PAPER • OPEN ACCESS

Microsecond-latency feedback at a particle accelerator by online reinforcement learning on hardware

To cite this article: Luca Scomparin *et al* 2026 *Mach. Learn.: Sci. Technol.* **7** 025056

MACHINE LEARNING
Science and Technology

PAPER

Microsecond-latency feedback at a particle accelerator by online reinforcement learning on hardware

Luca Scomparin*, Michele Caselle, Andrea Santamaria Garcia, Chenran Xu, Edmund Blomley, Timo Dritschler, Akira Mochihashi, Marcel Schuh, Johannes L Steinmann, Erik Bründermann, Andreas Kopmann, Jürgen Becker, Anke-Susanne Müller and Marc Weber

Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany
* Author to whom any correspondence should be addressed.

E-mail: luca.scomparin@kit.edu

Keywords: edge AI, FPGA, reinforcement learning, particle accelerator

OPEN ACCESS

RECEIVED
10 December 2025

REVISED
10 March 2026

ACCEPTED FOR PUBLICATION
2 April 2026

PUBLISHED
15 April 2026

Abstract

The commissioning and operation of future large-scale scientific experiments will challenge current tuning and control methods. Reinforcement learning (RL) algorithms are a promising solution due to their ability to dynamically adapt to changing environments and consider delayed consequences. In many real-world applications, RL policies must produce actions in real time, often within microseconds to milliseconds, imposing significant constraints on system latency and computational overhead that conventional machine learning libraries are not designed to handle. To control phenomena in real time at these timescales, RL needs to be deployed on-the-edge, namely on dedicated hardware located near the system it controls, without relying on a host CPU or cloud-based inference. In this work we present the design and deployment of an *experience accumulator system* in a particle accelerator. In this system, deep-RL algorithms run using hardware acceleration and act within a few microseconds, enabling the use of RL for control of phenomena like beam instabilities. The training uses the collected data offline to reduce the number of operations carried out on the acceleration hardware. The proposed architecture was tested in real experimental conditions at the Karlsruhe research accelerator, a synchrotron light source, where the system was used to control artificially induced horizontal betatron oscillations in real-time, with a control loop period of just 2.7 μs. The results showed a performance comparable to the commercial feedback system available at the accelerator, demonstrating the viability and potential of this approach. Due to the self-learning and reconfiguration capability of this implementation, a seamless application to other control problems is possible. Applications range from particle accelerators to large-scale research and industrial facilities.

1. Introduction

Machine learning (ML) techniques have been identified as a key technology for current and future particle accelerators [1–3]. In the case of control problems, a promising approach is the use of reinforcement learning (RL), a paradigm where an agent is trained to learn a policy in order to maximize a reward function, encoding a control goal, acting on an environment based on a set of observations. RL algorithms have already been proven to be a powerful tool through their application to control problems in several large-scale facilities [4–11].

Typically, a controller must operate within time scales comparable to those of the controlled dynamics. The fact that the evaluation of a control policy needs to be performed in a scheduled time frame is by definition a real-time constraint [12]. Failure to meet this kind of requirement means the controller could be partially effective or even completely ineffective. As such, when the inference time reaches the microsecond timescale, the use of conventional CPUs becomes challenging, as the several layers of abstraction and caching between the application and hardware limit the achievable latency performance. Specifically, the latency achieved can not only be higher than the requirements, but its high variance

could produce spurious, unpredictable violations of these timing constraints. Different kinds of computing hardware can circumvent these limitations by combining a higher degree of parallelization with a system of lower intrinsic overhead. Modern heterogeneous computing platforms are a good candidate for this, combining conventional CPUs with a field programmable gate array (FPGA) and an artificial intelligence (AI) accelerator [13]. An FPGA allows the definition of custom programmable digital logic that can reach levels of parallelism that are constrained only by the number of programmable cells available on the device. Moreover, they allow precise control over the timing of each operation. The AI accelerator allows the efficient execution of more specific tasks (e.g. floating point vector multiplication), for which FPGAs are less suitable. These three components share memory, allowing a customized and optimized data-flow architecture.

One of the main issues that prevents the application of RL to many large-scale facilities is their sample efficiency, the amount of interaction with an environment required to fully train an agent. This means that the data necessary to train a successful agent is usually difficult to obtain in the real world due to their low repetition rates [14]. This issue can be mitigated by pre-training on a simulation, with the risk that the agent might perform sub-optimally in case the simulation differs significantly from the real world [15]. A different approach is to use more sample-efficient algorithms, but those tend to complicate the tuning of the hyperparameters [8]. The benefit of bringing RL to facilities operating on microsecond timescales, like numerous synchrotron light sources, MHz-rate free electron lasers, and circular colliders, is that sufficient training data can be generated in real time, bypassing the need of training in simulation. Conversely, the constraints on the time taken to select an action require a more careful implementation of the agent.

In this work, we developed a closed-loop RL feedback system that was successfully deployed and commissioned at the Karlsruhe research accelerator (KARA), an accelerator test facility and synchrotron light source. This novel scheme enables direct RL online training in the microsecond time domain, setting the precedent as the first implementation of its kind in particle accelerators.

Notably, this approach can tackle non-linear control problems with interaction rates in the megahertz range such as the microbunching instability in synchrotron light sources [16–18]. In order to aid the first deployment of such a system, a simpler problem allowing more straightforward diagnosis of hardware issues that can potentially arise for such novel implementations. Having a standard, well-tested controller for the problem is strongly preferred, as it allows to test the system independently from the RL algorithm, and also provide a first metric of how well the RL-based controller performs.

The remainder of this work is structured as follows. Section 2 gives an overview of RL and the proximal policy optimization (PPO) algorithm, together with an introduction of FPGA implementations. Section 3 describes the experience accumulator architecture developed as part of this work. Section 4 discusses the integration of the hardware system to carry out measurements at KARA, followed by the experimental results in section 5. Finally, the implications of these results are detailed in section 6.

2. RL algorithms

In RL the environment is modeled as a Markov decision process (MDP), a 4-tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R})$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ are the transition dynamics, and $\mathcal{R}$ is the set of all rewards. The goal of the control problem is encoded in the scalar reward value R. An RL agent is tasked to maximize the expected return G_t in each step t, defined as the cumulative reward in all future steps

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

where $\gamma \in [0,1)$ is the discount factor, used to bound the cumulative reward in infinite horizon problems, extending the MDP to $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma)$.

This framework models decision-making in partially stochastic control processes and incorporates the Markov property, where the conditional probability distribution $\mathcal{P}$ of future states depends only on the present state and the selected action.

Often the full state of the environment is not directly accessible and needs to be inferred from observations, giving place to partially observable MDP defined as $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma,\mathcal{O},\epsilon)$, where $\mathcal{O}$ is the space of possible observations and ϵ the probability of transitioning to a new state s_t when o_t is observed $\epsilon = P(s_t|o_t)$.

Modern deep RL (DRL) algorithms approximate the policy and value functions by deep neural networks (DNNs), allowing RL algorithms to tackle complex environments with continuous state and

action spaces. DRL has achieved state-of-the-art performance in continuous control tasks through policy-based and value-based methods. Policy-based RL is a widely used class of modern RL algorithms, where weights of the DNN policy are updated by ascending the gradient of expected cumulative reward, typically estimated using sampled trajectories. They exhibit stable convergence and allow for stochastic policies, which are beneficial for exploration and robustness. However, they tend to suffer from high variance in gradient estimates, which can lead to slow and unstable learning. In contrast, value-based methods, like Q-learning and DQN [19], estimate the expected return of state-action pairs and are often more sample-efficient and effective in discrete action settings. Yet, they can be difficult to scale to continuous actions, and they rely on argmax operations that may result in unstable behavior, particularly in noisy environments. Actor-critic methods combine the strengths of both policy-based and value-based RL. They consist of two core components: the actor, which represents the policy and selects actions, and the critic, which estimates a value function and provides feedback to guide the policy updates. This architecture enables the actor to perform low-variance policy updates by leveraging value estimates from the critic, while still supporting stochastic or deterministic policy learning as in policy gradient methods. Popular actor-critic algorithms such as PPO [20], soft actor-critic (SAC) [21], and TD3 [22] are considered standard benchmarks thanks to their stability, sample efficiency, and ability to handle high-dimensional, continuous control tasks with relatively minimal tuning.

