending on the length of the experienced episodes.

---

[3] Available on GitHub (link in PDF).

[4] Note that these statistics have been measured with the original source code, without modification. Superficial differences in visual appearance are caused by the plot aesthetic settings.
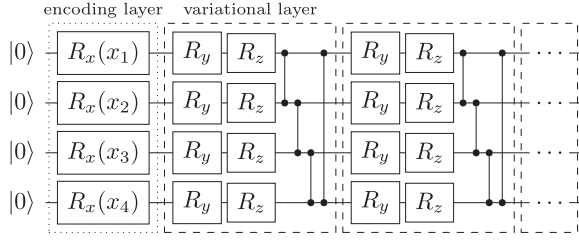
Fig. 2. Standard VQC architecture.

Performance evaluation—Measuring agent performance in terms of a moving average over previous runs is not a good indicator for learning success: Averaged returns have been generated by different policies, that is, trained on increasing numbers of transitions at different stages of ϵ-decay. Further, the averaging approach shadows any underlying instabilities as indicated by the raw episode returns: In all five runs, the blue line oscillates strongly between low and high return values, indicating that the underlying policy network/circuit fails to converge towards an optimal policy. Note that in complex environments, DQN convergence *can* be non-monotonic in terms of measured returns (see, e.g., Ref. [32]). Observing oscillations of this magnitude on `CartPole` (which can be learnt in an approximately monotonic fashion by a simple neural network with DQN, refer to 7) does not give a promising outlook on VQ-DQN's capability to generalise to more challenging tasks.

Besides, we would like to explicitly point out that the experiment is based on `CartPole-v1`, where return values of up to 500 can be achieved. In contrast, returns in `CartPole-v0` cannot exceed 200, which is important to take into account when judging closeness to optimality of particular approaches, especially when the visual display of episode return time series uses clipped axes.

One other study which overcame these instabilities using a VQ-DQN algorithm to solve the Cartpole environment is conducted by Skolik et al. [43]. Here the authors have used slightly different gate connectivity in their VQC compared to Lockwood and Si [31]. Apart from the change in VQC architecture, the authors also perform a gradient descent optimization step after every 30 sampling steps. They also present their total reward attained in each episode averaged over ten different agents rather than presenting a moving average.

Skolik et al. [43] have studied and tested various combinations of pure and quantum-classical hybrid VQC architectures in their work. However, the pure VQC model did exhibit the same instabilities exhibited by Lockwood and Si's model. Skolik et al. [43] used a hybrid VQC model where the inputs to and outputs from the VQC were multiplied with classical weights' along with the data re-uploading strategy [60] to overcome these instabilities. Data re-uploading is a strategy where the encoding circuit is reintroduced at multiple instances in a VQC. The standard encoding method follows a traditional neural network setup where the input to the network generally comes before the variational layers as shown in Fig. 2. However, in a gate-based VQC, both the input and the variational parameters are fed into the circuit as rotational angles. Therefore, there is no theoretical limitation on the maximum number of gates nor the number of repetitions of input features that can be fed into the circuit. Hence, the encoding circuit can be placed before every variational layer as shown in Fig. 3.

Reintroducing the encoding circuit increases the expressivity of the model [61]. It was shown by Schuld et al. [61] that the functions represented by VQCs are Fourier sums. In
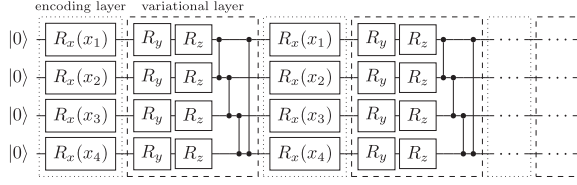
Fig. 3. VQC architecture with data re-uploading strategy.

which, the variational layers determine the amplitudes of the Fourier sum and the encoding layer fixes the frequency spectrum. Hence, the more encoding layers present via data re-uploading, the larger the frequency spectrum represented by the VQC the higher the expressivity of the represented function class can be. Even though the hybrid model exhibited a relatively stable learning behavior, the impact of classical weights on the overall training process is not distinguished nor studied. The results of our replication attempt of the work by Skolik et al.[5] are shown in Fig. 4. These experiments were conducted based on the parameters given in the Appendix section of Ref. [43]. The measurement results shown in Fig. 4 confirm the published results.

5. Experiments

Previous implementations of the VQ-DQN approach show various methodological issues [31] that we have discussed in detail in the previous section. For having a stable and uniform VQ-DQN framework that coincides with the classical RL practices and to provide a replication of existing results on top of mere reproduction, we re-implement the original deep Q-learning algorithm as described in [32,33] in Tensorflow [36] /Tensorflow-Quantum [35] (TFQ). In contrast to the previous implementations, which use TFQ too, our re-implementation allows to conveniently integrate extensions and has a higher degree of configurability of hyperparameters. Furthermore we included a flexible validation mechanism, which is used to evaluate the performance of a current policy. Since in previous implementations a fair comparison between different studies was difficult due to several meanings of return values (e.g. averaging over past episodes as in Ref. [31] vs. taking a single episodes return value as in Ref. [43]), we designed our validation mechanism to allow a uniform comparison of different classical and quantum RL approaches. (Section 5.1 discusses implementation details). This section covers experiments, which were conducted using the quantum simulators of the TFQ framework. In addition to our TFQ implementation, we also ported the code to the Qiskit [38] framework in order to run experiments on the IBM Quantum [62] devices, which is described in detail in Section 6.

