IOP Publishing Quantum Sci. Technol. 9 (2024) 025012 https://doi.org/10.1088/2058-9565/ad261b

Quantum Science and Technology

OPEN ACCESS

RECEIVED
18 January 2023

REVISED
31 August 2023

ACCEPTED FOR PUBLICATION
5 February 2024

PUBLISHED
21 February 2024

PAPER

Hybrid actor-critic algorithm for quantum reinforcement learning at CERN beam lines

Michael Schenk[1,*], **Elías F Combarro**[2], **Michele Grossi**[1], **Verena Kain**[1], **Kevin Shing Bruce Li**[1], **Mircea-Marian Popa**[3] **and Sofia Vallecorsa**[1]

[1] European Organisation for Nuclear Research, Espl. des Particules 1, 1211 Meyrin, Switzerland
[2] Computer Science Department, University of Oviedo, C. San Francisco 3, 33003 Oviedo, Asturias, Spain
[3] Politehnica University of Bucharest, Splaiul Independentei 313, 060042 Bucharest, Romania
[*] Author to whom any correspondence should be addressed.

E-mail: michael.schenk@cern.ch

Keywords: energy-based reinforcement learning, quantum machine learning, accelerator physics

Abstract

Free energy-based reinforcement learning (FERL) with clamped quantum Boltzmann machines (QBM) was shown to significantly improve the learning efficiency compared to classical Q-learning with the restriction, however, to discrete state-action space environments. In this paper, the FERL approach is extended to multi-dimensional continuous state-action space environments to open the doors for a broader range of real-world applications. First, free energy-based Q-learning is studied for discrete action spaces, but continuous state spaces and the impact of experience replay on sample efficiency is assessed. In a second step, a hybrid actor-critic (A-C) scheme for continuous state-action spaces is developed based on the deep deterministic policy gradient algorithm combining a classical actor network with a QBM-based critic. The results obtained with quantum annealing (QA), both simulated and with D-Wave QA hardware, are discussed, and the performance is compared to classical reinforcement learning methods. The environments used throughout represent existing particle accelerator beam lines at the European Organisation for Nuclear Research. Among others, the hybrid A-C agent is evaluated on the actual electron beam line of the Advanced Wakefield Experiment (AWAKE).

1. Introduction

The European Organisation for Nuclear Research (CERN) maintains a dense and diverse physics programme with numerous experiments requiring a broad spectrum of particle beam types to be produced and provided by the accelerator complex [1–4]. Combined with the requests for higher beam intensities and smaller beam sizes at an overall improved beam quality, this makes accelerator operation more and more challenging. The way forward is to exploit automation and improved modelling to boost machine flexibility, availability, and beam reproducibility. Ideally, as-built reversible physics models are available in the control rooms to adjust beam parameters. Whereas for many beam control problems, physics models are indeed used at the CERN accelerators, various systems are still tuned manually due to the lack of models or beam instrumentation. Recently, sample-efficient control algorithms, such as reinforcement learning (RL), have been introduced for some of these cases. Sample efficiency is essential for any optimisation algorithm in the context of accelerator operation to minimise the impact on beam time available for the physics experiments.

Q-learning is a popular RL algorithm [5], where the RL agent iteratively learns a so-called Q-function to define the optimal policy. While in classical deep Q-learning the Q-function is estimated using a deep neural network [6], the free energy-based reinforcement learning (FERL) approach utilises the free energy of a coupled spin system as a Q-function approximator. The spin system is typically represented by a quantum Boltzmann machine (QBM), and its free energy is determined, for example, through quantum annealing (QA) [7]. An improvement in the learning efficiency of an FERL agent compared to classical Q-learning

algorithms has already been demonstrated in earlier research; however, with the restriction to discrete state-action space environments [8, 9].

The goals of this study are twofold. First, to remove the limitation to discrete state-action space environments and to extend FERL to multi-dimensional continuous state-action spaces, opening doors for a broader range of real-world applications. This is particularly important for particle accelerator systems where the control parameters and observables are usually defined by continuous variables. The second objective is to compare the performance of the classical RL algorithms to their quantum or hybrid counterparts in terms of both sample efficiency and the number of parameters required to model the Q-function. The objectives are achieved in a two-stage approach. FERL is first extended to continuous state-space but still discrete action-space environments, and the impact of experience replay on the sample efficiency is studied. This is done for a one-dimensional beam steering environment of an existing beam line at CERN In a second stage, a hybrid actor-critic (A-C) scheme based on the classical Deep Deterministic Policy Gradient (DDPG) RL algorithm [10] is developed by combining a classical actor network with a QBM-based critic to allow for continuous state-action space environments. The hybrid scheme is validated and compared to its classical counterpart by applying it to a ten-dimensional environment of the electron beam line at the CERN AWAKE facility both in simulations and in the real world [2].

The rest of the paper is organised as follows. First, the main concepts of the RL and FERL domains are introduced in section 2, including an overview of existing related research. The contributions are discussed in section 3, comprising the development of the new hybrid A-C algorithm and experimental results for trajectory control at the two CERN beam lines with different complexities. This includes comparisons to the performance of the corresponding classical RL algorithms both using simulated quantum annealing (SQA) and D-Wave QA hardware, and an evaluation of a trained hybrid A-C agent on the real-world AWAKE electron beam line. Finally, some conclusions are raised, and some ideas for further study are proposed.

2. Background

2.1. RL

RL algorithms solve sequential decision-making problems by deciding which action $a \in A$ to take given a specific state $s \in S$ [5]. The problem can formally be described by using a discrete-time markov decision process consisting of state space S, action space A, state transition probabilities $P_a(s, s')$, and immediate transition rewards $R_a(s, s')$ emitted when moving from state s to s' under action a. The decision-making strategy is formally described through a policy function $\pi : S \times A \rightarrow [0, 1]$, which is a probability distribution that can be used to map each state to a chosen action. The optimal policy is learned through interaction with the environment to maximise the cumulative reward along the path of the visited states.

The algorithms discussed in this paper belong to the class of model-free RL algorithms, where the dynamics model of the environment under consideration is not explicitly learned or available. Within model-free algorithms, the class of policy optimisation methods learns the policy directly through policy gradient and is suitable for continuous action spaces. On the other hand, Q-learning methods seek to learn the so-called action-value or Q-function. The Q-function is a measure of the expected sum of discounted future rewards, assuming the agent in state s takes action a and then continues until the end of the episode following policy π. It is defined as

$$Q(s, a|\theta) = \mathbb{E}_\pi \left[\sum_{k=1}^{N} \gamma^{k-1} r_{t+k} | s_t = s, a_t = a \right], \tag{1}$$

where N is the number of states visited, starting from state $s = s_t$ at time step t and stopping at a terminal state at the end of the episode at time step $t + N$, $\gamma \in [0, 1]$ is a discount factor, and r_i is the reward the agent receives during iteration i. The Q-function is parameterised by θ, which denotes, for example, the network weights in case the Q-function is approximated by an artificial neural network.

The Q-function can be learned iteratively by using the training samples collected during agent-environment interactions and by employing the temporal-difference rule [5]

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right], \tag{2}$$

at time step t, where α is the learning rate. Once trained, that is, during exploitation, a Q-learning agent always takes the action that maximises the Q-function ('greedy policy').

One of the advantages of Q-learning in comparison to policy gradient methods is its sample efficiency due to experience replay [11]. All the state transitions, the chosen actions, and obtained rewards are stored in a replay buffer. During agent training, the Q-function updates are calculated based on mini-batches sampled from this buffer.

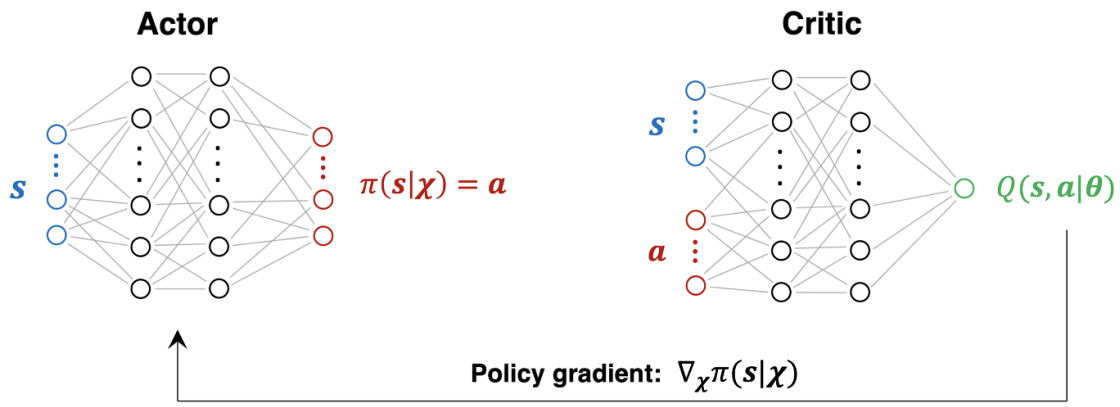

Figure 1. RL actor-critic scheme: the policy- or actor-network proposes action a given a state s, while the critic network estimates the Q-value of the proposed state-action pair. The actor network is trained through the policy gradient rule.