PPO is an on-policy, actor-critic algorithm. The actor is a policy network $\pi_\theta(a|s)$, where θ are the actor's parameters. The critic is a value network $V_\phi(s)$, with ϕ being the critic's parameters. It estimates the future returns of a state s for following the current policy. PPO aims to update the policy gradually in a direction that improves performance while ensuring the policy remains close to the previous iteration [20], preventing large deviations that can cause training instability and performance degradation. This is achieved by constraining the gradient update of the policy network, called *gradient clipping*.

The objective function used for the update is defined as

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}\left[\min\left(r_t(\theta)\hat{A}_{\theta,\phi}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{\theta,\phi}\right)\right], \tag{2}$$

where $\hat{A}_{\theta,\phi}$ is the advantage function, which uses the critic model to estimates how much improvement taking a certain action a_t brings compared to the current policy π_θ. $r_t(\theta)$ is the probability ratio between the old and the new policy

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}. \tag{3}$$

The update size is limited by a factor ϵ which is set to be $\epsilon = 0.2$ in this paper. The parameters of the actor and critic networks are updated using gradient ascent

$$(\theta,\phi) \leftarrow (\theta,\phi) + \alpha\nabla_{\theta,\phi}L^{\mathrm{total}}(\theta,\phi). \tag{4}$$

In this work, we chose the PPO algorithm as it is more robust to hyperparameter choices and easier to deploy than other options. One shortcoming of PPO is its relatively low sample efficiency, which is commonly addressed by training in simulations that can be highly parallelized. In our study, however, the sample efficiency is not a concern, because the achievable interaction rate with the environment is as high as the revolution frequency of the synchrotron storage ring, corresponding to MHz range.

It is worth noting that, given the interaction with an active environment, most non-simulated RL agents are subject to real-time constraints. Depending on the time scale of the environment, this requirement could be addressed by simply allocating enough computing power to the agent. When the latency requirements become more demanding, the compound overhead of transferring data and evaluating the agent can mandate more adaptable computing architectures like FPGAs or heterogeneous platforms, in order to control the scheduling of the data-processing pipeline better.

2.1. Related work: RL on FPGA

A brief outline of the currently available implementations of RL on FPGA is described below. A more comprehensive review can be found in [23]. There are two main classes of implementations, depending on the nature of the value function and policy. Several works store the action-value function in a tabular form, and based on this choose the action with the maximum expected cumulative reward for each state [24–28]. The main issue of this kind of system is that its resource usage grows exponentially with the size of the observation and action space, while not directly applicable to environments with continuous action and observation spaces. These kinds of environments are widely seen in particle accelerator applications and fit the problem of this work better.

A different approach is DRL, where the value function and policies are approximated with a NN [29–35]. This approach scales better with increasing size of the observation and action vector, but tends to have a more complex training routine. These implementations lack the flexibility of choosing different training algorithms and changing the NN topology during operation. Distinct algorithms and their respective hyperparameter selections might exhibit different performances based on the problem at hand. Therefore, the capability of flexibly choosing these components greatly reduces the deployment effort and expedites the development process. Moreover, the policy NN implementations shown in the [24–35] do not have real-time applications in mind, and as such the application to real-time particle accelerator controls would be limited.

For these reasons, we have chosen to implement a novel DRL system on custom hardware with the additional capability of performing such dynamic reconfiguration and naturally fulfilling microsecond-scale real-time constraints.

3. Experience accumulator architecture

In this study, we propose to implement the RL control system using an *experience accumulator* architecture. Depending on the complexity of the algorithm, the *training* processes in modern DRL can take a large amount of time compared to the time required for inference of the policy. As such, it is not possible to perform them at a microsecond-level timescale. A possible solution is to implement only a general real-time policy $\pi^{(\text{edge})}$ for *inference*, using low-latency computing platforms. During application time, the real-time policy network only needs to perform *forward passes*, i.e. predicting the next action a_i based on the observed signal o_i. The interactions of the policy NN with the environment are recorded for T steps, providing state-action-reward tuples $\{(s_i, a_i, r_i)_{i=1,\dots,T}\}$, that can be used for asynchronously training an emulated copy of the agent $\pi^{(\text{CPU})} \leftarrow \pi^{(\text{edge})}$ on conventional computing platforms, such as a CPU. As such, it is possible to achieve a greater degree of flexibility and abstraction (figure 1). This comes at the expense of higher overhead during training, while still maintaining low inference latency. For the PPO [20] algorithm agent used in this work, only the actor NN is implemented on hardware, while the critic can be evaluated at training time.

A similar approach is presented in a previous work [36], although it is applied to a system with a less stringent latency requirement, thus not requiring heterogeneous computing platforms, but an x86 processor. Additionally, in this study, we further extend the capability of the experience accumulator architecture by introducing a post-hoc reward definition. In the definition of RL control problems, the reward is provided step by step by the environment. In the case of a high-speed feedback system, interacting with a real-time environment, it would then become necessary to perform this computation at pace with the environment. The need for a real-time reward function implementation increases the complexity of the system, as it needs to be implemented in the FPGA or AI accelerator, requiring specialized expertise.

In the experience accumulator architecture, a further simplification is possible. The rewards obtained by an agent are only necessary during its training. If the reward is a function depending purely on the observation and action vectors, it is possible to skip its real-time computation and perform it at training time based on the stored observation-action tuples. This *training time reward definition* technique reduces development efforts as it allows the function to be implemented on a CPU, where the expertise required is more commonly available. In the implementation used in this work, the reward function is developed in Python and specified by the experimenter before starting the training, in this way simplifying the *reward engineering*.

4. Hardware infrastructure

In order to simplify the deployment of RL policies in an experience accumulator architecture, all the FPGA infrastructure necessary for interfacing with the particle accelerator and storing the agent's experience were implemented into an AMD-Xilinx Extensible Platform called KINGFISHER [37]. This approach has the effect of decoupling the low-level operations allowing the development of the policy and feature extraction algorithm in high-level programming languages, thanks to the aid of tools like high level synthesis.

The device of choice is the AMD-Xilinx Versal VCK190 [38], a novel heterogeneous computing platform comprising a FPGA and ARM processor in a system-on-chip architecture, with an AI engine (AIE) array capable of accelerating multiplication intensive tasks, as, for instance, NN inference.

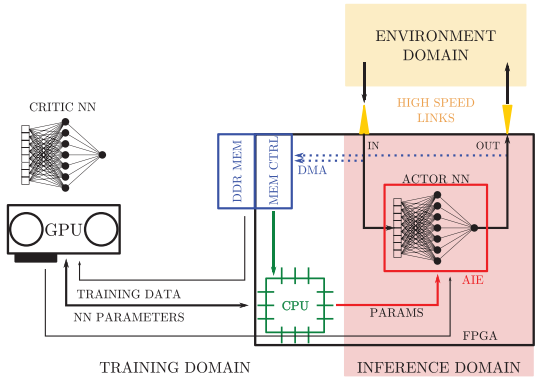

Figure 1. Schematic drawing of the experience accumulator architecture showing the partition between real-time or inference domain (on the right) and training-time domain (on the left) on a heterogeneous platform.

The KINGFISHER platform receives data via a 40 Gbps Aurora 64b/66b link [39]. This data can be used to create the observation data-stream. The action data-stream is used to control a digital to analog converter (DAC) to produce an analog control signal. For the purpose of future experiments special digital output interfaces are also available. The observation-action data-streams are written to the DDR memory, without active intervention of the CPU, by means of a direct memory access (DMA) block implemented in the FPGA. The ARM processor in the Versal core then copies the data in the DDR to a file that is made available over the network. Moreover, a control server makes the parameters of KINGFISHER available via the EPICS [40] control system. A trigger input is available to start the agent action at a precise time. Moreover, a counter allows the interaction with the environment only for a limited number of steps. In this way, the influence of an agent on the accelerator can be reduced to prevent beam losses or the excitation of instabilities.