Using our TFQ-implementation, we run a set of experiments to systematically evaluate the observed instabilities. Throughout all our experiments, we used the `CartPole-v0` environment to ensure comparability with [43] and [31], and also to keep computational cost at bay. Section 5.2 investigates the effects of the chosen input encoding and Q-value extraction method on performance and stability. Using these insights, we run an extensive cross-validation study

---

[5] The associated source code published by the authors of [43] is available on GitHub (link in PDF). Skolik et al. also provide a simplified implementation as a tutorial in the TFQ documentation (link in PDF). Note that we were not aware of these implementations during our reproduction process.
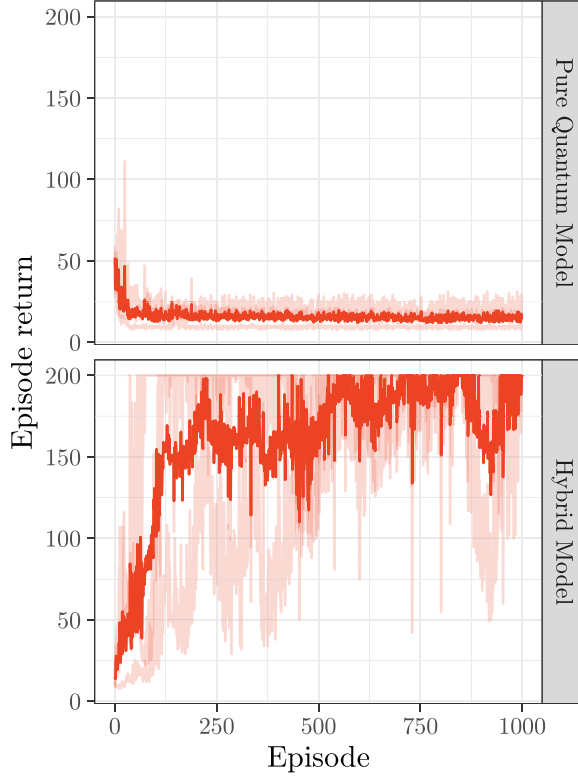
Fig. 4. Replication attempt of the results from Skolik et al. [43]. Here we replicated the hybrid quantum-classical model with data re-uploading and the pure quantum model with data re-uploading as proposed by Skolik et al. [43]. The experiments are based on the `CartPole-v0` environment, where the maximum achievable return value is 200.

described in Section 5.3. Additionally we have investigated properties of the VQC parameter space as a potential cause for instabilities; as the experiments conducted based on this speculation did not lead to a justifiable root cause, we focus only on the experiments on the input-encoding, Q-value extraction methods, and cross-validation mentioned above in this paper. However, we have included a brief discussion in Appendix A for reference.

5.1. Methodology

To describe our methodology, let us first set the employed conventions: By *sampling steps*, we refer to the transitions sampled from the ϵ-greedy behavior policy. By *training step*, we understand one iteration of gradient descent. Words in `monospaced` font indicate configurable parameters of the algorithms.

To ensure comparability between our different experimental setups, and especially between previous research and our dedicated experiments, we choose sampling steps as fundamental unit of training time. Each experiment is run for 50000 sampling steps. We deliberately use a long time horizon to capture any phenomena that may materialise late in the learning process caused by slow convergence, but retain the possibility to terminate successful

runs prematurely, as described in detail below. Initially, the replay memory is pre-filled with `train_after`=1000 sampling steps, corresponding to at least five full episodes, using a uniform random policy with $\epsilon = 1$.

A sampling step does not necessarily entail a training step; instead, a training step is carried out every `train_every` sampling steps. As backpropagation [63] on quantum devices is computationally intensive due to gradients being estimated via the parameter-shift rule [64,65], we introduced this parameter as a means to keep the number of training steps per episode feasible. We note, however, that in this paper, we only report validation results on quantum hardware, while the agent has been trained in simulation. Similarly, we update the target network parameters to equal the policy network parameters every `update_every` sampling steps. After the initial warm-up phase, we decay ϵ linearly over `epsilon_duration` sampling steps in total, starting at a value of `epsilon_start`=1, and ending at a value of `epsilon_end`=0.01. Keeping $\epsilon > 0$ ensures continued exploration with a near-greedy policy.

Since performance on the ϵ-greedy policy is not indicative of learnt performance when ϵ is large [66], we estimate the expected return achieved by the current greedy policy in regular intervals. Specifically, we measure return over a single episode on a copy of the training environment every `validate_every`=100 sampling steps (note that the parameter does not influence the actual training process, and is just used for performance monitoring). If the average validation return over the past consecutive 25 validation steps reaches 196 (recall that the maximum return is 200, and that we need to allow for some jitter), we regard the task as solved and terminate training early. While this differs from the official `CartPole-v0` benchmark (see https://gym.openai.com/envs/CartPole-v0/) that necessitates a return of at least 195 sustained over 100 episodes, we find that training is very unlikely to diverge past this point, given that ϵ has decayed sufficiently.[6]

5.2. Encoding and extraction methods

After experimentally verifying the correctness of our implementation, we replace the Q-network by a VQC using the circuit architectures proposed in Refs. [31,43]. The need for mapping input parameters onto quantum states has already been discussed in Section 2.3.2; we consider the following approaches:

1. **Continuous (C)**: continuous encoding applied to all input components.
2. **Scaled & Continuous (SC)**: scaled encoding applied to finite-domain input components, continuous encoding otherwise.
3. **Scaled & Directional (SD)**: scaled encoding applied to finite-domain input components, directional encoding otherwise.

Along with the encoding strategies, we also investigate the impact of different Q-value extraction methods on agent performance. This is necessary due to the mismatch between VQC outputs and Q-values. In particular, we distinguish between:

---

[6] We provide a set of results on the accompanying website that have enjoyed traversing the maximum number of episodes, and none of the results shows difference in convergence behaviour depending on the convergence criterion used. However, for experiments on the experimental IBM Quantum device, a reduced number of episodes is crucial to ensure practical feasibility of the calculations.
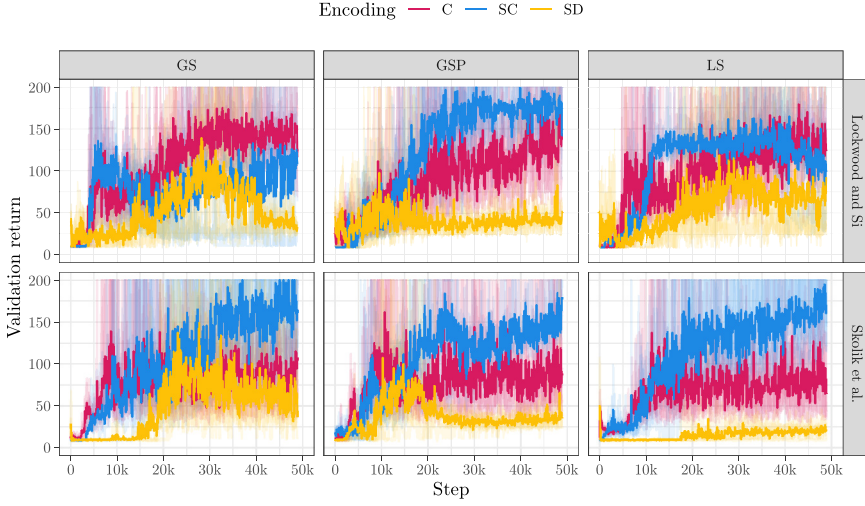
Fig. 5. Validation returns for using the VQC-layer structure as in Ref. [31] (top) and Ref. [43] (bottom) with different input encoding strategies. The columns correspond to the extraction strategy (ltr. Global Scaling (GS), Global Scaling with Quantum Pooling (GSP), Local Scaling (LS), Ref. Section 5.2). Results are averaged over five experiments each. These experiments are based on the `CartPole-v0` environment, where the maximum achievable return value is 200.