The Q-learning method is specifically suitable for discrete action spaces. A-C algorithms combine Q-learning with policy optimisation and hence offer a relatively sample-efficient method for continuous action spaces [5].

2.2. Deep Q-learning

The Q-function is typically approximated by a (deep) neural network for most real-world applications where the state-action space is large. Deep Q-learning or DQN is one of the most popular Q-learning algorithms [6]. DQN expects the state vector as input and provides the Q-values for all possible discrete actions at the output layer. Sorting allows determining the action that maximises the Q-function efficiently.

2.3. A-C algorithms

One major limitation of Q-learning is its restriction to discrete action-space environments. This is because the action for a given state is obtained with $a' = \arg\max_a Q(s, a)$ at inference and the update rule for agent training also needs to find the maximum over $Q(s, a')$ for all possible actions a' (see equation (2)).

The A-C scheme removes this limitation. The most basic algorithm is DDPG [10] with its architecture illustrated in figure 1. It consists of a 'critic' network parameterised by weights θ to approximate the Q-function and an 'actor' network parameterised by weights χ to learn the policy. According to this scheme, the actor proposes a (continuous) action given the current state, and the critic provides feedback on how good that state-action pair is by calculating its Q-value. DDPG interleaves Q-learning and policy updates. The update rule for the critic network is identical to equation (2) except that the term $\max_{a'} Q(s_{t+1}, a')$ is now replaced by $Q(s_{t+1}, \pi(s_{t+1}))$. Following the chain rule, the policy gradient updates can be calculated as

$$\nabla_\chi \pi = \mathbb{E}_\pi \left[\nabla_\chi Q(s, \pi(s|\chi)|\theta)\right] = \mathbb{E}_\pi \left[\nabla_a Q(s, a|\theta) \cdot \nabla_\chi \pi(s|\chi)\right]. \tag{3}$$

2.4. Energy-based RL

Other than DQN, FERL employs an energy-based model (EBM) [12, 13] to learn and approximate the Q-value function [7]. EBMs are based on statistical systems and each of the system's possible configurations has a specific associated energy state. Similar to physical systems, lower-energy states are more likely to occur compared to higher-energy states. The goal of training an EBM for FERL is hence to iteratively adjust its parameters such that the ground-state energy represents the Q-value. A particularly well-suited example of a statistical system for FERL is the Boltzmann machine (BM), an EBM represented by a probabilistic network of binary nodes [14]. BMs were shown to be universal function approximators of probability distributions over the states of their inputs [15–17]. This provides the foundation for using BMs as Q-function approximators. By basing the Q-function approximator on a physics-inspired model, such as the transverse-field Ising model [18], and the associated quantum BM to represent that system, we can define a Hamiltonian that is directly mappable to an actual physical system, such as a quantum annealer. This in turn allows implementing FERL on quantum hardware and investigating the potential benefits of the approach.

2.4.1. Clamped QBM

A BM consists of visible and hidden (latent) variables. They are denoted by v_i, with $i \in V$, and by h_j, with $j \in H$, respectively. Visible variables typically serve as the inputs and outputs, and hidden variables are added to increase the expressivity of the model.

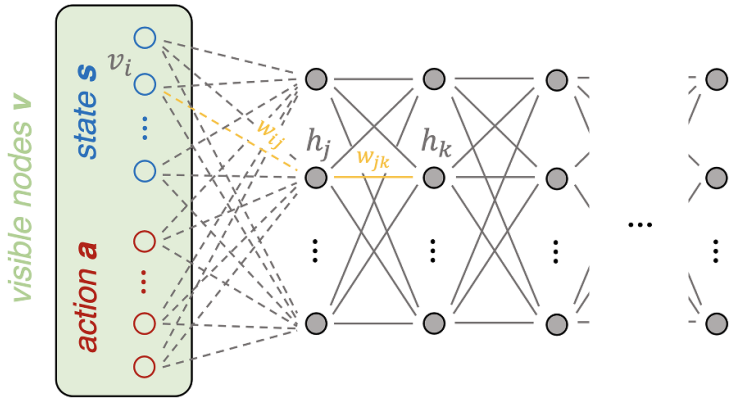

Figure 2. Clamped BM with topology typically used for FERL applications. Visible nodes are given by the state-action pair (green box). Hidden nodes are shown in grey. Visible-hidden and hidden-hidden couplings are illustrated by dashed and solid lines, respectively.

For the FERL algorithms discussed in this paper, the BM's topology is chosen to be clamped, building upon the research in [8]. This means that the visible variables are not part of the network. Instead, they enter the system's energy function as biases or self-couplings to the specific hidden variables assigned to them. These bias terms are used in a weighted sum, where the weights are given by the coupling strengths between visible and hidden variables. Bias terms are linear in the state of the hidden variable with which they are associated. On the other hand, couplings between hidden variables, such as $w_{jk} \in \mathbb{R}$, $j, k \in H$, correspond to quadratic contributors to the energy function.

For RL, the visible variables are composed of the vectors of the state-action pair $\mathbf{v} = (s, a) \in \mathbb{R}^{d_S + d_A}$, where d_S and d_A are the dimensionalities of the state and action space, respectively. Figure 2 illustrates a clamped BM typically employed for FERL.

A QBM can be represented by a physical system of coupled qubits in the presence of a purely transverse magnetic field, here pointing along the x-axis. Each node of the BM can assume a spin 'up' or 'down' state following a certain probability distribution. The system's energy states are described by the Hamiltonian of the transverse-field Ising model [18]

$$\mathcal{H}(\mathbf{v}) = -\sum_{\substack{i \in V, \\ j \in H}} w_{ij} v_i \sigma^z_{h_j} - \sum_{j,k \in H} w_{jk} \sigma^z_{h_j} \sigma^z_{h_k} - \Gamma \sum_{j \in H} \sigma^x_{h_j}, \tag{4}$$

where $\sigma^x_{h_j}$ and $\sigma^z_{h_j}$ are the Pauli spin matrices acting on the hidden node h_j for the x- and z-directions, respectively, and Γ denotes the transverse magnetic field strength. The transverse field introduces quantum fluctuations which enable, among others, quantum tunnelling [19].

When measuring the spin states along the z-coordinate, one loses information about the spin x-components. This can be overcome by replica stacking, a method developed in [8] to expand the transverse-field Ising model based on the Suzuki–Trotter expansion. By applying this technique, the non-zero transverse-field Ising model can be represented by a classical Ising model of one dimension higher with an effective Hamiltonian given by

$$\begin{aligned} \mathcal{H}^{\text{eff}}(\mathbf{v}) = -\frac{1}{N_r} \sum_{l=1}^{N_r} \left(\sum_{j,k \in H} w_{jk} h_{j,l} h_{k,l} + \sum_{\substack{i \in V, \\ j \in H}} w_{ij} v_i h_{j,l} \right) \\ - w^+ \left(\sum_{j \in H} \sum_{l=1}^{N_r} h_{j,l} h_{j,l+1} + \sum_{j \in H} h_{j,1} h_{j,N_r} \right), \end{aligned} \tag{5}$$

with $w^+ = \frac{1}{2\beta} \log[\coth \frac{\Gamma \beta}{N_r}]$, where N_r is the number of replicas, β the inverse temperature, and $h_{j,l}$ denotes the hidden node with index j in replica l.

For FERL, the negative free energy $F(\mathbf{v})$ of the clamped QBM is used to approximate the Q-function

$$Q(\mathbf{v}) \approx -F(\mathbf{v}) = -\langle \mathcal{H}^{\text{eff}}(\mathbf{v}) \rangle - \frac{1}{\beta} \sum_c \mathbb{P}(c|\mathbf{v}) \log \mathbb{P}(c|\mathbf{v}), \tag{6}$$

where c ranges over the spin configurations and $\mathbb{P}(c|\mathbf{v})$ denotes the probability of observing spin configuration c given visible nodes $\mathbf{v}$.

The temporal difference update rules for the QBM weights follow from the Bellman equations

$$w_{ij} \leftarrow w_{ij} + \alpha \Delta Q v_i \langle h_j \rangle, \tag{7}$$

$$w_{jk} \leftarrow w_{jk} + \alpha \Delta Q \langle h_j h_k \rangle,$$

with $\Delta Q = [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$, and $i \in V, j, k \in H$, and where $\langle h_j \rangle$ and $\langle h_j h_k \rangle$ are the expectation values of the hidden nodes and the products of hidden nodes, respectively. Based on equation (7), the weights of the QBM can be learned iteratively in analogy to classical Q-learning.