In order to guide the exploration of RL algorithms, a stochastic component is usually needed. Thus an intellectual property (IP) core implementing the permuted congruential generator algorithm [41] produces a continuous stream of 32-bit floating pseudorandom-numbers uniformly distributed between 0 and 1 at a sample rate of 125 MHz.

For the control of the horizontal betatron oscillation (HBO) the specific deployment configuration is shown in figure 2. To sample the fast-peaked signals with high MHz rates commonly generated by diagnostic instrumentation at accelerator-based light sources like synchrotrons, the KIT-developed KAPTURE system [42] is utilized. This system functions as a digitizer, enabling the sampling of four channels at a frequency equal to the accelerator's radio frequency (RF), with a selectable sampling interval down to 3 ps. The system utilizes a Highflex card with the capability of performing transfers directly to a GPU with the GPUDirect technology [43]. This setup provides bunch-by-bunch and turn-by-turn data. In order to obtain the bunch position relevant for the control of the HBO, this infrastructure is used to sample the horizontal position signal from an analog Dimtel BPMH-20-2G hybrid [44] combining the four signals from a button beam position monitor (BPM). The action analog signal is fed into a broadband amplifier and then applied to a stripline kickers used to influence the beam. A single BPM and action kickers are used as agent input and outputs. No calibration is performed on the input signal. Up to eight BPM signals could be added as inputs of the system [42], but this was not deemed necessary for the current experiment.

The reconfigurable nature of FPGAs makes this hardware architecture adaptable to other facilities. Specifically, the KAPTURE firmware needs to be adapted to the specific RF frequency and harmonic number of the accelerator. The RL model is agnostic of the specific hardware it is driving, so other feedback methods could in principle be employed by developing their specific interface.

4.1. Low-latency RL

The algorithm used in the current work is PPO. This choice was dictated by its stability with respect to the change of hyperparameters. The reduced sample efficiency compared to off-policy algorithms like SAC does not affect the current application, as it is counterbalanced by the high experience collection rate. Other RL algorithms are nonetheless easy to integrate thanks to the experience accumulator architecture.

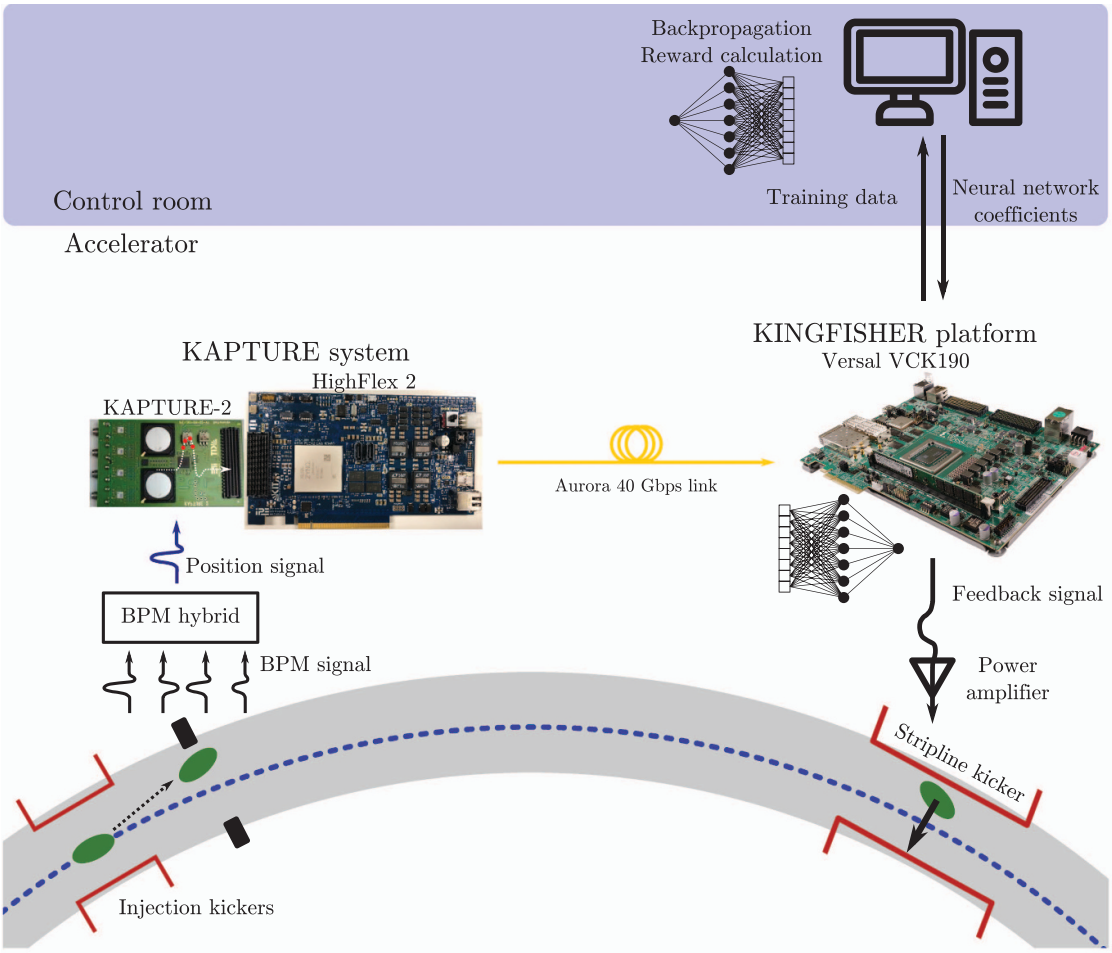

Figure 2. Drawing of the hardware infrastructure employed in this work. The bunch position in the beam pipe is measured by processing a BPM signal with an analog hybrid. The produced analog signal is sampled with KAPTURE and then forwarded by an HighFlex 2 board through high-speed links to a Versal device. The KINGFISHER system programmed on this Versal then connects this data stream into the RL controller, while applying the output action to a stripline kicker. Every episode, comprised of 2048 interaction steps with accelerator data is sent back to the control room for training.

The actor network is implemented in the AIE. The employed PPO algorithm implementation uses a NN to select the mean value of a Gaussian distribution, from which the action applied to the environment is chosen. The standard deviation of the probability distribution is a trainable parameter, that is updated together with the NN coefficients.

A schematic of the internal data processing within the actor and the KINGFISHER platform is shown in figure 3. The first AIE tile implements a circular buffer and streams the latest eight samples to the following kernel using the cascade stream interface. These eight samples represent the observation vector, the choice of which will be described more thoroughly in section 5.2. The next kernel implements the linear layer of a NN computing the values of the sixteen hidden neurons. A rectified linear unit (ReLU) activation function is applied to the outputs. Such an activation function was selected because of its simple implementation. This function can also be turned off while the system is still running in order to implement linear filters. The output of the network is then computed with a final linear layer and passed to the last output kernel. All these kernels keep forwarding the eight input values. The last kernel takes 16 values from uniformly-distributed random number data stream and sums them, producing an approximately Gaussian-distributed sample due to the central limit theorem. This is used to add a Gaussian noise with a standard deviation selectable at run-time to the output of the network. The deviation of this approach from a Gaussian distribution was tested on simulation, and was proven to not affect training. A final data vector containing the eight input values, the random value, and the output of the network previous to the noise addition is given to the experience accumulator logic.