1. **Local Scaling**: each output is scaled by a dedicated trainable weight as described in Ref. [43].
2. **Global Scaling (GS)**: all outputs are scaled by a single trainable weight.
3. **Global Scaling with Quantum Pooling (GSP)**: quantum pooling as described in Ref. [31], followed by global scaling.

5.2.1. Initial experiment

We conducted experiments for each combination of input encoding, Q-value extraction method and circuit architecture, totalling in 18 runs. To this end, we adapted hyperparameters from Ref. [43] to our slightly modified algorithm described in Section 5.1 (without data re-uploading). VQC weights are initialised to zero to avoid *barren plateaus* [67], i.e. the vanishing gradient problem as suggested in Ref. [68] and classical weights are initialised to one.

Results are shown in Fig. 5. As is apparent, instabilities occur in every run and are not tied to a specific encoding-/extraction setting. Nevertheless, some models only achieve comparatively low returns on average: In particular, runs involving directional encoding tend to perform sub-par, which we attribute to the high information-loss incurred by the encoding scheme. Directional encoding is therefore not considered in further experiments.

To minimize the number of classical parameters, we focus on global scaling (with and without pooling) in further experiments. While local scaling has not performed worse or less stable, the additional classical parameters increase model capacity, and might therefore shadow deficiencies on the quantum parts.

As described by Mnih et al. [40], Q-Learning is known to be instable, when a nonlinear function approximator, such as a classical neural network or a VQ-DQN, is used to rep-

Table 1
Hyperparameter settings for cross-validation.

Hyperparameter	Description	Default value
	Fixed parameters throughout cross validation runs	
`num_steps`	#sampling steps	50000
`train_after`	#sampling steps before first training step	1000
`train_every`	#sampling steps between training steps	10
`update_every_start`	initial #sampling steps between target network updates	30
`update_every_end`	final #sampling steps between target network updates	500
`update_every_duration`	#sampling steps for `update_every` increase	35000
`replay_capacity`	max. #transitions in replay buffer	50000
`optimizer`	Loss-function optimizer	Adam [74]
`batch_size`	batch size for gradient descent	32
`loss`	TD error loss function	L_2
`epsilon_start`	initial value for ϵ decay	1.0
`epsilon_end`	final value for ϵ decay	0.01
`validate_every`	#sampling steps between validation runs	100
`eta_end`	final value for learning rate η	0.01 ∗ `eta_start`
	Hyperparameters subject to cross validation	
`eta_start`	initial value for learning rate η	{0.001, 0.01, 0.1}
`eta_duration`	#training steps for learning rate decay	{2000, 4000}
`epsilon_duration`	#sampling steps for ϵ decay	{10000, 20000, 30000}
`gamma`	discount factor γ	{0.99, 0.999}

resent the state-action value function. Our implementation already incorporates mechanisms suggested by Mnih et al. to support convergence in Q-Learning. However, these mechanisms do not guarantee a stable behaviour and we can not rule out that the instabilities in VQ-DQN are caused by classical algorithmic constraints. In Section 7 we compare the VQ-DQN to a classical neural network using the same algorithm. Our results support the hypothesis that the reason for instabilities could be classical. Therefore, in the next subsection, we study the effect of classical hyperparameters on the training process of VQ-DQN.

5.3. Cross-Validation

As instabilities persist throughout our experiments, we turn to hyperparameters as a source of instabilities. To this end, we re-utilize the above setting (C, SC/GS, GSP) with hyperparameters from Ref. [43] as a starting point. Following recommendations [69–72] from classical supervised learning, we add a linear decay to the learning rate η. In particular, we decrease η over a period of `eta_duration` training steps from `eta_start` towards a target value of `eta_end`=0.01*`eta_start`. Additionally, we progressively increase the `update_every` parameters as learning progresses. This choice is motivated by the observation that the delta between target and policy network decreases as the agent becomes more proficient on the task. Finally, to optimize resource utilization and minimize training time, we increase the batch size from 16 to 32, since this does not have a major impact on the agent's performance [73].

We cross-validate over the following hyper-parameter choices: `eta_start` (i.e., the initial learning rate) $\in \{10^{-3}, 10^{-2}, 10^{-1}\}$, `eta_duration` (learning rate decay duration) $\in \{2000, 4000\}$, `epsilon_duration` $\in \{10000, 20000, 30000\}$, `gamma` $\in \{0.99, 0.999\}$. The remaining parameters have been kept fixed over all experiments and are listed in Table 1.
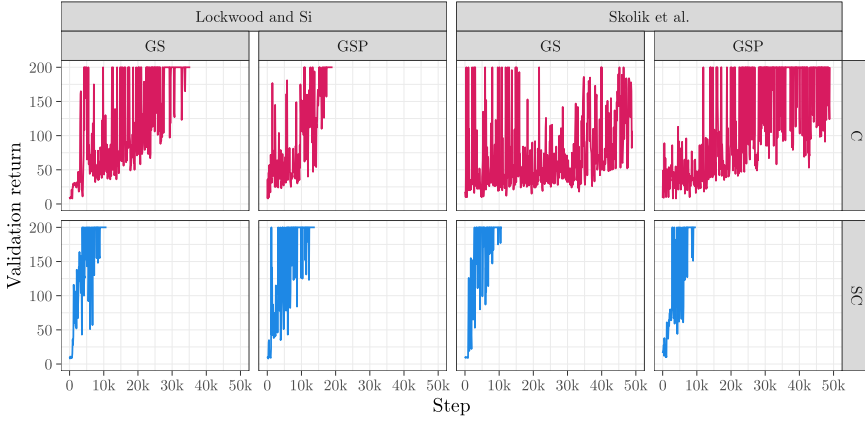
Fig. 6. Validation returns for our best-performing hyperparameter constellations in the baseline configurations. The figure considers the results for the VQC architecture described in Ref. [31] (left) and Ref. [43] (right). Columns correspond to the extraction method (ltr. Global Scaling (GS), Global Scaling with Quantum Pooling (GSP), Local Scaling (LS)), rows correspond to the input encoding strategy (Continuous (C), Scaled & Continuous (SC), see Section 5.2). The experiments are based on the `CartPole-v0` environment, where the maximum achievable return value is 200.