2.4.2. QA

Quantum adiabatic computing [20] is a model of quantum computation in which the adiabatic theorem [21] is exploited to obtain the ground state of a given Hamiltonian H_f. Initially, the ground state of a simple Hamiltonian H_i is prepared. Then, the system is evolved according to the time-dependent Hamiltonian

$$H(t) = A(t) H_i + B(t) H_f,$$

where $A(t), B(t) : [0, T] \to \mathbb{R}$ are such that $A(0) = B(T) = 1$ and $A(T) = B(0) = 0$. Provided that the evolution is slow enough, the adiabatic theorem ensures that, starting from the ground state of $H(0) = H_i$, the system will end up in the ground state of $H(T) = H_f$.

In practice, however, the time T can be extremely large (and, in some cases, very expensive or even impossible to compute [22]). For this reason, it is common to apply a heuristic implementation of quantum adiabatic computing that goes by the name of QA [23, 24]. In QA, H_f is taken from a restricted family of Hamiltonians, usually, those of the transverse-field Ising model, and T is fixed to some constant time, even if it does not guarantee adiabaticity.

QA is implemented by the Canadian company D-Wave in a series of quantum devices, some of which can be accessed online through a cloud-based service [25]. One of their most popular uses is the approximation of solutions to combinatorial optimisation problems. This is achieved by mapping the objective function to a Hamiltonian whose ground states are minimum-cost solutions to the original problem (see [23] for details). In this work, following [8], QA will be used to estimate the free energy of QBMs which serve as Q-function approximators, as explained in the previous subsection.

2.5. Related work

Quantum machine learning [26, 27] is a discipline aiming to establish a productive interplay between the parallel revolutions of quantum computing and artificial intelligence. Among various machine learning tasks and methods widely described in the literature, the quantum computing connection to the field of RL is still in a preliminary state, whereas A-C methods are among the state-of-the-art in current classical RL literature. With no ambition for completeness, but with a cross-sectional view of the literature, in this section, previous works in the field that made steps towards connecting quantum computing and RL are summarised.

Quantum enhancements for RL can be found mainly in sample efficiency and the acceleration of the algorithmic part of the learning agent. However, the importance of probing speed-up has only been studied for special models [9, 28] and by building on QA systems like those developed by D-Wave [25].

Additional consideration should be made concerning the technology adopted: different approaches are linked to different quantum formulations of the assigned problem, either QA or gate-based quantum computation. Some proposed algorithms can show a proved speed-up only in the fault-tolerant quantum regime. Current near-term quantum devices cannot handle complex quantum routines, so alternatives like the combination of deep BMs and quantum BMs or parameterised quantum circuits (PQC) for variational quantum algorithms (VQA) are being investigated for heuristic quantum speed-ups. Considering the commonly accepted division of RL algorithms into policy-based and value-based methods, a short summary of PQC implementation is provided in the following.

A first step in assessing PQCs in policy-based RL algorithms has been made in [29], where the authors focus on the role of data-encoding and readout strategies. Regarding the value-based approach, there are fundamental questions about the applicability of VQAs, particularly for Q-learning algorithms, where the role of the deep Q-network is to serve as a function approximator of the Q-function. Again, in [29] for the Q-learning approach with policy gradient-based RL, the authors identify families of environments that are proven to be hard for any classical learner but can be solved in polynomial time by a quantum learner in a policy learning setting.

In [30], a two discrete-state environment solution with PQC is studied, where a layer of additive weights follows the quantum model's output. Instead, in [31], the authors propose a continuous and discrete state spaces implementation where angle encoding is used for the continuous part with one initial layer of rotation gates. An interesting approach is also studied in [32] where the authors use VQA with amplitude encoding combined with a quantum circuit evolution algorithm. Evaluations on the Cart Pole benchmark as well as on a high-dimensional MiniGrid environment demonstrate that this is indeed a promising approach allowing mapping high-dimensional problems to noisy intermediate-scale quantum devices.

Another important reference that provides an extended discussion about the state of the art for quantum RL is [33]. From the theoretical point of view, the authors explore the possibility of approximating functions and the complexity of sampling from different models. This led to quantum generalisations of classical EBMs. They are quantum enhancements for a class of deep RL which can have further advantages over conventional methods by their capacity to capture more complex action-state distributions. Beyond general considerations, also for this case, the authors consider mainly one (simple) classical RL architecture, that is, deep Q-learning, which achieves better learning performances than conventional methods when the state and action spaces are large.

In terms of annealing, and concerning enhancements of the algorithmic part of the learning agent, speed-ups have been explored by building on the system of D-Wave computers. In [8] the authors implemented FERL on the D-Wave 2000Q device, where they proved that replica stacking is a successful method for estimating effective classical configurations obtained from a quantum annealer, with significant improvement over FERL using classical BMs. In [34], policy evaluation for discrete state spaces is embedded in the D-Wave QPU (Quantum Processing Unit). The authors also look into dealing with continuous state space in the context of self-driving vehicles. However, despite the interesting proposal, the work does not guarantee speed-up.

To the best of our knowledge, regardless of the computing paradigm considered, no previous proposal of a hybrid, that is, a quantum–classical, A-C scheme is present in the literature.

3. Contribution

3.1. Objectives and study cases

The main contribution of this paper is to develop and study a hybrid A-C scheme, combining a classical policy network with a quantum-based Q-network represented by a QBM. Given the improved sample efficiency observed empirically for FERL Q-learning on discrete state-action tasks [8, 9], the aim is to study whether an improvement in the training efficiency can also be observed for a hybrid A-C algorithm. The work was initially motivated by the existing control problems at the CERN accelerator complex, which would greatly benefit from sample-efficient RL algorithms and whose control variables and observations are usually defined over continuous spaces.

The environments under study are two existing CERN beam lines representing control problems of different degrees of complexity. The first describes a target steering task for a proton beam with a single control variable. In contrast, the second one represents the AWAKE electron beam line featuring ten control parameters. For both tasks, accurate simulations exist, which will be used mostly to train and study the different algorithms. The environments are built on top of the OpenAI gym [35] template to facilitate the use of existing classical RL algorithm implementations, such as Stable-Baselines3 [36].

In the following, two studies are presented. Study A considers an FERL Q-learning agent to solve a one-dimensional target steering problem. The impact of experience replay on the sample efficiency is assessed and compared to classical deep Q-learning. The action space is discretised, but the state space is continuous. In Study B, the performance of the hybrid A-C is evaluated for the ten-dimensional electron beam line and is compared to the classical DDPG algorithm using a continuous state-action space.

All the studies are first performed using the SQA library Sqaod [37]. Eventually, however, the FERL Q-learning and hybrid A-C agents are also adapted to and trained on D-Wave Advantage QA hardware [25]. Furthermore, for the AWAKE beam line environment, evaluations of the RL agents can also be performed in the real world, as equipment safety is less of a concern given the type and parameters of the particle beam. Hence, the agents trained in the simulated environment are also deployed on the real AWAKE beam line (sim-to-real transfer).

3.2. Hyperparameter selection

To guarantee a fair comparison between the classical RL and quantum FERL approaches, extensive hyperparameter searches were performed to ensure that the agents always train with the best sample efficiency possible. The search was realised by means of the Ray Tune [38] and Optuna [39] libraries. 500 hyperparameter sets were explored for every case. For every parameter set, the RL agents were trained 15

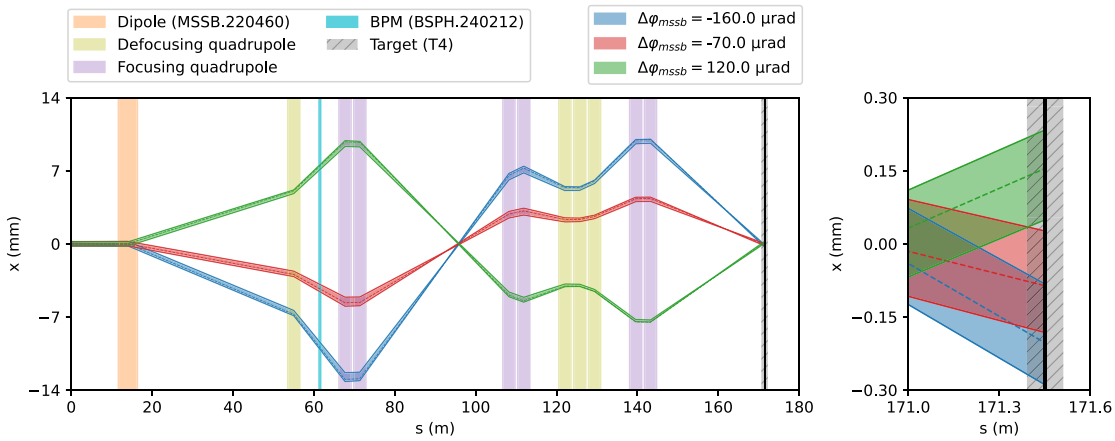

Figure 3. Beam target steering task at the CERN TT24-T4 beam line with one-dimensional state and action spaces. Left: horizontal beam trajectories obtained from tracking simulations are shown for three different settings of the main deflecting dipole (orange). Right: zoomed view on the target (grey, hatched) region showing the horizontal position of impact of the beam for the three settings of the main dipole.

times from scratch, and their performance was evaluated based on 500 randomly initialised episodes. The following hyperparameters were included in the search:

- **Exploration parameters:** initial ε-greedy and the fraction of training period with ε-greedy policy, maximum allowed iterations per episode;
- **Learning parameters:** initial learning rate, batch size, reward discount factor, target-net soft update factor;
- **Annealing parameters:** inverse annealing temperature, final transverse field.