A computer in the control room then fetches the data and uses a modified version of the Stable-baselines3 library [45] PPO [20] implementation to train the policy using the hyper-parameters in table 1. The new parameters are then loaded onto the AIE kernel at run-time and new data for training

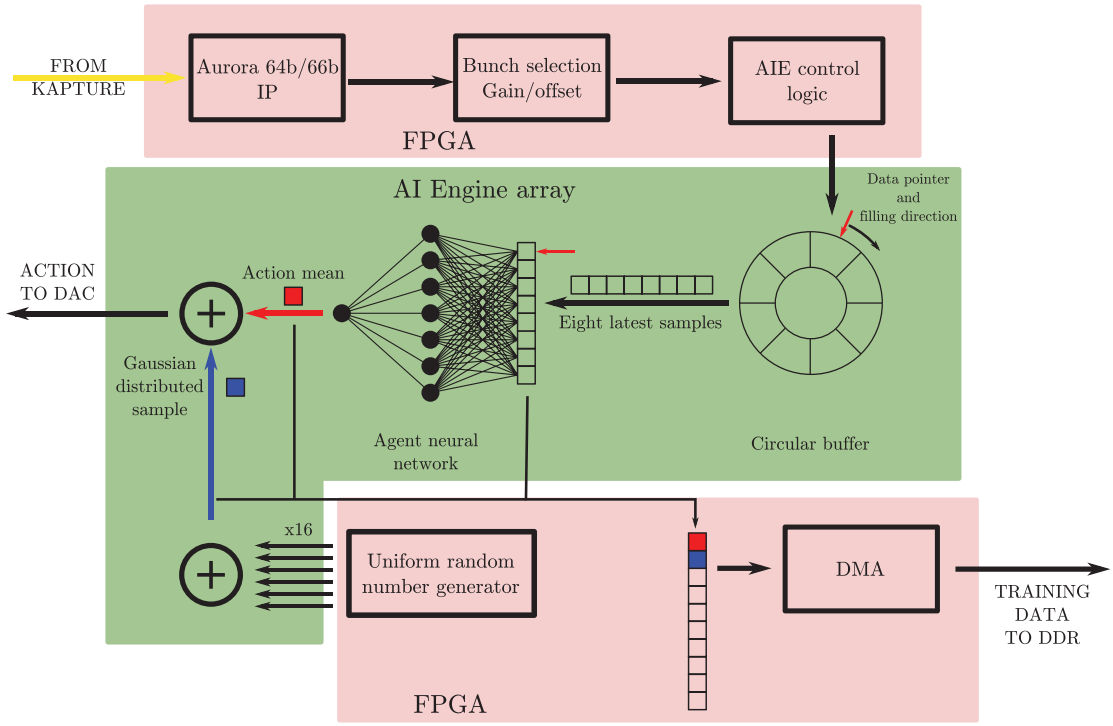

Figure 3. Schematic representation of the data path within the Versal VCK190 firmware. In the top part, the KAPTURE data stream coming from HighFlex 2 is decoded by an Aurora IP from the protocol used for fiber-optic communication. The bunch of interest for the control experiment is selected and gain/offset correction are applied. FPGA logic takes care of forwarding data and gracefully stopping the AIE. In the middle, a buffer stores the latest eight samples and feeds them to the NN. The exploration noise is added and the action is then output to a DAC, for control of the kicker. In the bottom side, a FPGA block takes the latest data from the inference and stores them in memory through a DMA for later training.

Table 1. Hyper-parameters used for the PPO algorithm.

Hyper-parameter	Value
Learning rate η	0.0012
Discount rate γ	0.99
Number of steps	2048
Batch size	64
Number of epochs	1

is gathered. The whole inference loop has a latency of 2.8 μs. For this work, a number of 2048 action steps was chosen. This number has been manually selected in such a way that the agent would not have enough time to cause beam loss and disrupt operation.

The critic network was chosen to be identical to the actor network. The remaining hyper-parameters are available in table 1.

5. Demonstration of real-time RL control on the accelerator

5.1. HBOs at KARA

The KARA is an electron synchrotron and light source with an injection energy of 0.5 GeV and with various operation modes between 0.5 GeV and 2.5 GeV. Its lattice is based on four sectors of double-bend achromats, with a total circumference of 110 m.

In synchrotrons, the horizontal beam dynamics is dominated at first order by the linear betatron motion. The bunch displacement from the reference orbit x can be expressed with Hill's equation

$$\frac{\mathrm{d}^2x}{\mathrm{d}s^2} + K_x(s)\,x = 0 \tag{5}$$

where s is the length along the ring and K_x are the periodic focusing functions.

The general solution to this equation is

$$x(s) = x_0\sqrt{\beta_x(s)}\cos\left[\theta(s) + \theta_0\right]$$
$$\theta(s) = \int_0^s \frac{\mathrm{d}s}{\beta_x(s)} \tag{6}$$

where β_x are the beta-functions and (x_0, θ_0) set the initial conditions. The number of full oscillations per revolution is defined as the horizontal tune Q_x. In injection mode, used for this experiment, $Q_x \approx 6.76$. An observer fixed in a given position of the accelerator will only observe the fractional part, corresponding to an oscillation frequency in the order of 700 kHz, depending both on the operation mode, the beam current, and more generally the state of the machine. This oscillation frequency sets the timescale at which the RL agent must be able to act, in this case in the order of a few microseconds.

A stripline kicker, based on the design from [46], is capable of affecting the HBOs by applying a bunch-by-bunch and turn-by-turn horizontal force to the beam based on the signal provided into an analog input.

The injection in the KARA storage ring makes use of three strong kicker and one septum magnet in order to properly merge the beam already in the machine with the one that is being injected. These kicker magnets can only activate at a rate of 1 Hz, for a few revolutions, and can move the beam on a displaced orbit in the horizontal plane. When the kicker switches back off, the displaced bunches will start performing betatron oscillations around the reference orbit. The goal of the RL agent is to damp this oscillation as quickly as possible. Notably, the strength of the injection kickers is orders of magnitude stronger than the stripline kickers used for feedback, thus an agent cannot damp an oscillation in a single kick, and turn-by-turn control is required. An acquisition trigger starts the RL agent interaction with the environment, followed by a second trigger after a constant delay for the injection kicker. In this way it is possible to test the agent with a predictable kick.

A classical controller for this problem exists and is already available in commercial solutions. These bunch by-bunch feedback systems [47] are usually based on a finite impulse response (FIR) filter that takes the input signal and applies a π rad phase at the frequency of the instability. In this way, a linear kick is produced with an opposite sign compared to the displacement, in this way damping the oscillations. The output of this kind of filter can be computed as

$$y(t) = \sum_{i=0}^{N} c_i x(t-i) \tag{7}$$

where c_i are the coefficients of the filter, and N is its order. The tuning of the filter coefficients directly impacts the performance of the controller, as it defines its behavior with respect to external noise and the bandwidth over which a suitable phase offset is produced. So far this is usually hand-tuned. A review on the topic is provided in [48].

5.2. Formulation as an RL task

For the current problem of controlling the HBO, the RL environment was modeled as follows. Given the HBO dynamics at a fixed position in the storage ring can be approximated by an harmonic oscillator, the position x and its derivative $\dot{x}$ are sufficient to have full knowledge of the state of the system. In a discrete-time setting, the derivative can be computed from the time difference of two consecutive samples,

$$\dot{x}(t) = \frac{x(t) - x(t-1)}{\Delta t}. \tag{8}$$

This in turn means that the two latest position samples, $x(t)$ and $x(t-1)$, are also a full representation of the system's state. In practice, though, only having two values is subject to measurement noise. Thus, the observation vector was defined as the last eight positions $\boldsymbol{o}_t = (x(t), x(t-1), \ldots, x(t-7))^{\mathrm{T}}$. The signal is sampled at the revolution frequency, i. e. ca. 2.7 MHz, thus eight samples span roughly two periods of the betatron oscillation.

The action is a force that is applied to the bunch through a stripline kicker. In the harmonic oscillator model, this corresponds to a driving force. Under this definition, the system is a MDP.

Several different reward definitions were chosen, and the respective performance of the final agents are compared in section 5.5. All rewards studied in this paper penalize the agent when the x position differs from zero, corresponding to the reference orbit.

