Accelerated Deep Reinforcement Learning for Fast Feedback of Beam Dynamics at KARA

Weijia Wang[1], Michele Caselle[1], Tobias Boltz[2], Edmund Blomley[2], Miram Brosi[3], Timo Dritschler[1], Andreas Ebersoldt[1], Andreas Kopmann[1], Andrea Santamaria Garcia[3], Patrick Schreiber[3], Erik Bründermann[2], Marc Weber[4], Anke-Susanne Müller[2], Yangwang Fang[5]
[1]Institute for Data Processing and Electronics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2]Institute for Beam Physics and Technology, Karlsruhe Institute of Technology, Karlsruhe, Germany
[3]Laboratory for Applications of Synchrotron Radiation, Karlsruhe Institute of Technology, Karlsruhe, Germany [4]Division V - Physics and Mathematics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[5]Unmanned System Research Institute, Northwestern Polytechnical University, Xi'an, China

Coherent synchrotron radiation (CSR) is generated when the electron bunch length is in the order of the magnitude of the wavelength of the emitted radiation. The self-interaction of short electron bunches with their own electromagnetic fields changes the longitudinal beam dynamics significantly. Above a certain current threshold the micro-bunching instability develops, characterized by the appearance of distinguishable substructures in the longitudinal phase space of the bunch. To stabilize the CSR emission, a real-time feedback control loop based on Reinforcement Learning (RL) is proposed. Informed by the available THz diagnostics, the feedback is designed to act on the RF system of the storage ring in order to mitigate the micro-bunching dynamics. To satisfy low-latency requirements given by the longitudinal beam dynamics, the RL controller has been implemented on hardware (FPGA). In this paper, a real-time feedback loop architecture and its performance is presented and compared with a software implementation using Keras-RL on CPU/GPU. The results obtained with the CSR simulation Inovesa demonstrate that the functionality of both platforms is equivalent. The training performance of the hardware implementation is similar to software solution, while it outperforms the Keras-RL implementation by an order of magnitude. The presented RL hardware controller is considered as an essential platform for the development of intelligent CSR control systems.

I. Micro-Bunching Instability

Coherent synchrotron radiation (CSR) in the terahertz (THz) frequency range is produced by electrons when the length of the bunches is in the range of a few millimetres and below. At the Karlsruhe Research Accelerator (KARA), the research electron storage ring and light source at KIT, the low momentum compaction (α_c) mode is used to produce short electron bunches and to study THz emission [1]. As shown in Fig (1), during the standard operation mode with long electron bunches, incoherent synchrotron radiation is emitted in the X-ray frequency range. In low α_c mode, high intensity CSR in the THz range is emitted.

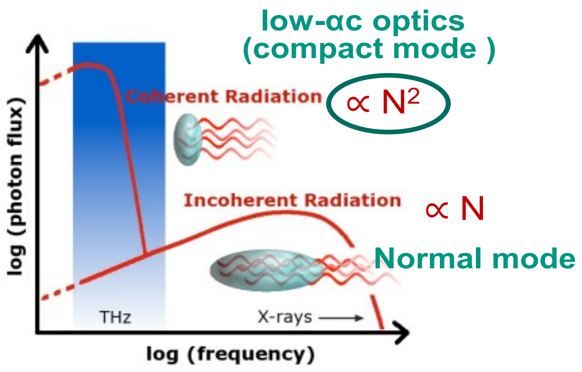

Fig. 1: In low α_c mode the power is proportional to the square of the number of particles; in standard operational mode the power is proportional to the number of particles.

Recent years have seen an increased interest in higher-power THz sources that open up a new range of research fields. Applications including semiconductor and high-temperature superconductor characterization, tomographic imaging, label-free genetic analysis, cellular level imaging, and chemical and biological sensing have transported THz research from relative obscurity into the limelight [2]. Dynamically changing substructures that appear during the micro-bunching instability translate into CSR power fluctuations [3] and limit the use of the emitted THz light in experiments that operate with the average intensity and work on short time scales. The goal is to improve the stability of the CSR emission and therefore improve the reproducibility for time resolved THz experiments in the future. In this paper, a novel system composed of a THz detector and high-throughput readout electronics with embedded real-time data processing based on AI algorithms is presented. It has been demonstrated that by dynamically acting on the radio-frequency (RF) cavities of the accelerator the fluctuations of

M. Caselle is with the Institute of Data Processing and Electronics, Karlsruhe Institute of Technology (KIT), Germany, e-mail: michele.caselle@kit.edu

the CSR emission can be reduced and the stability of the emission can be improved. In order to successfully control the CSR power fluctuations a fast feedback system with measurement of the CSR signal, intelligent controller, and actuator with a total latency in the order of microseconds is needed. A hardware platform for this purpose has been developed that will serve to optimize the beam properties at KARA.