Table 2
Hyperparameters cross-validation results. The table provides values for `eta_start` (η_s), `eta_duration` (η_d), `epsilon_duration` (ϵ_s), and `gamma` (γ). Encodings C, SC, GS, and GSP as defined in Section 5.2.

Architecture	Baseline				Baseline + data re-uploading			
	η_s	η_d	ϵ_d	γ	η_s	η_d	ϵ_d	γ
[31]/C/GSP	0.01	2000	20000	0.99	0.01	2000	30000	0.99
[31]/C/GS	0.001	4000	20000	0.99	0.01	2000	30000	0.999
[31]/SC/GSP	0.01	2000	20000	0.99	0.1	2000	20000	0.999
[31]/SC/GS	0.01	4000	30000	0.99	0.01	2000	30000	0.99
[43]/C/GSP	-	-	-	-	0.01	2000	30000	0.999
[43]/C/GS	-	-	-	-	0.01	2000	10000	0.99
[43]/SC/GSP	0.01	2000	10000	0.999	0.01	2000	10000	0.99
[43]/SC/GS	0.01	4000	30000	0.99	0.01	2000	10000	0.99

The following subsections describe our results obtained on the baseline setting (without data re-uploading), and a modified variant with data re-uploading, respectively.

5.3.1. Baseline

Results for the baseline case are depicted in Fig. 6 and Table 2. We only present a selection of the best-performing hyperparameter constellations due to space constraints, but provide the full set of results on the accompanying website. As evident from the figure, almost every model was able to achieve stable optimal performance (according to our early-stopping criterion). Generally, the SC encoding tends to convergence faster as compared to models with continuous encoding; in the best case (Skolik et al./SC/GSP), optimal performance is reached after a mere 97 validation steps. This shows that VQ-DQN is in fact capable of learning a stable optimal policy, albeit hyperparameter tuning is a sensitive influence factor.
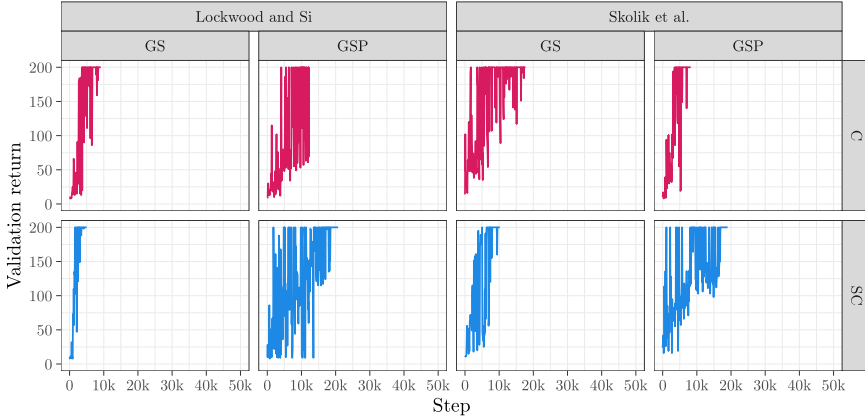
Fig. 7. Validation returns for our best-performing hyperparameter constellations in the baseline configurations with data re-uploading. The figure considers the results for the VQC architecture described in Ref. [31] (left) and Ref. [43] (right). Columns correspond to the extraction method (ltr. Global Scaling (GS), Global Scaling with Quantum Pooling (GSP), Local Scaling (LS)), rows correspond to the input encoding strategy (Continuous (C), Scaled & Continuous (SC), see Section 5.2). The experiments are based on the `CartPole-v0` environment, where the maximum achievable return value is 200.

5.3.2. Baseline with data re-uploading

From Fig. 6, it is evident that the performance of the VQ-DQN algorithm also suffers due to the choice of encoding strategy used along with the bad choice of hyperparameters. For example, the agent with the continuous encoding format does not learn an optimal policy in many cases. Here to increase the expressivity of the model, we can use techniques such as data re-uploading [43,60]. The results for the baseline case with data re-uploading are depicted in Fig. 7 and Table 2. As in Section 5.3.1, We only present a selection of the best-performing hyperparameter constellations due to space constraints. From the results shown in Fig. 7, we can conclude that the data re-uploading strategy does not significantly increase the VQ-DQN algorithm's performance. Though it increases the expressive power of the model, which in turn allows the agent to learn optimal behavior in some cases (for example, agent with Continuous (C) encoding), the performance change is negligible or even negative in most cases. Moreover, the data re-uploading strategy increases the gate count in the VQC architecture, and this increase in gate count is not ideal for the NISQ devices due to noise.

5.4. Discussion on instabilities in VQ-DQN

In our approach of VQ-DQNs, we train a VQC with a classical optimization loop. Such a setting is known to be prone to the *barren plateau* effect [67], which describes a problem of vanishing gradients that causes the inability to converge to an optimal return value. However, *barren plateaus* only occur in random VQCs. To counter randomness in the quantum circuits, we initialised all VQC parameters systematically to zeros. Since the VQCs in our experiments are neither very wide (4 Qubits), nor deep (5 "Layers"), randomness induced by gradient-based optimization is also limited. Therefore we rule out *barren plateaus* as the source of instabilities.

With the possibility of the barren plateau avoided, one can say that every agent with its unique architecture combinations and a reasonable encoding scheme is capable of learning the

optimal policy to solve the cart pole environment. This can be seen from the results shown in sections 5.3.1 and 5.3.2. The architecture combinations which did not learn an optimal policy during experiments conducted by different authors (Ref. [43] and [31]) showed a tendency to learn the optimal policy during our experiments. The reason why these agents show such a tendency is the selection of the right set of classical hyperparameters. The agents learned the optimal policy only for a few sets of classical hyperparameters during our hyperparameters search. This made us conclude that the VQC-DQN algorithms are highly sensitive to classical hyperparameters.