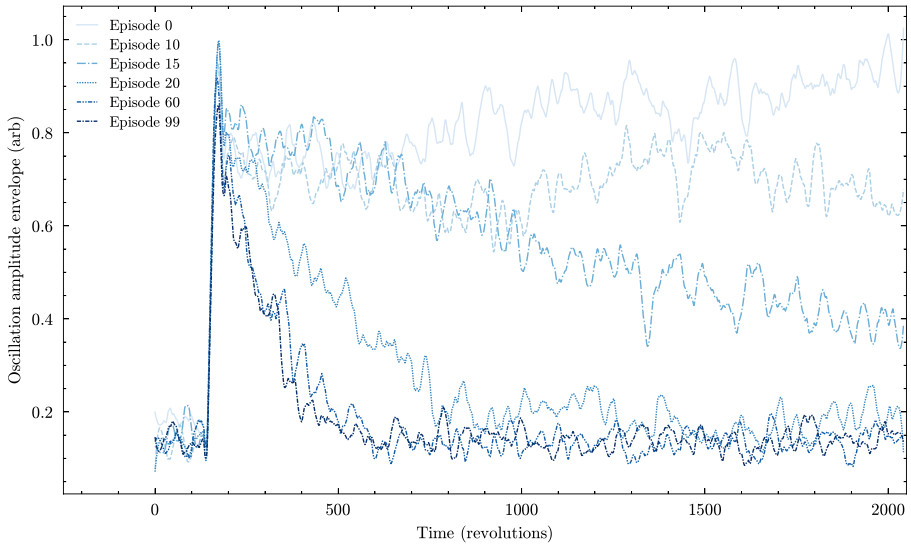

Figure 4. Smoothed envelope of oscillations measured by a BPM. The sharp increase around $t = 150$ revolutions is due to the injection kicker emulating an instantaneous external excitation. Notice how in episode 0 the randomly selected agent is destabilizing the beam, leading to an increase of the oscillation amplitude. Moreover, the rate of damping increases with the number of training steps, i.e. with the agent's experience.

5.3. Simulation study

In order to study the interaction of an agent with the HBO, an environment based on the Gymnasium library [49] was developed. The dynamics was modeled as a damped harmonic oscillator with user selectable undamped angular frequency ω_0 and damping ratio Γ. The environment stores the actions a_i performed on the system and convolves this vector with the Green's function $B(t, t')$ of the damped harmonic oscillator

$$B(t, t') = \Theta(t - t') \frac{e^{-\Gamma(t-t')}}{\omega} \sin \omega (t - t'), \tag{9}$$

with $\omega = \sqrt{\omega_0^2 - \Gamma^2}$ and $\Theta(t)$ is the Heaviside step function. As such, the position $x(t)$ is computed as

$$x(t) = \sum_i B(t, (i + \Delta\tau) T_{rev}) a_i, \tag{10}$$

where T_{rev} is the revolution time of the accelerator. An additional user selectable delay $\Delta\tau$ is added to the argument of the function to study the effect of latency. Gaussian noise is added to the samples, reflecting the behavior of real-life data. A kick of intensity one order of magnitude higher than what the agent can perform is applied at a random time to simulate the external kicker.

In order to guide the selection of an RL algorithm and agent structure, the environment was used to test the training performance with the algorithms available in the Stable-baselines3 library [45]. PPO and the observation vector definition of section 5.2 were thus validated in simulation before testing the complete system on the accelerator. This is necessary in order to disentangle a hardware platform failure from an issue with the RL problem formulation, in the case control could not be achieved during the tests at KARA.

5.4. RL control

The system discussed in this study was allowed to interact with the accelerator for 2048 revolutions (corresponding to 784 μs). During this period an external kicker excited the oscillations as shown in figure 4. After each of these episodes, a training step was performed, updating the coefficients of the NN. The new set of weights and biases were uploaded to the agent and the operation was repeated.

In order to study the evolution of the oscillation amplitude, the amplitude of the oscillation was obtained as the absolute value of the Hilbert transform of the raw oscillation signal $x(t)$. As shown in figure 4 an increase in the damping rate of the oscillations is an indication that the agent in question is achieving control of the environment. Moreover, it is possible to examine the trend of the cumulative reward obtained during each training episode as shown in figure 5. A clear increase in the obtained

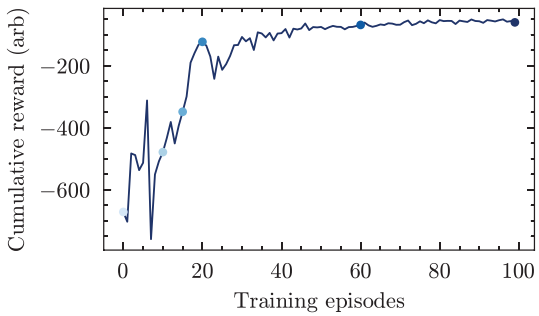

Figure 5. Cumulative reward obtained by an agent as a function of the number of training episodes. It can be seen how experience is gained (i.e. more episodes are used for training) and eventually plateaus. The colored points correspond to the episodes depicted in figure 4.

Table 2. Definition of the different reward functions used experimentally, where x denotes the transverse horizontal position of the beam from a reference, as read by the BPM.

Reward name	Definition
L1	$-\lvert x\rvert$
L2	$-x^2$
Tanhsq	$-\tanh(x^2)$

cumulative reward is visible as more episodes are used for training. This clearly shows that the agent improves with experience, as it is expected.

Several different training configurations were tested, each one with a different reward definition and the number of neurons in the hidden layer of the actor. This led to agents with different performances. An example of training configuration is L2, 12N, meaning the L2 norm defined in table 2 is used, together with an actor having 12 neurons in the hidden layer.

5.5. Training time reward definition

Figure 6 compares the performance of different reward functions employed during training. To do so, the oscillation amplitude was fitted with an exponential function

$$f(t;A,\lambda) = A\mathrm{e}^{-t\lambda} \tag{11}$$

and the damping rate λ was employed as a reward independent metric. Provided x is the position obtained from the BPM, the reward functions definitions are listed in table 2.

The agents trained with all of these three reward choices reached a final performance better than the FIR controller and the baseline with the untrained agents. For these tests sinusoidal coefficients were employed as is performed in the commercially available system [47]. These coefficients were manually tuned and thus are likely not the optimal ones.

5.6. Online network structure reconfiguration

In order to further increase the level of flexibility of the system, the possibility of dynamically modifying the NN structure without the need of re-implementing or re-packaging the firmware was implemented. This was achieved by overcommitting the amount of resources available to the NN, and then implementing a software layer that during configuration can pad weights and biases with zeroes or disable layers in order to achieve the desired architecture. This approach has the additional benefit of maintaining the number of computations constant, allowing to have identical latency between different training trials, thus removing this variability when comparing different agents.

Agents with several different layer sizes have been trained and their performance is shown in figure 6. The performance of all agents increased with training, outperforming the traditional FIR controller. It is worth noting how the FIR controller can in principle attain a similar level of performance with respect to the RL agent. However, for this to be achieved, extensive manual tuning is required, which is not necessary in the RL approach, as tuning is performed autonomously during the training

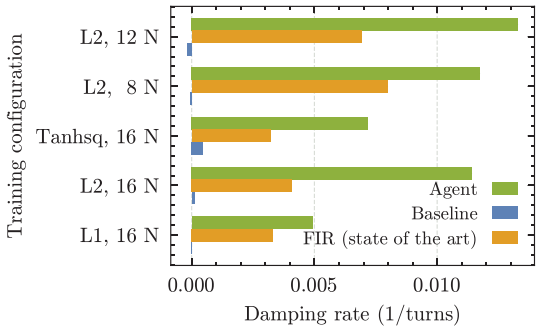

Figure 6. Damping rates of an instantaneous external excitation that is achieved for different kinds of training parameters. The *y*-axis denotes the reward function used for the specific training according to table 2 and the number of neurons in the hidden layer. The trained and untrained (baseline) RL agents are compared against an FIR controller. Note that the negative baseline values are caused by the agents with random coefficients actually exciting the instability. All agents outperform the state-of-the-art FIR controller showing higher damping rates.