The self-interaction in short electron bunches can be mathematically described by the CSR-induced wake potential [4]:

$$V_{\mathrm{CSR}}(q) = \int_{-\infty}^{\infty} \widetilde{\rho}(\omega) Z_{\mathrm{CSR}}(\omega) e^{i\omega q} \mathrm{d}\omega. \qquad (1)$$

The CSR wake potential is added as a perturbation to the Hamiltonian. Here, $q = (z - z_s)/\sigma_{z,0}$ denotes the generalized longitudinal position, $\widetilde{\rho}(\omega)$ represent the Fourier-transformed longitudinal bunch profile, and $Z_{\mathrm{CSR}}(\omega)$ is the CSR-induced impedance of the storage ring. The additional potential in Eq. (1) can be interpreted as a perturbation to the accelerating RF potential, and thus results in a perturbation of the synchrotron motion within the bunch. It causes the formation of micro-structures with a dynamic evolution at time scales in the order of the synchrotron period T_s. The effect of the instability on the charge density in the longitudinal phase space and the CSR power is shown in Fig (2). The CSR self-interaction of the bunch causes the formation of micro-structures in the longitudinal phase space density shown in Fig (2a). Their continuous variation leads to fluctuations in the emitted CSR power as a function of time in Fig (2b). Real-time control of the longitudinal beam dynamics during the occurrence of the micro-bunching instabilities can stabilize the fluctuations of the emitted CSR and therefore optimize and tailor the THz light for different applications [4].

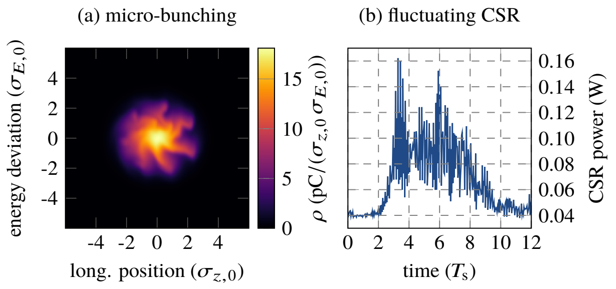

Fig. 2: CSR micro-bunching structures in the longitudinal phase space density ρ (a) and the resulting CSR power variations over time (b), where T_s denotes the synchrotron period ($\sim$ 100 µs).

II. Controller Design with Reinforcement Learning

In order to stabilize the CSR micro-bunching instability a machine learning approach based on Reinforcement Learning (RL) is selected. The goal of RL is to find the optimal sequence of decisions to maximize the expected reward by interacting with the controlled system. The interactions with a process make RL different to unsupervised and supervised learning. Unsupervised learning is typically used to detect intrinsic structures in a collection of unlabeled data while supervised learning aims to learn a mapping from input to output given as a collection of labeled examples. In contrast, RL algorithms are capable to learn purely from interaction with a real environment. As no classical controller design for the CSR problem is currently available, RL seems to be a promising approach. In the RL algorithm design there are two major components as shown in Fig (3), namely the agent and the environment. The state s is calculated from the measured information that can be obtained from the process. The state s is not necessarily equivalent to the complete description of the dynamics of the controlled system. The agent has a policy that maps the perceived state of the environment to an appropriate action. The RL agent interacts with its environment in discrete time steps. At each time step t, the agent receives the current state s_t and the reward r_t. It then chooses an action a, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the transition (s_t, a, s_{t+1}) is determined. The goal of an RL agent is to learn a policy which maximizes the expected cumulative reward. Several RL algorithms are available and currently under study, like Policy Gradient, Proximal Policy Optimization (PPO) [5] and Trust Region Policy Optimization (TRPO) [6]. For the development of a hardware loop in this article the DDPG (Deep Deterministic Policy Gradient) [7] algorithm has been employed.

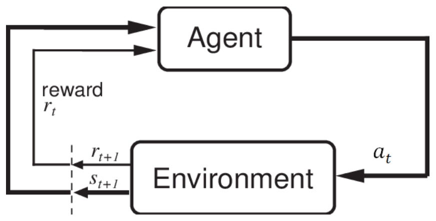

Fig. 3: Agent-process interaction in a RL control loop [8].