The annealing time was fixed at 100 steps (SQA), and the number of Trotter slices was set to 5. The same procedure was employed for all the studies discussed in the following.

3.3. Study A: FERL Q-learning with continuous state space

This section discusses the performance of the classical and FERL agents on a proton beam target steering environment with one-dimensional state and action spaces. The FERL method described in [8], where the authors work with discrete binary state-action space environments, is extended to continuous state space. This is possible in a clamped QBM where the visible state input nodes only enter as bias terms.

The sample efficiencies of FERL and classical Q-learning agents and the impact of experience replay on the latter are assessed. Furthermore, the sample efficiency is also measured depending on the complexity of the Q-network. Initially, this task was also formulated as a discrete, binary state-action space environment, reproducing the method employed in [8]. For completeness, those results are included in appendix.

3.3.1. Environment and RL task

The beam target steering environment is based on the beam optics of the TT24-T4 transfer line at CERN [40]. This line is about 170 m long and transports protons with a momentum of 400 GeV c^{-1} from the Super Proton Synchrotron (SPS) to some of the fixed-target physics experiments installed in the CERN North Area. TT24 is equipped with several dipole and quadrupole magnets to steer and focus the beam, various beam position monitors (BPM), and the actual target, which is placed at the end of the line. The objective of the task is to optimise the number of particles hitting the target by tuning the first dipole magnet in the line to maximise the event rates in the particle detectors.

The left-hand side of figure 3 shows the relevant elements of TT24 together with horizontal beam trajectories obtained from tracking simulations for three different settings of the main bending dipole (orange). Depending on the dipole deflection angle, the particles hit the target (grey, hatched) at different horizontal positions, as illustrated by the zoomed view on the right-hand side of the figure. There are focusing (purple) and defocusing (olive) quadrupoles along the beam line to keep the beam particles confined.

The RL task is formalised as follows. The state s is defined by the beam position reading of one of the BPMs installed in the beam line (cyan). The action is given by the relative deflection angle induced by the dipole magnet (orange). Two discrete actions are possible—either increasing or decreasing the deflection angle by a small, fixed amount of 15 μrad. The allowed range of deflection angles is $[-140, 140]$ μrad. The

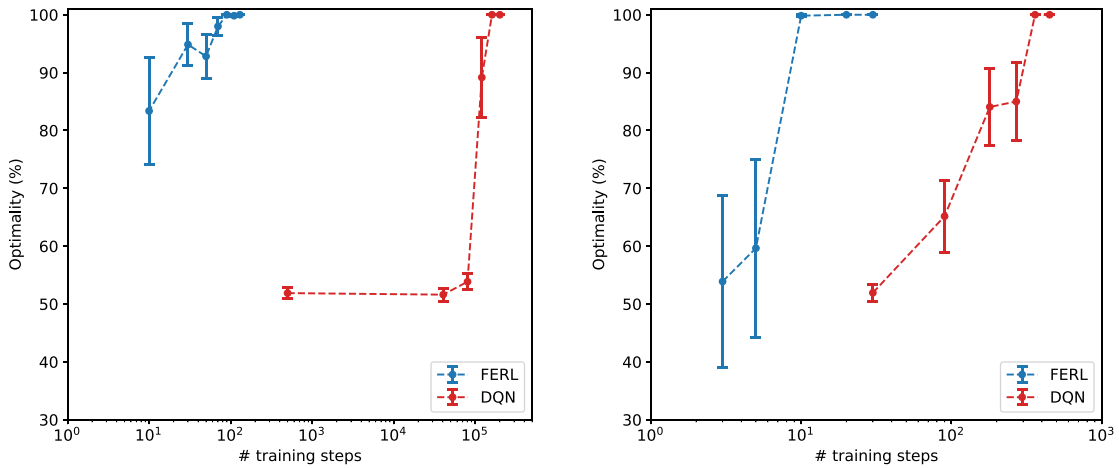

Figure 4. One-dimensional beam target steering environment with a continuous state and discrete action space: convergence study for agent optimality vs the number of training steps *without* (left) and *with* (right) experience replay for a classical DQN (red) and an FERL (blue) agent.

reward is calculated as the overlap integral between the target and the beam in the interval $x \in [-3\sigma_{\text{beam}}, 3\sigma_{\text{beam}}]$, assuming a Gaussian particle distribution with transverse root-mean-squared (rms) size σ_{beam}.

3.3.2. Results: classical deep Q-learning vs FERL

Figure 4 shows convergence studies of policy optimality vs number of training steps for the classical DQN (red) and the FERL (blue) agents after performing extensive hyperparameter optimisation. The optimality metric is defined as the fraction of the state space from where the trained agent will take optimal actions to reach the reward objective with the smallest number of steps possible. The classical agent uses a DQN architecture with two hidden layers with 128 nodes each. The number of hidden layers and the number of nodes per layer have been treated as hyperparameters, in addition to the parameters listed in section 3.2. The same number of nodes was assumed in every layer. The FERL agent, on the other hand, uses a 1×2 unit-cell Chimera graph QBM with a total of 16 qubits following the topology of the D-Wave 2000Q quantum annealer [25].

Two main conclusions can be drawn from this result. First, FERL significantly improves sample efficiency compared to the classical approach. With experience replay enabled, the number of training steps required to reliably achieve policy optimality of 100% is 10 steps for FERL and 380 for DQN. Second, for the given RL task, the impact of experience replay is significant not only for classical deep Q-learning (a well-known result [11]) but also for the FERL algorithm. Improvements in sample efficiency of a factor of 400 for DQN and 10 for FERL were observed when enabling experience replay.

In addition to improved sample efficiency, the QBM employed for FERL also exhibits a higher descriptive power than the classical DQN. This is illustrated by the plot in figure 5. It shows the required number of training steps to achieve 100% policy optimality vs the number of parameters in the Q-network on a log-log scale, with experience replay enabled. For the classical case, results for DQNs with one (red), two (green), and three (orange) layers are shown. Each data point has been obtained from a convergence scan with several training steps after individual hyperparameter optimisation. The labels next to the data points refer to the number of nodes per hidden layer. The performance of the FERL agent using a 1×2 unit-cell Chimera graph described above is shown in blue. It trains an optimal policy in only 10 steps using a QBM with 52 weights, while the most sample-efficient classical agent trains in about 380 steps requiring a DQN with two hidden layers and a total of about 10^4 weights. A classical agent with just one hidden layer and 64 nodes (258 DQN weights total) can also be trained successfully. However, it requires more than 500 training steps to become optimal. Adding more than two hidden layers does not seem to improve the sample efficiency further for the RL task under consideration.

3.3.3. Results: SQA vs D-Wave advantage QA

Figure 6 shows the state-action value estimates $\hat{Q}(s, a)$ (top), the derived greedy policy (middle), and histograms of the states visited during the training phase (bottom) for FERL agents trained with SQA (left) and on the D-Wave QA (right), respectively. The grey area marks the region where the beam intensity on target is beyond the set threshold, that is, the reward objective is reached.

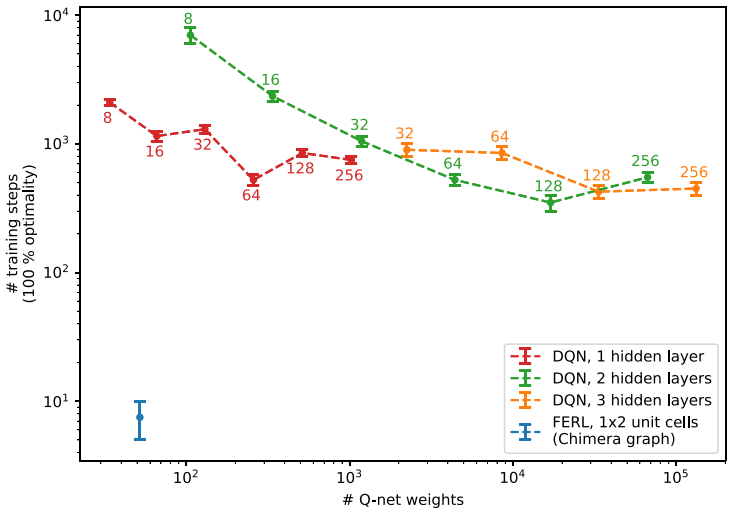

Figure 5. Required number of training steps to train a policy with 100% optimality vs number of DQN weights for different architectures (red, green, orange) and the FERL (blue) agent, respectively. The labels next to each data point denote the number of nodes per hidden layer.