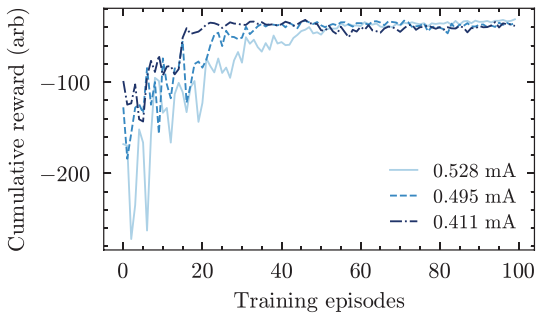

Figure 7. Comparison of different training episodes with different beam current conditions, where the final agents achieve almost identical performances.

procedure. The best performing agent, with 12 neurons in the hidden layer and trained with an L2-norm reward (in short notation: L2, 12N; cf also table 2), is used throughout this work for comparison with classical control techniques.

5.7. Training stability and robustness

The training procedure was repeated several times, with the same setting but a different beam current, to study the stability of the agents produced. An effect would be expected for two reasons. First of all the BPM signal is not normalized, so the amplitude will vary with current, as the produced signal is directly proportional to the bunch charge. Second of all the HBO tune is current dependent [50]. Despite a 20% reduction in beam current due to the natural decay of the beam, all resulting agents achieve a very similar final reward, as shown in figure 7. This shows the robustness of the RL agent against variations of current.

One of the main components of the KARA storage ring injection line is the injection septum magnet. Its impulse activation is necessary to guarantee the injection of the electron bunch coming from the booster into the main ring. The leaking magnetic field, though, also affects the beam that is already in the storage ring. This effect is visible in figure 8, which corresponds to a shift in the position of the beam. Such an effect was not present in the simulation, but it was nonetheless possible to train an agent capable of correctly handling this new phenomenon. This is an example of the versatility and adaptability of RL algorithms, that are sometimes able to autonomously learn from situations they are not originally designed for.

5.8. Improvement during cumulative reward plateau

As can be seen in figure 7, the cumulative reward reaches a plateau around step number 50. Nonetheless, if one studies the trend of the damping rate measured at a BPM in a different part of the ring, it is still possible to observe an increase of the damping rate even around step 100 as shown in figure 9. This is due to the fact that noise in the input data adds an offset to the cumulative reward that hides small improvements. Such a phenomenon needs to be considered in future experiments as it could potentially hinder further improvement of the agent.

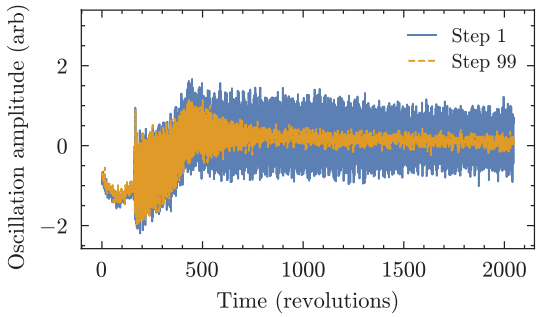

Figure 8. Evolution of the horizontal position of the beam after an instantaneous external excitation at time $t \approx 175$ revolutions, influenced by an RL agent before and after training. The septum magnet was active, inducing a baseline shift superimposed on the usual exponential decay. Nonetheless, the trained agent (orange) is capable of damping the oscillation compared to the untrained one (blue).

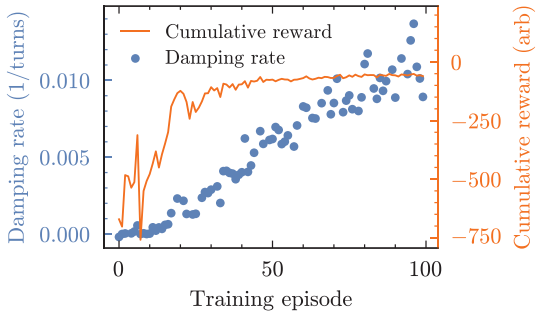

Figure 9. Damping rate (blue) measured with a different BPM system as a function of the training step, compared with the reward function (orange). Notice the absence of a plateau in the damping rate curve.

6. Discussion

We trained several working agents and their performance can be compared and evaluated.

As can be seen in figure 6, the performance of the controllers are not constant. In the case of the FIR filter, this behavior stems from its input signal not being normalized to the beam current. As such, variations in the beam current affect the amplitude of the position signal and, due to the FIR filter linear nature, in turn changes the action signal amplitude. This is also the case for the RL agent, but its retraining capability automatically adapts to the variation in beam current. This could in principle also be achieved with an FIR filter, e.g. by properly normalizing the horizontal BPM position signal, or by manually retuning the coefficient. Nonetheless, the trained agent outperformed the FIR controller in all of our evaluations. Additionally, the untrained RL agent is shown as the baseline, with clearly inferior performances when compared to the trained agent and the FIR controller.

Compared to the FIR controllers conventionally used, NN agents are capable of exhibiting non-linear output response. This allows the implementation of more complex policies. Particularly shallow NNs with ReLU activation functions have been shown to behave linearly in some cases [51], due to the piecewise linear nature of the output and the limited number of linear regions. When pure sinusoidal input is fed into a device under test, the non-linear behavior produces harmonics of the fundamental sinusoidal input. The total harmonic distortion plus noise (THD+N), is defined as

$$\mathrm{THD{+}N} = \frac{\sqrt{A_{\mathrm{N}}^2 + \sum_{i=2}^{\infty} A_i^2}}{A_1}, \tag{12}$$

where A_i is the amplitude of the ith harmonic, where $i = 1$ is the fundamental, and A_{N} is the noise amplitude. This metric expresses the amount of non-linear components in the output of a given device. For a linear controller, like a FIR filter, in the case where noise is negligible, THD+N is approximately zero. The THD+N was computed for different amplitudes of a sinusoidal input at the betatron oscillation frequency. The behavior for the L2, 12N agent is shown in figure 10. The amount of non-linear harmonic content is consistent, with a steep increase at a level compatible with the noise floor of the signal

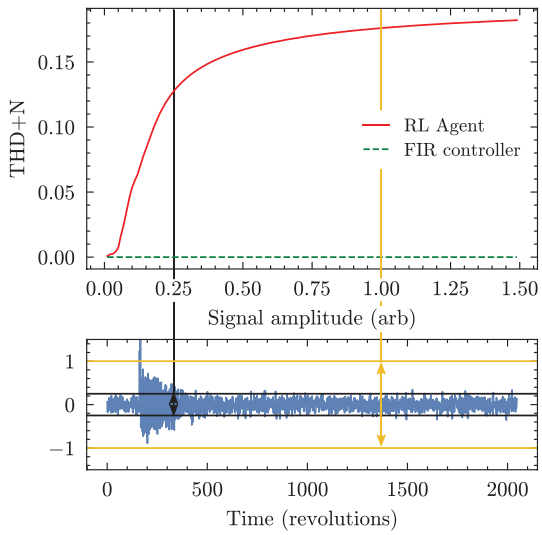

Figure 10. Total harmonic distortion plus noise (THD+N) as a function of the input signal amplitude for an (L2-norm, 12 neurons) agent. Note that the FIR filter with negligible noise will have a (THD+N) close to 0. In the bottom plot, a training signal is shown, allowing to determine which part of the training episode employs a higher non-linear behavior. The noise level is indicated with the black lines, while an higher amplitude is shown with the yellow lines. Signals above the noise level tend to produce more non-linear actions.

provided to the agent. This might indicate that the agent is learning to apply more complex actions in the case of high-amplitude, and thus highly penalized, observations.