III. Hardware Implementation

Preliminary studies demonstrate that the dynamic modulation of the RF amplitude seems to be a particularly suitable and effective choice to counteract the CSR-induced perturbation. The influence of RF modulations on the micro-bunching dynamics has also been tested experimentally, for example in [9, 10]. An adaptive RF modulation scheme is a promising proposition to exert influence on the longitudinal beam dynamics of the micro-bunching instability as it provides the required flexibility to respond to the varying perturbation by the CSR wake potential. In order to build a control loop and to stabilize the CSR emission with high intensity and low fluctuation, a fast detector for beam diagnostics, and an RL hardware platform are required. These

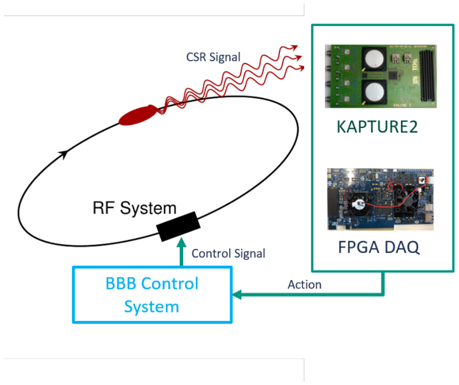

Fig. 4: Hardware implementation of the fast feedback loop at KARA.

two components close the control loop as shown in Fig (4). A commercial zero-biased Schottky barrier diode detector is employed to measure the THz radiation. A quasi-optical version with a silicon lense and a log-spiral antenna from ACST GmbH is used due to its broad spectral sensitivity from 50 GHz up to 2 THz [11]. The alternatives are a wave-guide coupled Schottky barrier diode detectors from Virginia Diodes, Inc. where the spectral sensitivity depends on the chosen wave-guide and can range from WR15 (50-75 GHz) to WR0.65 (1100-1700 GHz) [12]. This detector is connected to the fast digitizer board KAPTURE- 2. The fast electrical signal is digitized and transferred to the RL platform HighFlex2. The incoming data is processed in real-time and the feedback signal is sent to the bunch-by-bunch control system which controls the RF kicker cavity of the accelerator.

A. *Front End Electronics for CSR bunch-by-bunch measurements*

For bunch-by-bunch measurements the ultrafast digitizer KAPTURE-2 is employed. The KAPTURE-2 (Karlsruhe Pulse Taking Ultra-fast Readout Electronics) [13] is a broad-band picosecond sampling system for ultra-short pulses operating at a pulse rate of up to 1 $GPulse/s$. KAPTURE-2 is able to acquire and sample ultra-short pulses with a local sampling rate that exceeds 300 GS/s. Up to 8 parallel sampling channels are integrated in one system which are used to digitize each pulse with 8 sampling points. The data rate of the KAPTURE-2 is $8 \times 1\,\mathrm{GS/s} \times 12\ bit$ per sample point.

B. *High-throughput DAQ board*

To face the upcoming demand of high data throughput and fast machine learning data processing close to the data source, the readout board HighFlex2 shown in Fig (6) based on PCIe generation 4 has been developed. The main processor on HighFlex2 is a Xilinx ZYNQ UltraScale+. It includes a 64-bit quad-core ARM processor with clock rates of up to 1.5 GHz. This embedded processor is employed to implement the RL framework. The ZYNQ is equipped with an FPGA with up to 600 k Configurable Logic Blocks (CLB), several tens of megabytes of block RAM and UltraRAM and more than 2900 DSP slices [14]. This heterogeneous hardware is a promising platform to implement machine learning algorithms.

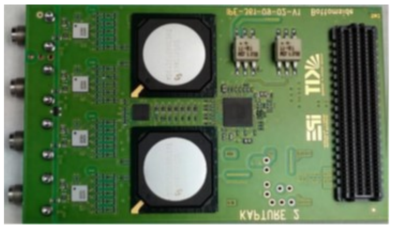

Fig. 5: KAPTURE-2 front-end card with 4 channels, several cards can be combined together to increase the number of the sampling channels.

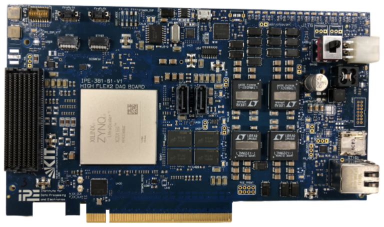

Fig. 6: High-throughput DAQ board HighFlex2.

KAPTURE-2 and its readout card have been designed with a modular approach which allows to scale-up the number of channels. The precise clock distribution implemented in KAPTURE-2 guarantees the synchronous operation of multiple devices with a precision of 25 ps. This modularity can be used to scale-up the number of THz detectors or to increase the number of sampling points per pulse.