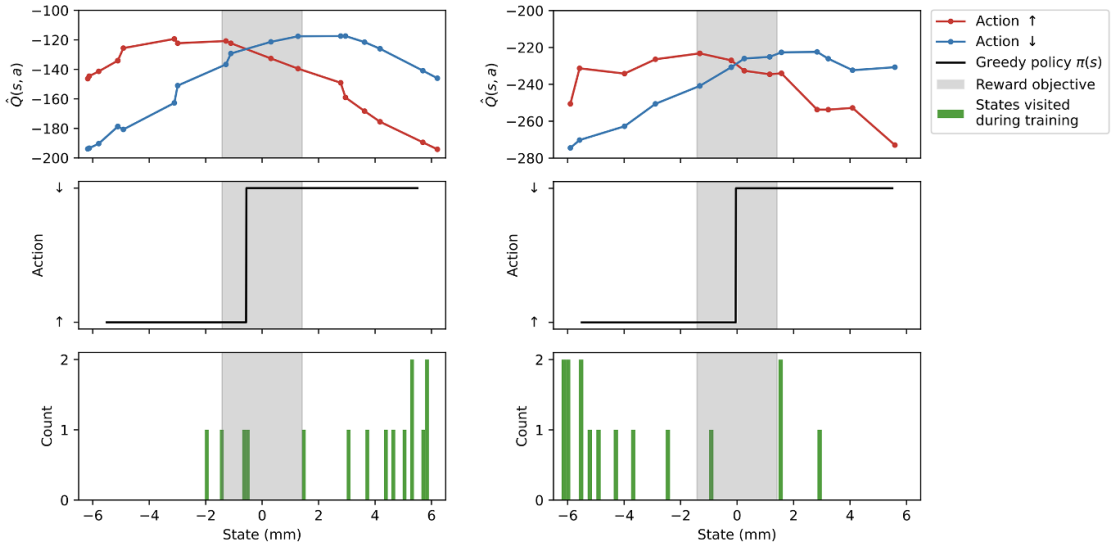

Figure 6. Q-value estimates after training (top), derived greedy policies (middle), and histograms of visited states during training (bottom) on the one-dimensional beam steering environment with continuous state space. The training was performed using SQA (left) and running on the D-Wave QA (right).

The agent has only visited 15 states during training in both cases. While the specific values of the learned Q-functions are not the same, in both cases, one obtains greedy policies which are 100% optimal. The hyperparameters were tuned with SQA and were directly used on the D-Wave QA. The results demonstrate that the RL agent can be trained successfully on a real QPU. Furthermore, the number of states visited during the entire training phase is comparable to what was required using SQA (figure 4, right, blue curve).

3.4. Study B: hybrid A-C scheme

3.4.1. Motivation and description of the algorithm

In the A-C scheme depicted in figure 1, it is possible to replace the classical Q-network with a QBM. This allows combining FERL with the policy update of a classical actor network and means that only one state-action pair evaluation is needed at every iteration.

Given the impressive results for Q-learning with FERL it is expected that the FERL critic also trains more efficiently in the hybrid A-C scheme, which should speed up the convergence of the actor network.

One main ingredient to the A-C scheme is the policy gradient evaluation (equation (3)) which includes the calculation of the derivative $\nabla_a Q(s, a|\theta)$, where $Q(s, a|\theta)$ is given by equation (6). Using numerical differentiation with finite differences is the most straightforward method of estimating this derivative

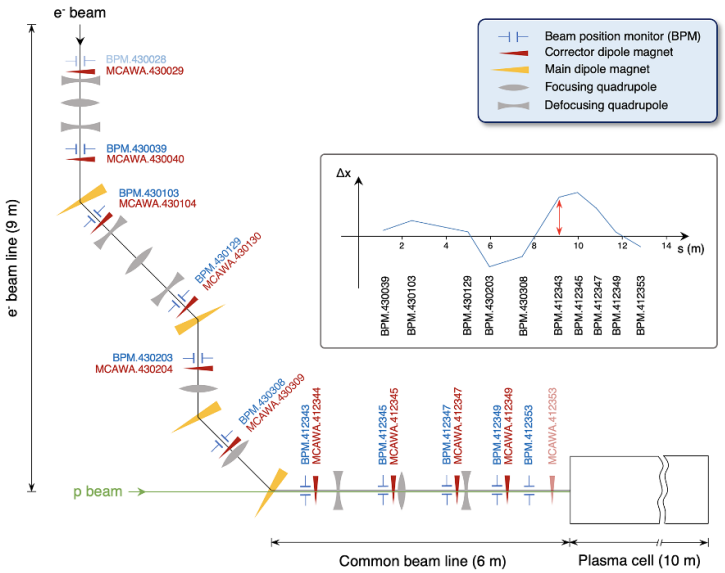

Figure 7. Schematic of the electron beam line of the CERN AWAKE experiment with 10 trajectory correctors and 10 beam position readings along the line.

without making additional assumptions on the probability distributions of spin configurations c. Alternatively, one could consider a semi-analytic approach. The calculation of $\partial_{\mathbf{v}=(s,a)} \mathcal{H}_v^{\text{eff}}$ is straightforward. However, the derivative of the entropy term cannot be easily evaluated except if $\mathbb{P}(c|\mathbf{v}) \approx 1$ for a specific spin configuration $c = c_0$, given a configuration of visible nodes $\mathbf{v}$, in which case the entropy term would be negligible. This paper does not develop the latter approach further, and numerical differentiation with finite differences is used instead.

The hybrid scheme also has the advantage that once the agent is trained, the (quantum) critic is no longer required at inference time. Since the policy is represented by a purely classical network, deploying the trained agent in a real-world environment becomes straightforward. This is particularly true in a sim-to-real RL setting, as discussed for the AWAKE beam line below.

3.5. The CERN AWAKE facility

The Advanced Wakefield Experiment (AWAKE) at CERN uses high-intensity 400 GeV proton bunches from the SPS as a plasma wakefield driver. Electron bunches are simultaneously steered into the plasma cell to be accelerated by the proton-induced wakefields. Electron energies up to 2 GeV have been demonstrated over a plasma cell of 10 m length corresponding to an electric field gradient of 200 MV m^{-1} [2]. The ultimate goal for AWAKE is to reach a field gradient of 1 GV m^{-1}. These numbers are to be compared to conventional accelerating structures using radio-frequency (rf) cavities in the X-band regime, which are currently limited to accelerating field gradients of about 150 MV m^{-1} [41].

3.5.1. Environment and RL task

The AWAKE electron source and beam line are particularly interesting for algorithm preparation and testing due to the high repetition rate and insignificant damage potential in case of losing the beam at accelerator components. The AWAKE electrons are generated in a 5 MV photocathode rf gun, accelerated to 18 MeV and then transported through a beam line of 12 m to the AWAKE plasma cell. The trajectory is controlled with 10 horizontal and 10 vertical steering dipoles according to the measurements of 10 BPMs, see figure 7. The BPM electronic read-out is at 10 Hz and acquisition through the CERN middleware at 1 Hz.

A hybrid A-C model was used for trajectory correction on the AWAKE electron beam line with the goal that the trained agents correct the line with similar accuracy as the response matrix-based singular value decomposition algorithm that has been traditionally used [42].

The RL problem of the AWAKE electron beam line is formalised as follows. The state s is defined by a ten-dimensional vector of horizontal beam position differences measured with respect to a reference trajectory. Similarly, the action is a ten-dimensional vector of corrector dipole magnet kick angles within a range of $\pm 300\,\mu$rad. Finally, the reward is defined as the negative rms of the measured beam trajectory with respect to the reference at all the BPMs.

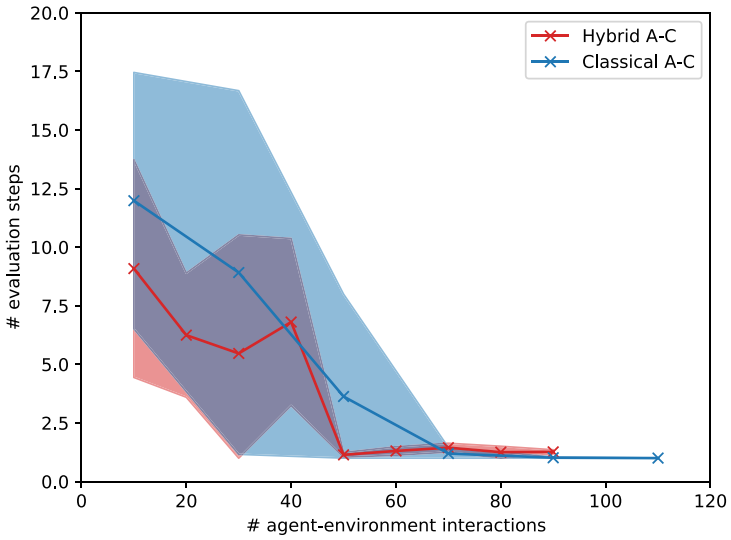

Figure 8. Comparison of the classical (blue) and hybrid (red) A-C algorithms in terms of the number of required agent-environment interactions, that is, sample efficiency, to reach optimal agent behaviour. These are results after extensive hyperparameter tuning. The hybrid A-C has been trained using SQA.