The trade-off between training through interaction with a simulated or real-world environment is an important aspect of the application of RL to large-scale facilities. A simulation-driven approach is sometimes necessary to ensure safety, both of the facility and its personnel, while in other cases, it is dictated by the time necessary to obtain the training dataset. One advantage of real-world training is that trained agent can be readily transferred into operations without the additional burdens that come with training on simulated environments. Agents trained on simulation can be challenging to transfer to the real machine due to non-modeled phenomena and differences between the training and real-world environments in general. This work presents the opportunity to compare these two approaches. To do so, the time necessary to train an agent through interaction with the accelerator and simulation is reported in table 3. It is important to consider that the online case comprised waiting for a 1 Hz trigger, controlling the kickers. These numbers comprise the overhead of transferring training data, the time necessary for NN back-propagation and the interaction time with the environment. Given both approaches used the same model, the back-propagation time can be assumed to be equal. The interaction time, on the other hand, is 17.6 s for the simulation, while the trial on the accelerator takes 0.076 s or 76 ms. It is relevant to notice how the simulation being employed, and discussed in greater detail in the method section, is lightweight while still performing two orders of magnitudes slower than gathering data on the machine. As such, approximately 50% of the real-world training time is consumed by data-access overhead. This could be reduced in future implementations of the system with several approaches: using higher speed network links paired with lower overhead network protocols, or by accelerating the training directly on the FPGA and AIE array sharing memory with the experience accumulator hardware.

In this first proof of principle test, the experimental setup was designed to be intrinsically safe. The experiment was performed at currents where beam loss would not lead to damage to the machine. For future application, in-depth failure mode analysis is key to guarantee safety. There are currently two possible ways of attacking this problem. The first, the one implicitly used in this work, is to have external systems that handle the safety of the accelerator. The second, more complex one, is to employ the current developments in safe-RL to limit the actions of the agent. A combination of these would in principle lead the safest outcome.

In certain scenarios, RL agents may experience performance degradation when exposed to out-of-distribution data, such as observation, states, or rewards that have not been covered during training. Given the training is performed with interaction data produced by the environment that we are trying to control, out-of-distribution data occur only if the environment undergoes some form of drift. Once the policy is fixed after training, such drift can occur over time. To address this, in future works one

Table 3. Comparison of the time necessary to perform 100 training steps for different training platforms.

Training platform	Interaction time	Training time
Simulation CPU	17.6 s	137 s
Simulation GPU	17.6 s	227 s
Online CPU	0.076 s	260 s

could incorporate online performance monitoring and trigger retraining when it degrades, or investigate continual learning methods.

In conclusion, for systems with dynamics with an interaction rate that is much higher at the accelerator compared to a simulation, the techniques described in this article will greatly improve the time necessary for training. Furthermore, safety measures need to be in place to prevent damage to the accelerator or danger to people. All this considered, environment-driven training procedures, i.e. training directly on the real-world task, in some cases becomes not only possible, but also more flexible than a simulation study as the total deployment time would be reduced.

7. Outlook

The proof-of-principle system discussed is a promising option for future low-latency high-throughput deployments of RL algorithm in particle accelerators and other suitable facilities. Future works is needed to address some limitations of this novel workflow. First, the hardware infrastructure could be improved in order to work in the multi-bunch case, in this way serving as a natural testbed for multi-agent RL experiments. Second, care is needed when choosing the time when training is carried out and what data this should use. Thirdly, HBOs and other instabilities related to them have a complex behavior. Instabilities can appear and disappear, and the equilibrium behavior can be optimized to reduce noise induced by beam motion. This could in principle be addressed with further studies on the RL task formulation.

Lastly, the current hardware system still requires involvement from an edge computing expert in order to deploy new NN architectures, such as new layers and activation functions, on hardware. In future works, advances in NN compilers could be employed to reduce the dependency on hardware expertise.

8. Conclusions

In this study, we introduced the experience accumulator architecture and the training time reward definition, marking a significant step in implementing real-world on-the-edge experiment-based learning for RL-based controllers. Particularly, in scenarios where simulation costs are prohibitive and data generation rates are high, this methodology emerges as a promising solution, enabling the deployment of RL controllers. Our application of this approach to the real-time control of particle accelerator dynamics yielded an inference latency of a few microseconds.

Utilizing advanced heterogeneous computing platforms such as KINGFISHER, our presented implementation facilitates the deployment of plug-and-play RL systems operating at microsecond latency scales. The efficacy of this system was demonstrated through the control of the HBO at the accelerator test facility KARA, resulting in a functional controller comparable to state-of-the-art FIR controllers used during standard operation phases of synchrotron light sources. This development is fundamental as a technological demonstrator. For the problem examined in this work, an FIR controller is a reliable solution that will not be replaced by RL techniques. On the other hand, this opens the door to intelligent control of non-linear dynamics in this timescale [18] in systems such as particle accelerators and fusion experiments that are not achievable with conventional control techniques. This represents, to our knowledge, the first real-time microsecond latency deployment of RL algorithms running on hardware and learning directly from experimental data for particle accelerator control.

The workflow we described is highly customizable and can be tailored to a wide variety of applications. Furthermore, the framework allows users without specialized FPGA to carry out real-time RL

algorithmic exploration (value network, reward definition, etc). In this way, the expert's knowledge can be harnessed in this way, facilitating the transfer to operation of novel RL solutions.

Acknowledgments

The authors thank Lars Eisenblätter, Jennifer Derschang and Alexander Bacher for their assistance with the electronics used in this work. Luca Scomparin and Chenran Xu acknowledge the support by the DFG-funded Doctoral School 'Karlsruhe School of Elementary and Astroparticle Physics: Science and Technology'. Andrea Santamaria Garcia acknowledges funding by the BMBF ErUM-Pro Project TiMo (FKZ 05K19VKC).

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.

ORCID iDs

Luca Scomparin 0000-0002-1773-0769
Michele Caselle 0000-0003-3115-1170
Andrea Santamaria Garcia 0000-0002-7498-7640
Chenran Xu 0000-0002-5034-2207
Edmund Blomley 0000-0003-1480-2437
Timo Dritschler 0000-0002-6218-0743
Akira Mochihashi 0000-0002-1574-3144
Marcel Schuh 0000-0002-8633-8656
Johannes L Steinmann 0000-0003-1772-5828
Erik Bründermann 0000-0003-4119-3489
Andreas Kopmann 0000-0002-2362-3943
Jürgen Becker 0000-0002-5082-5487
Anke-Susanne Müller 0009-0000-1503-5669
Marc Weber 0000-0002-3639-2267

References

[1] Edelen A *et al* 2018 Opportunities in machine learning for particle accelerators misc (arXiv:1811.03172[physics.acc-ph])
[2] Roussel R *et al* 2024 Bayesian optimization algorithms for accelerator physics *Phys. Rev. Accel. Beams* **27** 084801
[3] Santamaria Garcia A, Bründermann E, Caselle M, Carne G D, Müller A-S, Scomparin L and Xu C 2024 How can machine learning help future light sources? *Proc. 67th ICFA Adv. Beam Dyn. Workshop Future Light Sources (FLS'23)* (*ICFA Advanced Beam Dynamics Workshop* (JACoW Publishing) pp 249–56
[4] Degrave J *et al* 2022 Magnetic control of tokamak plasmas through deep reinforcement learning *Nature* **602** 414
[5] Kaiser J *et al* 2024 Reinforcement learning-trained optimisers and bayesian optimisation for online particle accelerator tuning *Sci. Rep.* **14** 15733
[6] Madysa N, Velotti F, Biancacci N, Alemany-Fernández R, Kain V and Goddard B 2022 Automated intensity optimisation using reinforcement learning at LEIR *JACoW IPAC* **2022** 941
[7] Bruchon N, Fenu G, Gaio G, Lonza M, O'Shea F H, Pellegrino F A and Salvato E 2020 Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser *Electronics* **9** 781
[8] Kain V, Hirlander S, Goddard B, Velotti F M, Della Porta G Z, Bruchon N and Valentino G 2020 Sample-efficient reinforcement learning for cern accelerator control *Phys. Rev. Accel. Beams* **23** 124801
[9] O'Shea F, Bruchon N and Gaio G 2020 Policy gradient methods for free-electron laser and terahertz source optimization and stabilization at the fermi free-electron laser at Elettra *Phys. Rev. Accel. Beams* **23** 122802
[10] St. John J *et al* 2021 Real-time artificial intelligence for accelerator control: a study at the Fermilab booster *Phys. Rev. Accel. Beams* **24** 104601
[11] Santamaria Garcia A, Xu C, Kaiser J, Eichler A and Hirlaender S 2025 Reinforcement learning in particle accelerators *IPAC'25 - 16th Int. Particle Accelerator Conf.* (*Proc. IPAC'25*) (JACoW Publishing) pp 2481–6
[12] Shin K and Ramanathan P 1994 Real-time computing: a new discipline of computer science and engineering *Proc. IEEE* **82** 6
[13] Gaide B, Gaitonde D, Ravishankar C and Bauer T 2019 Xilinx adaptive compute acceleration platform: versal architecture *Proc. 2019 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays* (*FPGA'19*) (Association for Computing Machinery) pp 84–93