C. *Reinforcement Learning Hardware Implementation*

The RL algorithms are difficult to implement in programmable logic but a variety of toolkits are available for CPU and GPU. To ensure fast processing and low latencies we consider two solutions for the implementation of RL. The first is to implement the RL using the heterogeneous FPGA-GPU infrastructure developed in [15] and the second method employs the embedded ARM processors available on HighFlex2. In the first architecture, the FPGA card is connected to an external processing node where the machine learning platform is installed. This infrastructure has been designed to provide a scalable solution for future generations of real-time data processing in photon sciences and high energy physics in terms of data throughput and processing performance. To meet real-time constraints, the data is directly transferred from the FPGA into the GPU memories, bypassing CPU memory system by high-performance ad-hoc direct memory access (DMA). In this case, standard frameworks like for example Tensorflow [16], Keras [17], or Caffe [18] can be used. In the second method, a lite-weight RL framework is implemented on the ARM processor of the ZYNQ device. This approach is less scalable, but offers the opportunity to realize the complete algorithms on-chip without the need to transfer data to external processing units. A RL software platform is already available on ARM and tested with some textbook examples with the Policy Gradient (PG) [19] method and could also successfully be employed to implement the fast RL controller of KARA. This controller uses an algorithm according to the Deep Deterministic Policy Gradient, which is an extension of the work in [19]. The framework is written in C/C++ and is executed as bare-metal code without operating system overhead on the ARM processor. The second solution has been selected in this paper in order to achieve the best performance of the bunch control system.

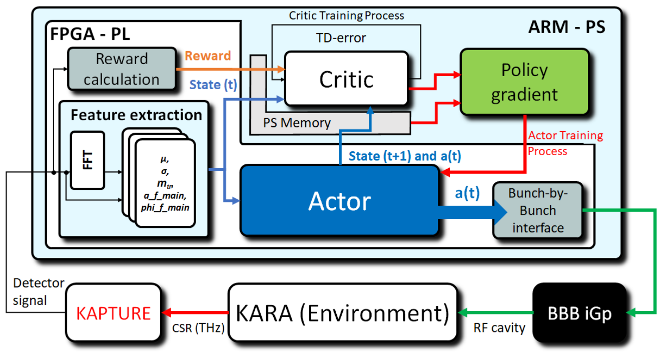

Fig. 7: Hardware solution for RL control.

To optimize the performance of the RL controller, the hardware implementation on HighFlex2 has been divided into two parts: the Actor neural network inference, located in the FPGA, and the training policy with the Critic network, located in the ARM processors. The RL hardware implementation with the external interfaces to KARA are shown in Fig (7). The THz pulse sampled by KAPTURE-2 is processed in the FPGA. The first processing step is the extraction of eight features that define the state of the beam. These include the mean and standard deviation of the CSR power, as well as other additional parameters derived from the CSR signal. To maintain a high flexibility concerning both the modifications of the definition and calculation of the features, the code is written in the high-level language OpenCL/C++ and synthesized for execution on an FPGA by Vivado-HLS (high Level synthesis) from Xilinx. At the same time, also the logic for the calculation of the reward function has been written in OpenCL/C++. Because the aim is to stabilize the emitted CSR, a possible choice for the reward function is:

$$R_t \doteq \omega_1 \mu_{t':t} - \omega_2 \sigma_{t':t} \ , \tag{2}$$

where $\mu_{t':t}$ and $\sigma_{t':t}$ denote the normalized mean and standard deviation of the time series $P_{t,\mathrm{CSR}}$ in the interval $[t', t]$, and $\omega_{1,2} > 0$ are simple weighting factors. This definition of the reward reflects the goal of having a CSR power signal of high intensity and a low fluctuation, and corresponds to a smooth charge distribution that is not significantly changing in time. Eq. (2) has proven up to now to be a reasonable choice. The extracted features build the state vector s_t with eight signals $(s_t^{(1)}, \ldots, s_t^{(8)})$, which contains the beam physics information that is processed by the Actor network. This network consists of a fully-connected dense layer neural network on FPGA, optimized to provide an output with very low latency. The output of the Actor is defined within the so-called "action space". As the additional CSR wake potential acts as a perturbation to the RF potential, one promising approach seems to be centred around the RF system. Therefore, the action space is defined as a two dimensional vector, constituting an RF amplitude modulation

$$a = (A_{\mathrm{mod}}, f_{\mathrm{mod}}). \tag{3}$$

The action is mapped to the bunch-by-bunch system interface, which will set a modulation signal to the RF driver according to

$$V_{\mathrm{RF}}(t) = \hat{V}(t) \sin(2\pi f_{\mathrm{RF}} t + \varphi_{\mathrm{RF}}), \tag{4}$$

where $V_{\mathrm{RF}}(t)$ is the RF signal applied to the RF cavity and $\hat{V}(t)$ is the RF amplitude modulation

$$\hat{V}(t) = \hat{V}_0 A_{\mathrm{mod}} \sin(2\pi f_{\mathrm{mod}} t + \phi_{\mathrm{mod}}). \tag{5}$$