3.5.2. Results: SQA

Figure 8 shows the convergence of the classical vs hybrid A-C algorithms after hyperparameter optimisation. 15 instances of each algorithm were trained and evaluated on 500 random episodes. The points correspond to the mean number of steps required for the trained actor network to reach the reward objective, set to an equivalent trajectory rms of 1.6 mm, starting from a random initial state. The confidence bound corresponds to the standard deviation over the different instances and variation in the evaluation of episodes. The anomaly observed for the hybrid A-C at 40 agent-environment interactions is likely due to the stochastic behaviour of RL agent training combined with the relatively low number of individual trainings. Once the agents converge, the fluctuations between individual evaluations become small.

The number of agent-environment interactions does not necessarily correspond to the number of weight updates of the critic and actor networks since a fraction of the agent-environment interactions at the beginning of the training is obtained following a random policy to fill the replay buffer. The fraction of initial random interactions is a hyperparameter tuned with Ray Tune. The QBM consists of 4×4 unit cells of the D-Wave Chimera graph (8 qubits each) and features 128 qubits [25].

For the case of the AWAKE trajectory steering environment, an improvement in sample efficiency of about 30% of the hybrid A-C algorithm over the classical counterpart can be observed. With the hybrid A-C, a near-perfect behaving actor is obtained after training with 50 agent-environment interactions. The classical algorithm requires about 70 interactions. The advantage of the hybrid A-C algorithm in terms of sample efficiency may be more apparent for environments with more complex dynamics requiring a lot more agent-environment interactions for successful training. This is currently under study.

Figure 9 illustrates the performance differences between the classical and hybrid A-C agents after training for 50 and 70 agent-environment interactions, respectively. Histograms show the required number of steps and initial and final rewards during the evaluation phase. Comparing the results for the hybrid A-C after 50 (red) and 70 (orange) interactions demonstrates that there is no additional improvement in performance with more training iterations, neither in the number of required steps to reach the objective nor in the final reward values achieved. On the other hand, the classical A-C agent trained for 50 interactions (blue) has not yet converged. This is visible from the distributions of required steps and the final rewards. In about 8% of evaluation episodes, the agent did not manage to solve the task in the 10 steps available. Accordingly, the final rewards are far from the objective for these cases. Finally, comparing the performance of the classical A-C after 70 interactions (green) with any of the hybrid A-C results, the classical agent shows better behaviour in terms of required steps: it always reaches the objective within 1 step, while the hybrid agents require 2 steps in about 5%–7% of cases. Regarding final rewards, the hybrid agents achieve slightly better mean values than the best classical agent, although with a larger variance. Given the statistical fluctuations, no clear statement can be made here.

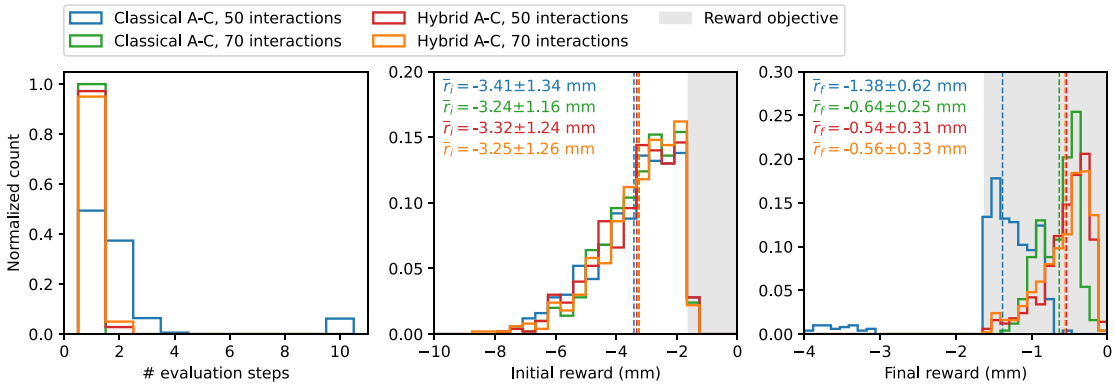

Figure 9. Evaluation of the RL agents trained with a classical A-C algorithm (DDPG) and the hybrid A-C (SQA), respectively, after 50 and 70 interactions between agent and environment. Shown are the number of steps taken (left), the distribution of initial rewards (middle), and the distribution of the final rewards at the end of an episode (right). The evaluation has been performed with 500 randomly initialised episodes.

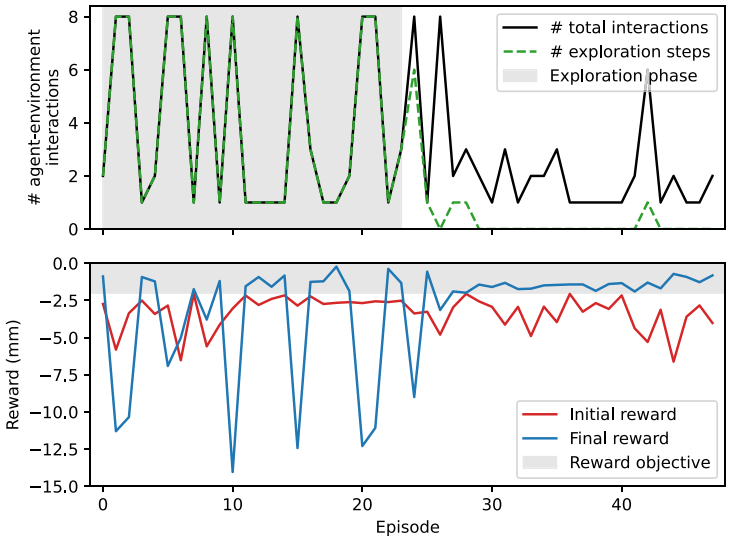

Figure 10. Single RL agent training evolution on D-Wave advantage systems using the simulated AWAKE environment with a reward objective of −2 mm.

3.5.3. Results: D-Wave advantage QA

The developed hybrid A-C algorithm was also tested on the D-Wave Advantage system. Due to the limited QPU time available, the problem was slightly modified to ensure that it could be solved within the given time frame. To that end, the reward objective was relaxed from an rms value of 1.6 mm to 2 mm. Using SQA, training parameters, such as the number of agent-environment interactions and replay batch size, were adjusted to adapt the problem to run successfully within the available QPU time.

The training evolution of the RL agent in the simulated AWAKE environment is shown in figure 10. From around episode 30 onwards, the agent consistently reaches the reward objective. The evaluation of the actor network trained on the D-Wave Advantage (blue) and SQA (red) on 500 randomly initialised episodes are shown in figure 11. The plot on the left-hand side displays the number of steps required to reach the reward objective from the initial random state. The middle and right-hand side plots show the distribution of initial and final rewards. The QA-trained agent typically reaches the reward objective within 1 step (80% of cases). Occasionally, it requires 2 steps (18%), and rarely 3 steps (2%), but it reduces the beam trajectory excursions to an rms better than 2 mm in 100% of cases.

A direct comparison with an RL agent trained with SQA is shown in figure 11. Both agents were trained with identical hyperparameters. The plots show that their performance is comparable, particularly in terms of the number of steps required to reach the reward objective. The agent trained on the D-Wave QA seems to produce marginally better results on average as can be seen from the distributions of final rewards in figure 11.