The results from Sections 5.3.1 and 5.3.2 elucidate that the data re-uploading strategy does not always outperform its corresponding architecture without data re-uploading in sampling efficiency. One possible reason for this could be that the optimal hyperparameter set required for these architectures might fall outside the search space used in the experiments. One other possible reason for this underperformance can be inferred from the work of Schuld et al. [61]. Schuld et al. show that the function represented by a VQC is a Fourier sum. In particular, the variational layers determine the amplitudes and the encoding layers determine the frequency spectrum. As shown in ref [75], when it comes to data re-uploading strategy, the variational layers between two encoding layers might not be expressive enough which reduces the overall expressivity of the VQC. The expressivity can be increased by increasing the number of variational layers. However, this leads to an architectural change which is out of scope for this study.

6. Validation on IBM quantum device

Results from Sections 5.3.1 and 5.3.2 illustrate that a VQC can learn a stable policy to solve the `CartPole-v0` environment using the DQN algorithm if the right set of hyperparameters are used. In order to gauge the detrimental influence of device noise on an agent trained using an ideal simulator in solving the environment, we tested the trained model in an actual IBM quantum device [62]. As a first step, we had to port the VQ-DQN algorithm from the Tensorflow/TFQ API [35,36] to the Pytorch/Qiskit API [37,38] as the IBM quantum devices use the Qiskit API [37] as their primary programming library. There is one significant difference between the Qiskit API [37] and the TFQ API [35] to be noted here. The TFQ [35] API calculates the expectation value analytically, whereas the Qiskit API [37] estimates the expectation value by simulating the `ideal quantum device` and measuring its outcomes. Likewise, the expectation values are estimated in the IBM quantum device [62] by measuring the outcome multiple times. Further, we trained the best-performing model without data re-uploading from Section 5.3.1 using Qiskit `qasm_simulator` [37] and verified the correctness of our implementation in comparison to the results from Section 5.3.1. We chose a model without data re-uploading due to the fact that the quantum devices available right now are prone to noise. Hence adding more gates via data re-uploading in NISQ devices seems counter-productive. Once the correctness was verified, we uploaded the weights trained using the `qasm_simulator` to the IBM Quantum (ibmq_ehningen) device and validated the learned policy. The results of these validation runs are shown in Fig. 8.

Though the agents trained in the ideal simulator learned an optimal policy to solve the `Cartpole-v0` environment, testing the trained agent in the ibmq_ehningen device did not reproduce the optimal behavior. This degradation in behavior is due to the noise present in the IBM Quantum device. An agent trained in the IBM Quantum device from scratch might reduce the effect of noise and learn a policy close to the optimal policy. Additionally, different
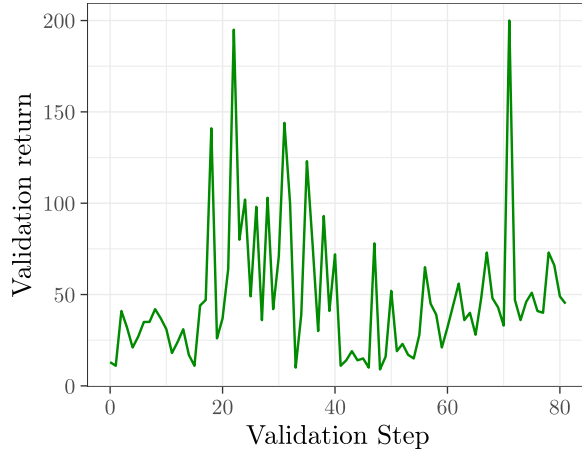
Fig. 8. Results of our validation run on `ibmq_ehningen` [62]. The experiments are based on the `CartPole-v0` environment, where the maximum achievable return value is 200.

types of error mitigation techniques can be employed to reduce the effects of noise at the cost of additional overhead. However, when we attempted to train the agent from scratch on the IBM quantum device, the training turned out to be infeasible due to the following practical issues:

1. We observed waiting times in the queue to start a job execution (referred to as fair-share queue for jobs in IBM Quantum systems) in the cloud-based IBM Quantum device that were typically two orders of magnitude (or more) larger than the actual job execution time. As (roughly speaking) a single action selection corresponds to a single job in the fair share queue, even completion of a single episode takes a substantial amount of time.
2. The overall time it takes to achieve low-variance estimators of expectation values can become quite large due to the large number of shots (i.e., measurement samples) taken for a single circuit instance.

Here, the first hindrance can be overcome in time as the availability of quantum devices and resources is expected to increase in the near future. As improvements in hardware and orchestration of quantum and classical computational resources progress, we might also be witness to an increased number of circuit layer operations per second (CLOPS) [76]. When we started the training process in the `ibmq_ehningen` device, the job execution time for each action selection took between 15 to 30 s, and each training step took around 3 min (as the training step performs gradient decent via parameter-shift rule). These long execution and waiting times make the training process in real quantum devices impractical for training algorithms like VQ-DQN, where the agent has to interact with the environment sequentially.

7. Comparison to classical neural network

A popular "quantum advantage" claimed by a good fraction of the literature in QRL is that the VQC has better state-action pair representation, samples efficiently, and learns an
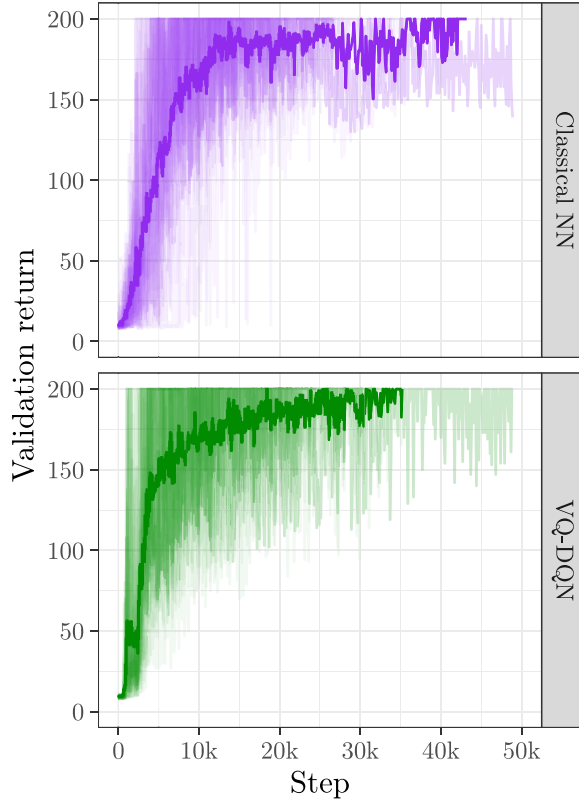
Fig. 9. Comparison between VQ-DQN and classical NN averaged over 30 different agents. The experiments are based on the `CartPole-v0` environment, where the maximum achievable return value is 200.