Here, A_{mod} and f_{mod} denote the amplitude and frequency of the applied RF amplitude modulation, which are dynamically adjusted by the Actor network. The phase offset of the modulation ϕ_{mod} is not changed by the feedback and set to zero in the simulation. Dynamically modifying these two parameters should provide the agent with a substantial amount of control over the RF system. The global RF phase φ_{RF} remains untouched in this study. Both parameters, the amplitude A_{mod} and the frequency f_{mod}, are then sent to a commercial signal processor for the bunch-by-bunch feedback system (model dimtel iGp12-720F [20]). The HighFlex2 card is connected to the bunch-by-bunch feedback system by a custom parallel interface. Based on preliminary studies, the dynamic modulation of the RF amplitude seems to be a particularly suitable approach to effectively counteract the CSR-induced perturbation. The influence of RF modulations on the micro-bunching dynamics has also been tested experimentally [9, 10].

The behavior of the Actor is fundamental for the training phase. Therefore, a memory buffer has been instantiated at the ARM processor to store the complete status of the actor. In particular, the action a, the state s_t, the next state s_{t+1}, and the reward calculated by the FPGA r_t are stored.

The Critic network consists of a fully-connected dense layer network inference implemented on the ARM processor. It accesses the Actor status via the interface memory and generates the so called "action-value function" $Q(s_t, a)$. This function represents an estimate of the reward the agent will be able to receive after taking action a in state s_t.

As shown in Fig (7) the training process of the Actor is performed by the policy gradient. The Actor training path and its target will be shown in more detail in Fig (9).

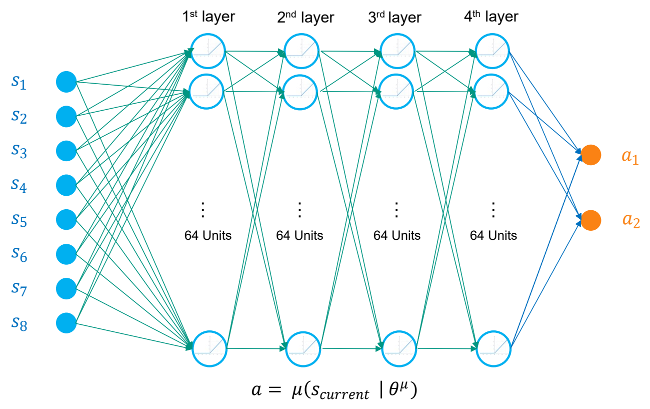
1st layer
2nd layer
3rd layer
4th layer
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8
64 Units
a_1 a_2
$a = \mu(s_{current} \mid \theta^{\mu})$

Fig. 8: The DDPG Actor architecture.

D. *Reinforcement Learning architecture and its training process*

The Actor network consists of four fully-connected dense layers, each layer consists of 64 neurons using a rectified linear unit (ReLU) as activation function, the most common type in neural networks. It receives the 8-dimensional state vector $s_t = (s_t^{(1)}, \dots, s_t^{(8)})$ extracted from the recent CSR power signal $P_{t,\mathrm{CSR}}$ and generates the action vector a with the signals $(a^{(1)}, a^{(2)})$ defined in Eq. 3 to be propagated further to the bunch-by-bunch interface. The neural network architecture is shown in Fig (8).

The training method of the Actor is based on the policy gradient algorithm. As shown in Fig (9), the training process receives the state s_t from the buffer memory. The Actor network converts s_t in the action a. The action a and state s_t are propagated to the Critic network to evaluate the action-value function $Q(s_t, a)$.

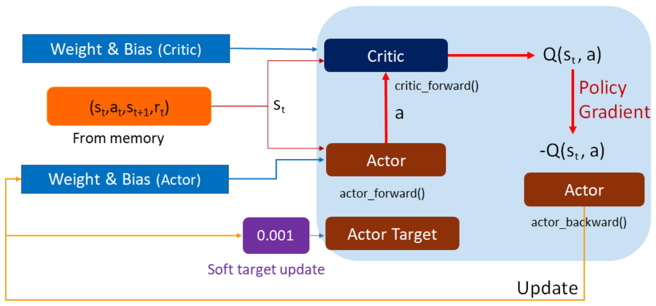

Fig. 9: Training process of the Actor network and the actor target network. The update rate of the Actor target network is reduced by soft target update coefficient.