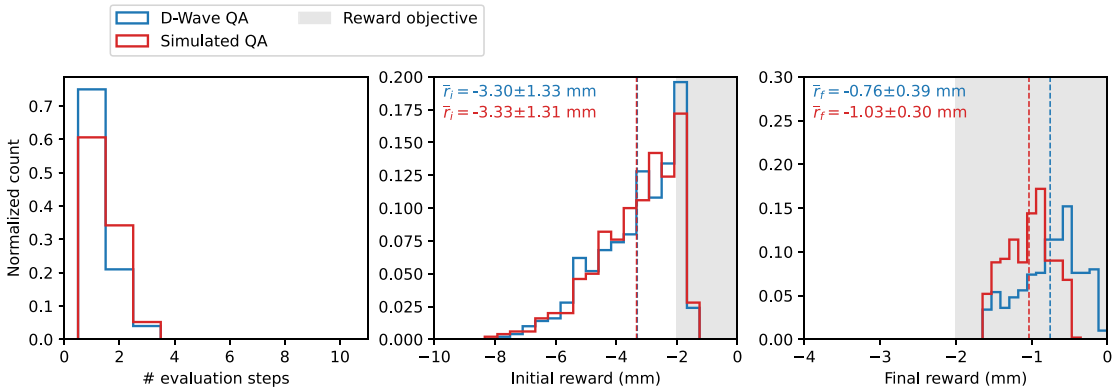

Figure 11. Evaluation of the RL agent trained on D-Wave QA hardware (blue) and simulated QA (red) using identical hyperparameters. Shown are the distributions of the number of steps required to reach the reward objective from random initial states (left), as well as the initial (middle) and final (right) rewards.

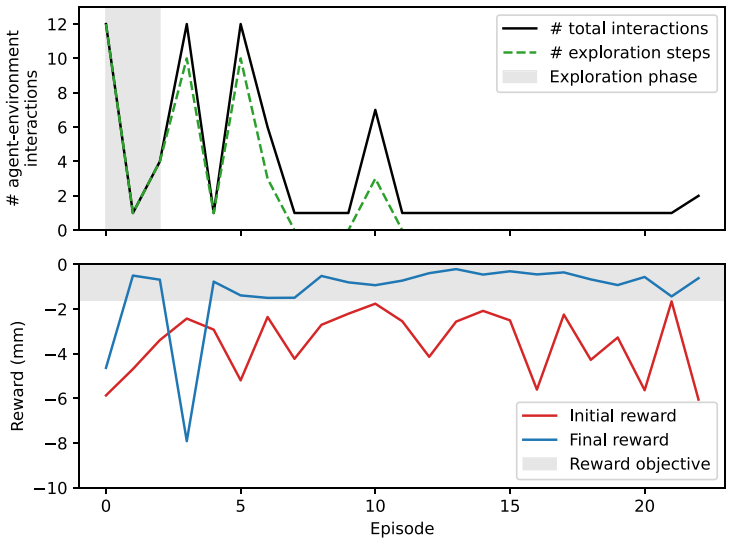

Figure 12. Training of RL agent (SQA) on the simulated AWAKE environment for deployment on the real beam line.

3.6. Results: deployment on real beam line

The hybrid A-C agents were also evaluated in the real AWAKE environment to test sim-to-real transfer. Both agents—trained on D-Wave QA hardware (figure 13, bottom row) and with SQA (figure 13, top row)—were tested. The training evolution is given in figures 10 and 12 with reward objectives set to −2.0 mm (QA) and −1.6 mm (SQA), respectively. The number of agent-environment interactions is shown (top) together with the evolution of initial and final rewards during the training period. As explained above, using the hybrid A-C scheme, the QBM-based critic is no longer required at inference time since the policy is encoded entirely in the classical actor network.

Figure 13 shows histograms of the required number of steps as well as initial and final rewards on the simulated (red) and real (blue) AWAKE environments during evaluation. Note that the agents were deployed in the real-world environment without additional training. Both agents can solve the tasks in the simulated and real AWAKE environment. Also, both agents perform better in the simulated environment in terms of final rewards and number of steps required indicating small differences between the simulation and the real beam line. While the measurements on the real beam line were performed back-to-back for the SQA- and QA-trained agents, the distribution of initial rewards (blue, middle column) does not fully overlap, with a tendency towards more challenging episodes for the QA-trained agent (bottom row). The QA-trained agent failed to solve the task for one of the initial states in the real environment, likely indicating that its training had not fully converged due to the limited QPU time available.

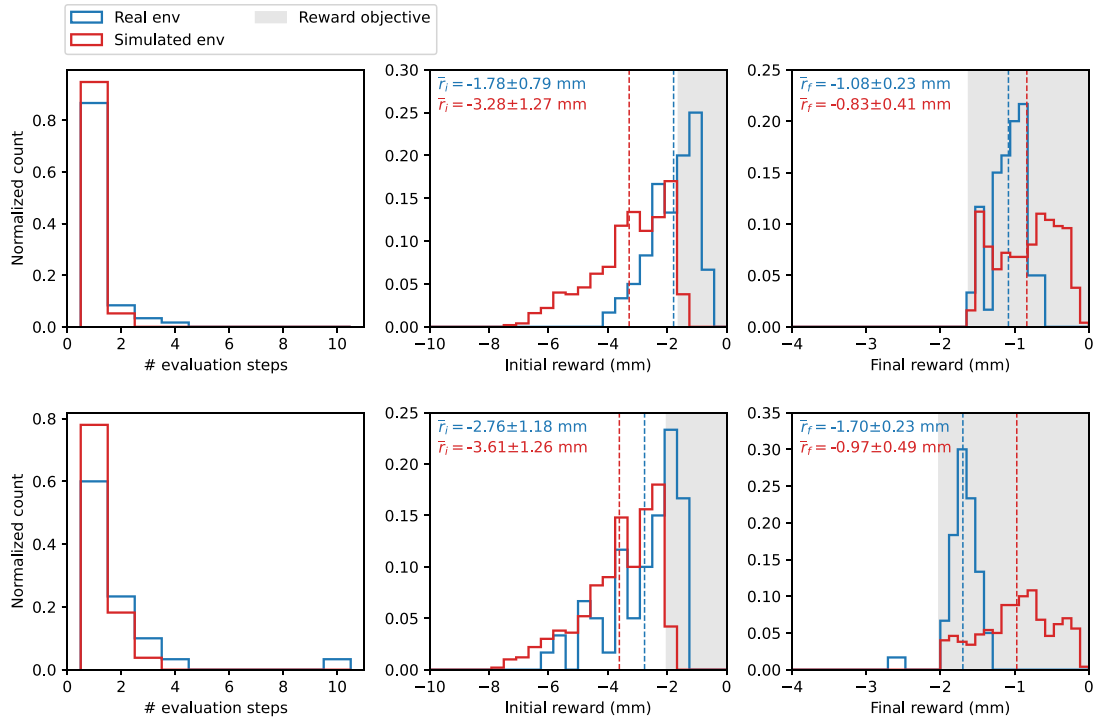

Figure 13. Top: evaluation of an SQA-trained agent on the simulated and real AWAKE environments (reward objective: −1.6 mm). Bottom: evaluation of the D-Wave QA-trained agent in the simulated and real AWAKE environments (reward objective: −2.0 mm). Shown are the distributions of the number of steps required to reach the reward objective from random initial states (left), as well as the initial (middle) and final (right) rewards.

4. Conclusions

Earlier research has demonstrated that FERL with QBMs can remarkably increase the sample efficiency compared to the classical deep Q-learning (DQN) algorithm. The results were obtained for control problems defined in discrete action-space environments. This article confirms the improved training efficiency of FERL compared to DQN on the example of the beam trajectory correction task in the TT24-T4 beam line delivering protons to CERN's fixed-target physics experiments. The task was defined using discrete actions but continuous states, extending the applicability of previously reported FERL methods in the literature where the state and action spaces were both discrete. Another interesting result was that FERL learns very efficiently even without experience replay, but the sample efficiency improves further and by a significant amount when including it.

In a second study, this paper investigated whether the increased sample efficiency could be exploited for continuous action-space environments often encountered in real-world control problems. To that end, the authors developed a hybrid RL model that can handle continuous states and actions. Inspired by the A-C scheme of the DDPG algorithm, the critic network was replaced by a clamped QBM and trained using the FERL approach. Based on the simulated ten-dimensional trajectory correction problem of the AWAKE electron beam line, RL agents were trained with the new method employing SQA and D-Wave Advantage QA hardware. The agents were then successfully evaluated in the real facility. An increase in sample efficiency was again observed, but the improvement compared to the classical algorithm was not as significant as for the discrete action space problem. Further studies will be carried out to test whether this is generally true or only for the problem studied here, which has linear dynamics. As a final note, the chosen hybrid quantum A-C architecture also has a practical advantage—at inference, that is, in the accelerator control room, only the classical actor network is required which greatly simplifies deployment and usage for accelerator operation.

Data availability statement

The data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

This work was partially funded by MCIN/AEI/10.13039/501100011033 under Grant PID2020-119082RB-C22, by Gobierno del Principado de Asturias under Grant AYUD/2021/50994, and by the Quantum Spain project funded by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government and the European Union through the Recovery, Transformation and Resilience Plan—NextGenerationEU.

Appendix. Complementary results

A.1. FERL Q-learning with discrete, binary state-action space

Figure A1 shows the results of policy optimality vs number of training steps with the classical deep Q-learning (red) and FERL (blue) agents, without (left) and with (right) experience replay. Again, the study is based on the one-dimensional TT24-T4 target beam steering environment described in section 3.3.1. However, this time it uses a discrete, binary state-action space, in analogy to the studies in [8].