optimal policy faster than the classical neural network [30,31,43]. Hence to compare the sample efficiency of a VQ-DQN-agent trained on an ideal simulator against a classical neural network, we trained a simple fully-connected network with one hidden layer to solve the `Cartpole-v0` environment. To ensure a fair comparison, we restricted the total number of parameters of the network to 58, and did cross-validation on the same set of hyperparameters as explained in Section 5.3.

The results shown in Fig. 9 indicate that initially, the VQC seems to learn faster than the neural network. For a more rigorous discussion we resort to Ref. [77], where sample efficiency of an algorithm is defined for an online learning setting as the number of time steps from which on an agent trained by the algorithm perceives an average reward exceeding a certain threshold V_{thresh} with high probability.

For a weaker statement adapted to a numerical treatment, we propose to use significance testing under the null hypothesis of mean reward being smaller than V_{thresh}. Thus, we define sample efficiency as the number of time steps from which on the null hypothesis is rejected with respect to the given threshold. As statistical test we propose to use a one-sample t-test [78,79], in particular its one-sided version as we compare the performance of a particular algorithm against a given threshold. Thus, we perform sufficiently many independent runs of each algorithm and fix the significance level at $\alpha = 0.05$.

With respect to this metric the variational quantum circuit indeed crosses $V_{\text{thresh}} = 120$ faster than the classical network; however, for larger threshold values, no definite statement can be made.

8. Conclusion

We have systematically studied the performance of quantum-assisted reinforcement learning schemes on both simulators and physical quantum computers. We find—not quite unexpected—that at the current, early state of technological development, quantum computers do not bring any measurable advantage in this scenario. We even find that simulated quantum systems do not bring clear advantages over classical approaches.

Nonetheless, a number of constructive insights can be drawn from our experiments. Following previous work, we have trained models on classical simulators and only performed the execution step on quantum hardware. This approach, albeit practically necessitated by current-day hardware, creates a mis-match in terms of handling noise: For future work, we recommend including noise in the training process, especially since Ref. [80] suggests for small-scale systems that existing noise models lead to a good match between simulation and hardware, and therefore provide a more faithful basis for comparing between algorithmic performance on simulated and physical hardware.

Most importantly, our results do *not* corroborate observations made when reinforcement learning on quantum computers was first introduced into the literature in Ref. [30]: While the authors in this approach upload weights determined by classical training onto a quantum machine as we do in this paper, they find that executing the model does not vary much between simulation and NISQ machine. We, on the contrary, observe a total mismatch in performance. We expect the most probable explanation for this discrepancy to lie in (a) the size of the machine (five versus 27 qbits) and the problem of choice (cognitive-radio versus cartpole; a random policy as would be caused by growing amounts of noise from NISQ devices is obviously better suited to the former than the latter).

We encounter additional hindrances towards the practical application of quantum computers: Waiting time on queues in a shared, cloud-like environment is a major practical issue, which will however be alleviated with the broader availability of quantum chips. Nonetheless, the temporal contributions of sequential elements of algorithms to the overall computation time would also occur in a non-shared setting and do substantially increase wall-time run-times, which is an obvious impediment to practical utility.

As long as noise and imperfections are unavoidable, we find that adapting algorithms and approaches to account for these issues is a major design challenge for quantum algorithms. One possible approach would be to equip simulated QPU designs with appropriate, yet tunable and physically realistic noise behaviour. By seeking optimal models and parameters under these unavoidable constraints, an "ideal" noise model can be identified, and future QPUs be built such that design trade-off decisions are taken so that the resulting hardware closely mimics the identified noise and imperfection behaviour. In other words, we hypothesise that in the space of hardware design decisions, and assuming that hardware imperfections impact different computations in a different way, this opens a degree of freedom that can be leveraged to design custom algorithmic-specific hardware.

Funding

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Appendix A. Reparameterization

As part of our initial investigation into potential sources of divergent behavior, we analysed model weight distributions over time. Here, we noticed that in some runs, the magnitudes of the learned VQC weights increase indefinitely over time. This behavior is intriguing, considering that the respective parameters control qubit rotations (in X-, Y- or Z-basis) in radians, and therefore naturally "wrap" at 2π (in the sense that a parameter θ and $\theta + 2\pi$ specify the same circuit). The periodicity in parameter space translates to a periodicity in the loss function. This can intuitively be thought of as "copies" of the loss landscape at 2π increments along the respective dimensions. In consequence, circuit weights growing beyond 2π imply that an adjacent copy of the optimization landscape is visited and a potential minimizing value along this dimension has been overshot.

To rule out that this phenomenon impedes learning, we modify the quantum circuit by squashing the (unbounded) VQC parameters θ to the range $[0; 2\pi]$ using the transformation $f(\theta) = 2\pi \cdot \sigma(\theta)$ (where σ is the well-known sigmoid function). This effectively ensures that any given set of parameter values uniquely specifies the qubit transformation enacted by the circuit, while being optimizable by backpropagation.

Unfortunately, all of our experiments involving reparameterizations did not yield better performance or further insights. We take this as evidence that periodicities are, after all, not the root cause of instabilities in VQ-DQN.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org

[2] K.P. Murphy, Machine Learning - A Probabilistic Perspective, Adaptive Computation and Machine Learning Series, MIT Press, 2012.

[3] Y. LeCun, Y. Bengio, G.E. Hinton, Deep learning, Nat. 521 (7553) (2015) 436–444, doi:10.1038/nature14539.

[4] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016) 39:1–39:40. http://jmlr.org/papers/v17/15-522.html

[5] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, J. Peters, Stable reinforcement learning with autoencoders for tactile and visual data, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9–14, 2016, IEEE, 2016, pp. 3928–3934, doi:10.1109/IROS.2016.7759578.