The hardware implementation of the policy gradient is developed in order to accelerate the processing of the action-value function $Q(s_t, a)$. In the DDPG algorithm the loss function is defined as the sign-inverted action-value function, therefore the loss-functions is: $-Q(s_t, a)$ [19]. Because the training process will calculate all weights and biases of the neural network shown in Fig (8) by minimization of the the loss function $-Q(s_t, a)$, the training of the network is also reached by an increased quantity $Q(s_t, a)$. Then the new value of both, weights and biases are updated by the backward propagation through the Actor network.

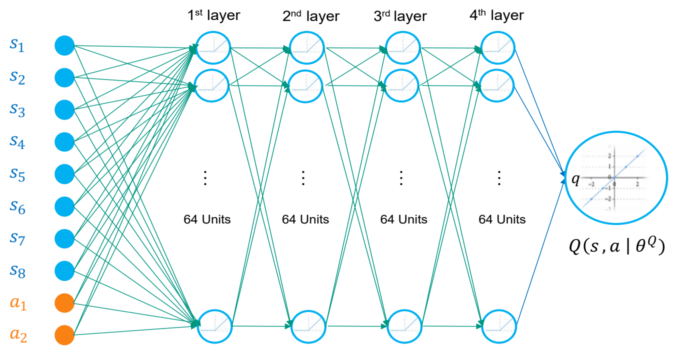

Fig. 10: The DDPG Critic network architecture.

In the DDPG algorithm additional target networks for Actor and Critic are used to improve the stability of the optimization and their output is denoted with a' and Q'. The Actor target network is implemented in the ARM processor. It represents a copy of the Actor network with both weights and biases updated with a rate reduced by a factor, the so called "soft target update" coefficient. In the current implementation the "soft target update" is set to 0.001. The final step of the Actor training will be the update of the Actor target with a reduced update rate, as shown in Fig (9).

The Critic network consists of four fully-connected dense layers, where each layer consists of 64 neurons with a ReLU activation function. The Critic network in DDPG receives as input the state vector s_t and the action a and generates the output $Q(s_t, a)$ as shown in Fig (10). An overview of the training mechanism of both the Critic and Critic target networks is shown in Fig (11). It is implemented in the ARM processors and is based on the Temporal-Difference (TD) method.

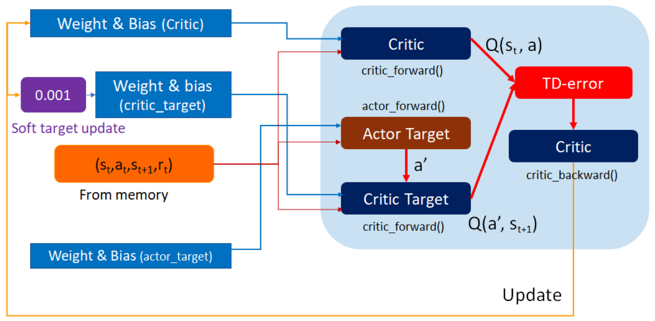

Fig. 11: Training process of the Critic network and the Critic target. The update rate of the Critic target network is reduced by soft target update coefficient.

The Critic network reads the states s_t from the buffer memory and a from the FPGA Actor and generates the output function $Q(s_t, a)$. At the same time, also the target Critic reads the state s_{t+1} and a' action generated by the Actor target network and generates Q'. The TD-error is then calculated by the difference between $Q(s_t, a)$ and the so-called target value $r_t + \gamma(Q')$. Here γ is the discount factor, which is set to 0.9 [7]. This difference is used to evaluate the loss function of the Critic network. Finally, the new values of the weights and biases are updated by backward propagation of the Critic network. Similar to the Actor training, a further Critic target network is also implemented on the ARM processor. It represents a copy of the Critic network, where weights and biases are updated with a rate reduced by the "soft target update" coefficient.

IV. Simulation Setup

The simplified simulation setup shown in Fig (12) is used to develop, study, and measure the performance of the RL controller. To improve the flexibility during development, the Actor and Critic DDPG inferences and training networks are implemented as bare-metal applications on the ARM processors of the HighFlex2 readout card. The bunch structure and the CSR behaviour are simulated by the beam physics simulator Inovesa, solving the Vlasov-Fokker-Planck (VFP) equation [21]. Inovesa and the calculation of features and reward function are executed on a PC, which is connected to HighFlex2 by an Ethernet link as shown in Fig (12).