[14] Hirlaender S, Lamminger L, Zevi Della Porta G and Kain V 2023 Ultra fast reinforcement learning demonstrated at CERN AWAKE *JACoW* **IPAC2023** THL038
[15] Kaiser J *et al* 2023 Learning to do or learning while doing: reinforcement learning and bayesian optimisation for online continuous tuning (arXiv:2306.03739)
[16] Boltz T 2021 Micro-bunching control at electron storage rings with reinforcement learning *PhD Thesis* Karlsruher Institut für Technologie (KIT) (https://doi.org/10.5445/IR/1000140271)
[17] Scomparin L 2025 Real-time reinforcement learning with online training for large-scale facilities *PhD Thesis* Karlsruher Institut für Technologie (KIT) (https://doi.org/10.5445/IR/1000180745)
[18] Luca S *et al* 2024 Preliminary results on the reinforcement learning-based control of the microbunching instability *Proc. 15th Int. Particle Accelerator Conf.* (*IPAC'24 - 15th Int. Particle Accelerator Conf.* (JACoW Publishing) pp 1808–11
[19] Mnih V *et al* 2015 Human-level control through deep reinforcement learning *Nature* **518** 529
[20] Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O 2017 Proximal policy optimization algorithms (arXiv:1707.06347[cs.LG])
[21] Haarnoja T, Zhou A, Abbeel P and Levine S 2018 Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor *35th Int. Conf. on Machine Learning, ICML 2018* vol 5 p 2976 (arXiv:1801.01290v2)
[22] Fujimoto S, Van Hoof H and Meger D 2018 Addressing function approximation error in actor-critic methods
[23] Rothmann M and Porrmann M 2022 A survey of domain-specific architectures for reinforcement learning *IEEE Access* **10** 13753
[24] Baranwal A R, Ullah S, Sahoo S S and Kumar A 2021 ReLAccS: a multilevel approach to accelerator design for reinforcement learning on FPGA-based systems *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **40** 1754
[25] Sahoo S S, Baranwal A R, Ullah S and Kumar A 2021 Memorel: a memory-oriented optimization approach to reinforcement learning on FPGA-based embedded systems *Proc. 2021 on Great Lakes Symp. on VLSI* (*GLSVLSI'21*) (Association for Computing Machinery) pp 339–46
[26] Da Silva L M D, Torquato M F and Fernandes M A C 2019 Parallel implementation of reinforcement learning Q-learning technique for FPGA *IEEE Access* **7** 2782
[27] Meng Y, Kuppannagari S, Rajat R, Srivastava A, Kannan R and Prasanna V 2020 QTAccel: a generic FPGA based design for Q-table based reinforcement learning accelerators *2020 IEEE Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)* pp 107–14
[28] Spanò S, Cardarilli G C, Di Nunzio L, Fazzolari R, Giardino D, Matta M, Nannarelli A and Re M 2019 An efficient hardware implementation of reinforcement learning: the Q-learning algorithm *IEEE Access* **7** 186340
[29] Shao S, Tsai J, Mysior M, Luk W, Chau T, Warren A and Jeppesen B 2018 Towards hardware accelerated reinforcement learning for application-specific robotic control *2018 IEEE 29th Int. Conf. on Application-Specific Systems, Architectures and Processors (ASAP)* pp 1–8
[30] Hu C-W, Hu J and Khatri S P 2022 Td3lite: FPGA acceleration of reinforcement learning with structural and representation optimizations *2022 32nd Int. Conf. on Field-Programmable Logic and Applications (FPL)* pp 79–85
[31] Cho H, Oh P, Park J, Jung W and Lee J 2019 FA3c *Proc. 24th Int. Conf. on Architectural Support for Programming Languages and Operating Systems* (ACM) (https://doi.org/10.1145/3297858.3304058)
[32] Su J, Liu J, Thomas D B and Cheung P Y 2017 Neural network based reinforcement learning acceleration on FPGA platforms *ACM SIGARCH Comput. Archit. News* **44** 68
[33] Watanabe H, Tsukada M and Matsutani H 2021 An FPGA-based on-device reinforcement learning approach using online sequential learning *2021 IEEE Int. Parallel and Distributed Processing Symp. Workshops (IPDPSW)* pp 96–103
[34] Li M-J, Li A-H, Huang Y-J and Chu S-I 2019 Implementation of deep reinforcement learning *Proc. 2nd Int. Conf. on Information Science and Systems* (*ICISS'19*) (Association for Computing Machinery) pp 232–6
[35] Meng Y, Kuppannagari S and Prasanna V 2020 Accelerating proximal policy optimization on CPU-FPGA heterogeneous platforms *2020 IEEE 28th Annual Int. Symp. on Field-Programmable Custom Computing Machines (FCCM)* pp 19–27
[36] Book G, Traue A, Balakrishna P, Brosch A, Schenke M, Hanke S, Kirchgässner W and Wallscheid O 2021 Transferring online reinforcement learning for electric motor control from simulation to real-world experiments *IEEE Open J. Power Electron.* **2** 187
[37] Scomparin L *et al* 2022 KINGFISHER: a framework for fast machine learning inference for autonomous accelerator systems *JACoW* **IBIC2022** 151
[38] Xilinx VCK190 evaluation board (available at: www.xilinx.com/products/boards-and-kits/vck190.html/)
[39] Aurora 64B/66B protocol specification (SP011) (available at: https://docs.amd.com/v/u/en-US/aurora_64b66b_protocol_spec_sp011)
[40] Experimental physics and industrial control system (EPICS) (available at: https://epics.anl.gov/)
[41] O'Neill M E 2014 PCG: a family of simple fast space-efficient statistically good algorithms for random number generation *Technical Report* Harvey Mudd College
[42] Caselle M 2017 KAPTURE-2. a picosecond sampling system for individual THz pulses with high repetition rate *J. Instrum.* **12** C01040
[43] Caselle M, Perez L A, Balzer M, Dritschler T, Kopmann A, Mohr H, Rota L, Vogelgesang M and Weber M 2017 A high-speed DAQ framework for future high-level trigger and event building clusters *J. Instrum.* **12** C03015
[44] Dimtel BPMH-20-2G hybrid (available at: www.dimtel.com/products/bpmh)
[45] Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M and Dormann N 2021 Stable-baselines3: reliable reinforcement learning implementations *J. Mach. Learn. Res.* **22** 1
[46] Dehler M, Schlott V, Bulfone D, Lonza M and Ursic R 2000 Current status of the ELETTRA/SLS transverse multibunch feedback *Proc. EPAC* pp 1894–6
[47] Dimtel iGp12 bunch-by-bunch feedback system (available at: www.dimtel.com/products/igp12)
[48] Lonza M and Schmickler P B H 2017 Multi-bunch feedback systems *CERN Yellow Reports: School Proc. Vol 3 (2017): roceedings of the CAS* (CERN)
[49] Towers M *et al* 2023 Gymnasium (available at: https://doi.org/10.5281/zenodo.8127026)
[50] Blomley E 2021 Investigation and control of beam instabilities at the Karlsruhe research accelerator using a 3-D digital bunch-by-bunch feedback system (https://doi.org/10.5445/IR/1000137621) *PhD Thesis* Karlsruher Institut für Technologie (KIT)
[51] Montúfar G, Pascanu R, Cho K and Bengio Y 2014 On the number of linear regions of deep neural networks *Proc. 28th Int. Conf. on Neural Information Processing Systems-Volume 2* (*NIPS'14*) (MIT Press) pp 2924–32