[6] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J.W. Pachocki, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, W. Zaremba, Learning dexterous in-hand manipulation, CoRR(2018). arXiv preprint arXiv:1808.00177

[7] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, S. Levine, QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation, CoRR (2018) arXiv preprint arXiv:1806.10293.

[8] S. Bhalla, S.G. Subramanian, M. Crowley, Deep multi agent reinforcement learning for autonomous driving, in: C. Goutte, X. Zhu (Eds.), Advances in Artificial Intelligence - 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings, Lecture Notes in Computer Science, vol. 12109, Springer, 2020, pp. 67–78, doi:10.1007/978-3-030-47358-7_7.

[9] A. Baheri, S. Nageshrao, H.E. Tseng, I.V. Kolmanovsky, A. Girard, D.P. Filev, Deep reinforcement learning with enhanced safety for autonomous highway driving, in: IEEE Intelligent Vehicles Symposium, IV 2020, Las Vegas, NV, USA, October 19, - November 13, 2020, IEEE, 2020, pp. 1550–1555, doi:10.1109/IV47402.2020.9304744.

[10] Z. Huang, J. Wu, C. Lv, Efficient deep reinforcement learning with imitative expert priors for autonomous driving, CoRR (2021) arXiv preprint arXiv:2103.10690.

[11] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489, doi:10.1038/nature16961.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, Nature 550 (2017a) 354, doi:10.1038/nature24270.

[13] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T.P. Lillicrap, K. Simonyan, D. Hassabis, Mastering Chess and Shogi by self-play with a general reinforcement learning algorithm, CoRR (2017b) arXiv preprint arXiv:1712.01815.

[14] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T.P. Lillicrap, D. Silver, Mastering Atari, Go, Chess and Shogi by planning with a learned model, CoRR (2019) arXiv preprint arXiv:1911.08265.

[15] A.P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, C. Blundell, Agent57: outperforming the Atari human benchmark, CoRR (2020) arXiv preprint arXiv:2003.13350.

[16] M.G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: an evaluation platform for general agents, CoRR (2012) arXiv preprint arXiv:1207.4708.

[17] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, CoRR (2015) arXiv preprint arXiv:1509.06461.

[18] H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, J. Modayil, Deep reinforcement learning and the deadly triad, CoRR (2018) arXiv preprint arXiv:1812.02648.

[19] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, A. Madry, Are deep policy gradient algorithms truly policy gradient algorithms? CoRR (2018) arXiv preprint arXiv:1811.02553.

[20] A. Agarwal, S.M. Kakade, J.D. Lee, G. Mahajan, Optimality and approximation with policy gradient methods in Markov decision processes, CoRR (2019) arXiv preprint arXiv:1908.00261.

[21] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information (10th Anniversary Edition), Cambridge University Press, 2016. https://www.cambridge.org/de/academic/subjects/physics/quantum-physics-quantum-information-and-quantum-computation/quantum-computation-and-quantum-information-10th-anniversary-edition?format=HB

[22] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Rev. 41 (2) (1999) 303–332, doi:10.1137/S0036144598347011.

[23] L.K. Grover, A fast quantum mechanical algorithm for database search, in: G.L. Miller (Ed.), Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22–24, 1996, ACM, 1996, pp. 212–219, doi:10.1145/237814.237866.

[24] L.K. Grover, A framework for fast quantum mechanical algorithms, in: J.S. Vitter (Ed.), Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23–26, 1998, ACM, 1998, pp. 53–62, doi:10.1145/276698.276712.

[25] D. Stilck França, R. García-Patrón, Limitations of optimization algorithms on noisy quantum devices, Nat. Phys. 17 (11) (2021) 1221–1227, doi:10.1038/s41567-021-01356-3.

[26] H. Buhrman, B. Loff, S. Patro, F. Speelman, Limits of quantum speed-ups for computational geometry and other problems: fine-grained complexity via quantum walks, CoRR (2021) arXiv preprint arXiv:2106.02005.

[27] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum 2 (2018) 79, doi:10.22331/q-2018-08-06-79.

[28] A. Bayerstadler, G. Becquin, J. Binder, T. Botter, H. Ehm, T. Ehmer, M. Erdmann, N. Gaus, P. Harbach, M. Hess, J. Klepsch, M. Leib, S. Luber, A. Luckow, M. Mansky, W. Mauerer, F. Neukart, C. Niedermeier, L. Palackal, R. Pfeiffer, C. Polenz, J. Sepulveda, T. Sievers, B. Standen, M. Streif, T. Strohm, C. Utschig-Utschig, D. Volz, H. Weiss, F. Winter, Q. Technology, A.C. QUTAC, Industry quantum computing applications, EPJ Quantum Technol. 8 (1) (2021) 25, doi:10.1140/epjqt/s40507-021-00114-x.

[29] F. Bova, A. Goldfarb, R.G. Melko, Commercial applications of quantum computing, EPJ Quantum Technol. 8 (1) (2021) 2, doi:10.1140/epjqt/s40507-021-00091-1.

[30] S.Y.C. Chen, C.H.H. Yang, J. Qi, P.Y. Chen, X. Ma, H.S. Goan, Variational quantum circuits for deep reinforcement learning, IEEE Access 8 (2020) 141007–141024, doi:10.1109/ACCESS.2020.3010470.

[31] O. Lockwood, M. Si, Reinforcement learning with quantum variational circuits, CoRR (2020) arXiv preprint arXiv:2008.07524.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M.A. Riedmiller, Playing Atari with deep reinforcement learning, CoRR (2013) arXiv preprint arXiv:1312.5602.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[34] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning, Phys. Rev. A 98 (2018) 032309, doi:10.1103/PhysRevA.98.032309.

[35] M. Broughton, G. Verdon, T. McCourt, A.J. Martinez, J.H. Yoo, S.V. Isakov, P. Massey, R. Halavati, M.Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D.V. Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J.R. McClean, R. Babbush, S. Boixo, D. Bacon, A.K. Ho, H. Neven, M. Mohseni, TensorFlow quantum: a software framework for quantum machine learning, 2021, arXiv preprint arXiv:2003.02989

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org, https://www.tensorflow.org/.

[37] H. Abraham, Qiskit: an open-source framework for quantum computing, 2019, 10.5281/zenodo.2562110

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: an imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[39] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: NIPS'99: Proceedings of the 12th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, 1999, pp. 1057–1063.