Similarly to figure 4, which was obtained for a continuous state space, the results here show that the FERL agent manages to outperform the classical deep Q-learning agent. However, the gain in sample efficiency is much smaller than in the former case. Again, the comparison between the left and right plots confirms the importance of experience replay for both types of algorithms. In the best case, the FERL algorithm reliably reaches 100% optimality with about 100 training steps, while classical deep Q-learning requires about 220 steps.

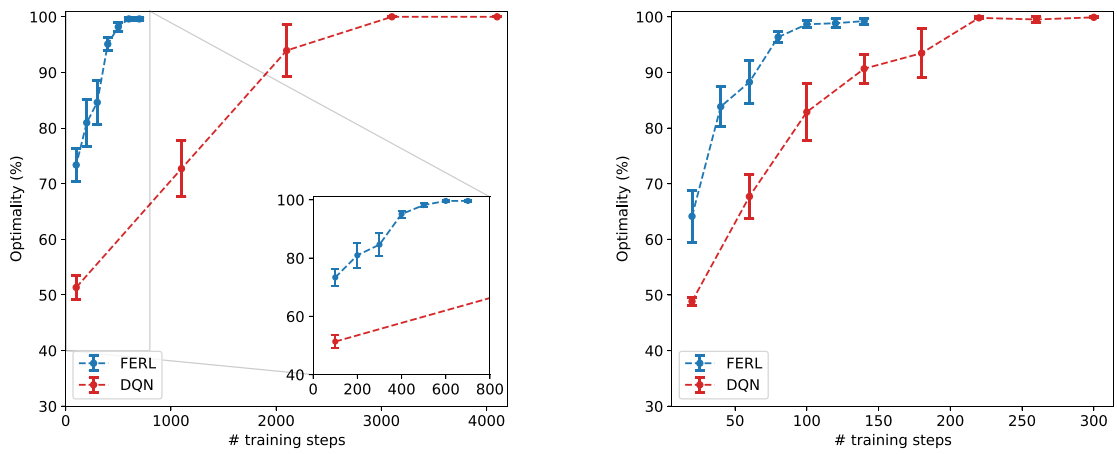

Figure A1. One-dimensional beam steering environment with a discrete, binary state-action space: convergence study for agent optimality vs the number of training steps *without* (left) and *with* (right) experience replay for a classical DQN (red) and an FERL (blue) agent.

ORCID iDs

Michael Schenk https://orcid.org/0000-0001-9438-812X
Elías F Combarro https://orcid.org/0000-0003-3808-4273
Michele Grossi https://orcid.org/0000-0003-1718-1314
Verena Kain https://orcid.org/0000-0002-3135-2004
Sofia Vallecorsa https://orcid.org/0000-0002-7003-5765

References

[1] Gatignon L 2018 *Rev. Sci. Instrum.* **89** 052501
[2] Adli E *et al* 2018 *Nature* **561** 363–7
[3] Bartosik H and Rumolo G 2022 Performance of the LHC injector chain after the upgrade and potential development (arXiv:2203.09202)
[4] Montbarbon E *et al* 2019 *Nucl. Instrum. Methods Phys. Res.* B **461** 98–101
[5] Sutton R S and Barto A G 2018 *Reinforcement Learning: An Introduction* 2nd edn (A Bradford Book)
[6] Mnih V *et al* 2015 *Nature* **518** 529–33

[7] Sallans B and Hinton G E 2004 *J. Mach. Learn. Res.* **5** 1063–88
[8] Levit A, Crawford D, Ghadermarzy N, Oberoi J S, Zahedinejad E and Ronagh P 2017 Free energy-based reinforcement learning using a quantum processor (arXiv:1706.00074)
[9] Crawford D, Levit A, Ghadermarzy N, Oberoi J S and Ronagh P 2018 *Quantum Inf. Comput.* **18** 51–74
[10] Lillicrap T P *et al* 2016 Continuous control with deep reinforcement learning *4th Int. Conf. on Learning Representations, ICLR 2016 (San Juan, Puerto Rico, 2–4 May 2016)* (*Conf. Track Proc.*) ed Y Bengio and Y LeCun (arXiv:1509.02971)
[11] Lin L J 1992 *Mach. Learn.* **8** 293–321
[12] Ackley D H, Hinton G E and Sejnowski T J 1985 *Cogn. Sci.* **9** 147–69
[13] Melko R G, Carleo G, Carrasquilla J and Cirac J I 2019 *Nat. Phys.* **15** 887–92
[14] Hinton G E and Sejnowski T J 1983 Optimal perceptual inference *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 448–53 (available at: www.cs.toronto.edu/∼hinton/absps/optimal.pdf)
[15] Sussmann H J 1988 Learning algorithms for Boltzmann machines *Proc. 27th IEEE Conf. on Decision and Control* vol 1 pp 786–91
[16] Younes L 1996 *Appl. Math. Lett.* **9** 109–13
[17] Le Roux N and Bengio Y 2008 *Neural Comput.* **20** 1631–49
[18] de Gennes P G 1963 *Solid State Commun.* **1** 132–7
[19] Johnson M *et al* 2011 *Nature* **473** 194–8
[20] Farhi E, Goldstone J, Gutmann S and Sipser M 2000 Quantum computation by adiabatic evolution (arXiv:quant-ph/0001106)
[21] Born M and Fock V 1928 *Z. Phys.* **51** 165–80
[22] Cubitt T S, Perez-Garcia D and Wolf M 2015 *Nature* **528** 207–11
[23] McGeoch C C 2014 *Adiabatic Quantum Computation and Quantum Annealing* (*Synthesis Lectures on Quantum Computing*) (Morgan & Claypool Publishers) (https://doi.org/10.1007/978-3-031-02518-1)
[24] Crosson E J and Lidar D A 2021 *Nat. Rev. Phys.* **3** 466–89
[25] D-Wave Systems Inc. 2022 *D-Wave System Documentation* (available at: https://docs.dwavesys.com/) (Accessed 4 August 2022)
[26] Schuld M, Sinayskiy I and Petruccione F 2015 *Contemp. Phys.* **56** 172–85
[27] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195–202
[28] Paparo G D, Dunjko V, Makmal A, Martin-Delgado M A and Briegel H J 2014 *Phys. Rev.* X **4** 031002
[29] Jerbi S, Gyurik C, Marshall S, Briegel H J and Dunjko V 2021 Parametrized quantum policies for reinforcement learning *Advances in Neural Information Processing Systems 34: Annual Conf. on Neural Information Processing Systems 2021, NeurIPS 2021 (Virtual, 6–14 December 2021)* pp 28362–75 (arXiv:2103.05577)
[30] Chen S Y C, Yang C H H, Qi J, Chen P Y, Ma X and Goan H S 2020 *IEEE Access* **8** 141007–24
[31] Lockwood O and Si M 2020 Reinforcement learning with quantum variational circuits *Proc. 16th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'20)* (AAAI Press) (https://doi.org/10.5555/3505464.3505499)
[32] Chen S Y C, Huang C M, Hsing C W, Goan H S and Kao Y J 2022 *Mach. Learn.: Sci. Technol.* **3** 015025
[33] Jerbi S, Trenkwalder L M, Poulsen Nautrup H, Briegel H J and Dunjko V 2021 *PRX Quantum* **2** 010328
[34] Neukart F, Von Dollen D, Seidel C and Compostella G 2018 *Front. Phys.* **5** 71
[35] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W 2016 OpenAI gym (arXiv:1606.01540)
[36] Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M and Dormann N 2021 *J. Mach. Learn. Res.* **22** 1–8 (available at: https://jmlr.org/papers/v22/20-1364.html)
[37] Morino S 2018 Sqaod: simulated quantum annealing library (available at: https://github.com/shinmorino/sqaod) (Accessed 4 August 2022)
[38] Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez J E and Stoica I 2018 Tune: a research platform for distributed model selection and training (arXiv:1807.05118)
[39] Akiba T, Sano S, Yanase T, Ohta T and Koyama M 2019 Optuna: a next-generation hyperparameter optimization framework *CoRR* (arXiv:1907.10902)
[40] D'Alessandro G L *et al* 2021 Target bypass beam optics for future high intensity fixed target experiments in the CERN North Area *Proc. Int. Particle Accelerator Conf. (IPAC'21)* vol 12 (JACoW Publishing) pp 3046–8 (available at: https://jacow.org/ipac2021/papers/wepab185.pdf)
[41] Agustsson R, Carriere R, Chimalpopoca O, Dolgashev V, Gusarova M A, Kutsaev S V and Yu Smirnov A 2022 *J. Phys. D: Appl. Phys.* **55** 145001
[42] Chung Y, Decker G and Evans K 1993 Closed orbit correction using singular value decomposition of the response matrix (Argonne National Laboratory) (available at: www.osti.gov/servlets/purl/10178130)