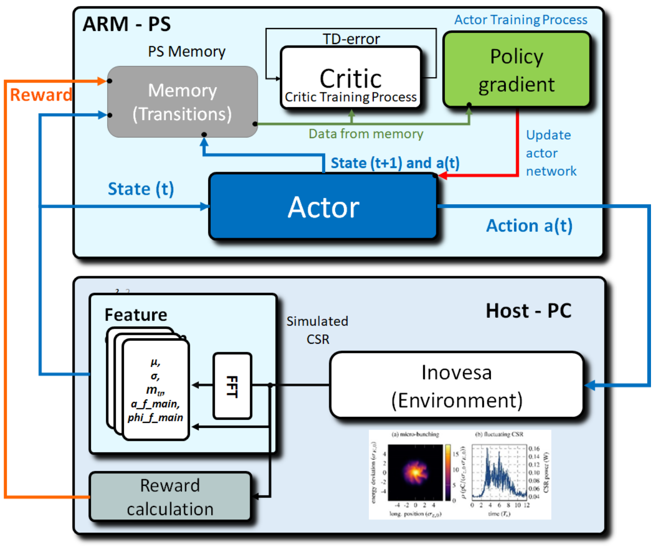

Fig. 12: Configuration of the simulation setup with HighFlex2 and Inovesa CSR simulation. Compared to the targeted implementation shown in Fig (7), feature extraction, and calculation of the reward function are shifted to the PC, while both the Actor and Critic are implemented at the ARM processor.

The HighFlex2 board acts as a server and responds to the demands of the environment (Inovesa Beam Dynamics). The Actor network is responsible of generating the proper action a, corresponding to an amplitude and frequency modulation of the RF signal, which is applied to the Inovesa simulation. Because the Actor network will be implemented as fast inference engine on FPGA, the floating-point precision of the neural network has been reduced from 32bit (FTP32) to 8 bit fixed-point values (INT8). This quantization increases significantly the performance of the neural networks, while preserving sufficient accuracy [22]. The comparison of the hardware controller with reduced precision and the software implementation with full precision demonstrates the validity of this approach for the control of the CSR micro-bunching. This can be seen in Fig. 13, further described in the next section.

V. Results

In this section the performance of the RL controller implemented as shown in Fig (12) is presented. The results of hardware (FPGA), CPU, and GPU implementations are compared. The measurements are intended to evaluate several aspects of the proposed RL implementation: the learning behaviour and its comparison with a Keras-RL implementation executed on GPU, the latency during the training phase, and the latency of the trained Actor.

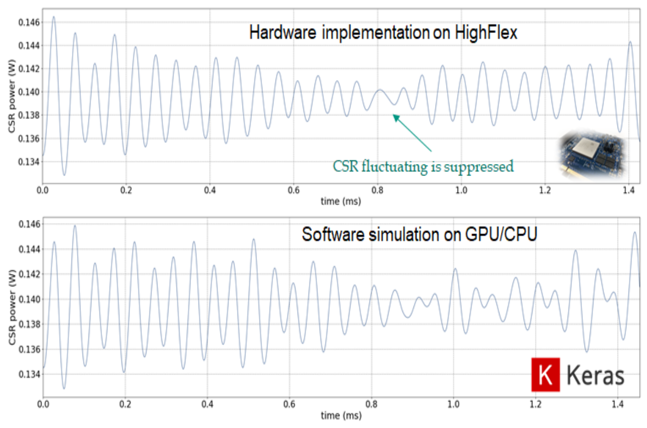

Fig. 13: Comparison of the CSR power stabilization achieved with the hardware implementation on FPGA and the software RL controller.

The RL training is characterized using a defined starting point and a well-defined terminal state. The complete sequence, from the starting state to the end state, is called an episode. One episode is a sequence that contains all states, actions, and rewards, ending with the terminal state. Fig (13) shows the CSR behaviour of a single episode during the training phase. For each time step, a new value of the amplitude modulation is generated by the Actor inference and is applied to the RF cavity. As a result of the amplitude modulation, the time behaviour of the fast CSR dynamics changes as shown in the plots of Fig (13). As the RL controller starts to interact with the environment and to learn from it, the CSR fluctuation starts decreasing and the rewards from the environment increase. The episode is terminated when the fluctuation of the CSR reaches a certain programmable threshold. The threshold value and its effect on the learning phase is defined according to the minimum reward that the RL agent reaches. More than 3500 episodes have been simulated with Inovesa where for each episode the RL agent learns how to interact with the accelerator.