[40] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, CoRR (2016) arXiv preprint arXiv:1602.01783.

[41] L.J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Mach. Learn. 8 (1992) 293–321, doi:10.1007/BF00992699.

[42] F. Arute, K. Arya, R. Babbush, D. Bacon, J.C. Bardin, R. Barends, R. Biswas, S. Boixo, F.G. Brandao, D.A. Buell, et al., Quantum supremacy using a programmable superconducting processor, Nature 574 (7779) (2019) 505–510, doi:10.1038/s41586-019-1666-5.

[43] A. Skolik, S. Jerbi, V. Dunjko, Quantum agents in the gym: a variational quantum algorithm for deep q-learning, arXiv preprint arXiv:2103.15084(2021).

[44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym, CoRR (2016) arXiv preprint arXiv:1606.01540.

[45] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3) (1992) 279–292, doi:10.1007/BF00992698.

[46] Z. Wang, N. de Freitas, M. Lanctot, Dueling network architectures for deep reinforcement learning, CoRR (2015) arXiv preprint arXiv:1511.06581.

[47] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, in: Y. Bengio, Y. LeCun (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings, 2016.

[48] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, D. Silver, Distributed prioritized experience replay, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30, - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018. https://openreview.net/forum?id=H1Dy---0Z

[49] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A.D. Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, D. Silver, Massively parallel methods for deep reinforcement learning, CoRR (2015) arXiv preprint arXiv:1507.04296.

[50] A.P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, C. Blundell, Never give up: Learning directed exploration strategies, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020, OpenReview.net, 2020. https://openreview.net/forum?id=Sye57xStvB

[51] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, W. Dabney, Recurrent experience replay in distributed reinforcement learning, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019, OpenReview.net, 2019. https://openreview.net/forum?id=r1lyTjAqYX

[52] J.N. Tsitsiklis, B. Van Roy, An analysis of temporal-difference learning with function approximation, IEEE Trans. Automat. Control 42 (5) (1997) 674–690, doi:10.1109/9.580874.

[53] H. van Hasselt, Double q-learning, in: J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, A. Culotta (Eds.), Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6–9 December 2010, Vancouver, British Columbia, Canada, Curran Associates, Inc., 2010, pp. 2613–2621. https://proceedings.neurips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html

[54] R.S. Sutton, A.R. Mahmood, M. White, An emphatic approach to the problem of off-policy temporal-difference learning, CoRR (2015) arXiv preprint arXiv:1503.04269.

[55] D. Dong, C. Chen, H. Li, T.J. Tarn, Quantum reinforcement learning, IEEE Trans. Syst. Man Cybern. Part B 38 (5) (2008) 1207–1220, doi:10.1109/TSMCB.2008.925743.

[56] V. Dunjko, J.M. Taylor, H.J. Briegel, Framework for learning agents in quantum environments, CoRR (2015) arXiv preprint arXiv:1507.08482.

[57] F. Flamini, A. Hamann, S. Jerbi, L.M. Trenkwalder, H.P. Nautrup, H.J. Briegel, Photonic architecture for reinforcement learning, CoRR (2019) arXiv preprint arXiv:1907.07503.

[58] F. Neukart, D.V. Dollen, C. Seidel, G. Compostella, Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces, CoRR (2017) arXiv preprint arXiv:1708.09354.

[59] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M.A. Riedmiller, Deterministic policy gradient algorithms, in: ICML, in: JMLR Workshop and Conference Proceedings, vol. 32, JMLR.org, 2014, pp. 387–395. http://dblp.uni-trier.de/db/conf/icml/icml2014.html-SilverLHDWR14

[60] A. Pérez-Salinas, A. Cervera Lierta, E. Gil-Fuster, J. Latorre, Data re-uploading for a universal quantum classifier, Quantum 4 (2020) 226, doi:10.22331/q-2020-02-06-226.

[61] M. Schuld, R. Sweke, J.J. Meyer, Effect of data encoding on the expressive power of variational quantum-machine-learning models, Phys. Rev. A 103 (3) (2021), doi:10.1103/physreva.103.032430.

[62] Ibm quantum, https://quantum-computing.ibm.com/2021.

[63] D.E. Rumelhart, J.L. McClelland, Learning internal representations by error propagation, 1987, pp. 318–362.

[64] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning, Phys. Rev. A 98 (3) (2018), doi:10.1103/physreva.98.032309.

[65] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A 99 (3) (2019) 032331.

[66] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning, arXiv preprint arXiv:1611.02167(2016).

[67] J.R. McClean, S. Boixo, V.N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes 9 (1) (2018) 4812, doi:10.1038/s41467-018-07090-4. https://www.nature.com/articles/s41467-018-07090-4

[68] A. Skolik, J.R. McClean, M. Mohseni, P. van der Smagt, M. Leib, Layerwise learning for quantum neural networks, Quantum Mach. Intell. 3 (1) (2021) 1–11, doi:10.1007/s42484-020-00036-4.

[69] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition(2016) 770–778.

[70] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, 2020,

[71] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, arXiv preprint arXiv:2005.14165(2020).

[72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need (2017) 5998–6008.

[73] Y. Bengio, Practical Recommendations for Gradient-Based Training of Deep Architectures, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 437–478.

[74] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980(2014).

[75] M. Periyasamy, N. Meyer, C. Ufrecht, D.D. Scherer, A. Plinge, C. Mutschler, Incremental data-uploading for full-quantum classification, 2022, 10.48550/ARXIV.2205.03057

[76] A. Wack, H. Paik, A. Javadi-Abhari, P. Jurcevic, I. Faro, J.M. Gambetta, B.R. Johnson, Quality, speed, and scale: three key attributes to measure the performance of near-term quantum computers, 2021, arXiv preprint arXiv: 2110.14108

[77] S.M. Kakade, On the sample complexity of reinforcement learning, Gatsby Computational Neuroscience Unit, University College London, 2003 Ph.D. thesis.

[78] Student, The probable error of a mean The probable error of a mean, Biometrika 6(1908) 1–25.

[79] P. Henderson, R. Islama, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters (2017). arXiv preprint arXiv:1709.06560

[80] N. Meyer, Variational quantum circuits for policy approximation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Nuremberg, Germany, 2021 Master's thesis.