Further studies are required to reach long-term stability of the demonstrated mitigation. This preliminary result shows that the DDPG algorithm and its hardware implementation are able to learn from the simulation environment. The comparison of the hardware implementation and Keras on GPU in Fig (13) shows a similar behaviour. The RL framework developed on FPGA, that should approximate the behaviour of the software implementation, works as expected. Fig (14) shows the comparison between the amplitude modulation generated by the Actor on HighFlex2 and on GPU.

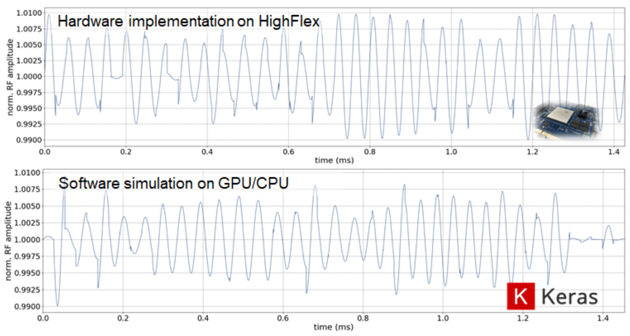

Fig. 14: Comparison of the RF amplitude generated by the RL controllers implemented on FPGA and in software.

The performance of both the training latency and the Actor latency of the proposed RL have been compared with the Keras-RL implemented on CPU and GPU. Each training process has been repeated 50 times and the average of the latency is calculated. Table (I) shows the resulting processing times for DDPG for the KARA application. The GPU has the worst performance, with a mean latency of 6037 µs, and a large standard deviation (STD) of 55 µs. The results of CPU and ZYNQ differ in mean and STD. The ZYNQ has the lowest training latency with only 1648 µs and the CPU has about 1800 µs. Although the average one-batch training time is similar at CPU and ZYNQ, the ZYNQ values are fluctuating less than the CPU values. The ZYNQ has a STD of only 0.1343 µs while the CPU varies by $\sim$ 16 µs STD.

TABLE I: Comparison of training and inference performance of ZYNC, CPU and GPU.

Property	ZYNC	CPU	GPU
Training performance			
mean	1648 µs	1800 µs	6037 µs
STD	0.1343 µs	16 µs	55 µs
Inference performance			
mean	16.932 µs	200 µs	557 µs
STD	0 µs	3.52 µs	19 µs

The GPU inference time takes about 557 µs with 19 µs STD and the CPU inference time is 200 µs with 3.52 µs STD. The FPGA inference time is measured by Vivado HLS, with a fixed latency 4233 clock cycle at 4 ns clock period. Thus, the ZYNQ inference is expected to be 16.932 µs with no STD.

VI. Conclusion

Driven by the self-interaction of electrons at high currents in short bunch mode, the micro-bunching instability in storage rings results in a fast and dynamic perturbation of the longitudinal charge distribution, leading to fluctuations of the emitted CSR. The goal is to stabilize the CSR emitted in the THz range by controlling the longitudinal phase space dynamics of the beam. To establish extensive control over the dynamics, a fast feedback loop based on RL processing is proposed and developed, which reacts to small changes in the charge distribution and adjusts the parameters of the RF system accordingly. In this paper, two major steps towards a full control of the micro-bunching instability are presented: the overall closed-loop hardware design for the KARA experimental setup and the performance evaluation using the physics simulation Inovesa. To validate the concept, the full simulation of the physics of the beam, including the fast dynamics of the THz emissions, has been set up using Inovesa. It interacts with the HighFlex2 (RL agent) to mimic the micro-bunching instability. The RL ZYNQ implementation on HighFlex2 has been compared with the CPU/GPU inference based on Keras-RL. Both RL implementations show a similar behaviour and same trend of reward collection. The result confirms that the developed RL framework to deploy a RL on ZYNQ devices is working as expected. Furthermore, the comparison shows a drastic reduction of the training time compared to the Keras-RL on CPU/GPU. Moreover, a very-low and fixed latency is guaranteed when the Actor inference is deployed on the FPGA, which ensures a reliable feedback for physics experiments. Given the generality of the approach and the immanent capability to learn from interaction with an environment, the implementation reported in this paper could potentially be employed to control the fast beam dynamics at other accelerator facilities.

References

[1] A.-S. Müller, "Accelerator-based sources of infrared and terahertz radiation," *Reviews of Accelerator Science and Technology*, vol. 03, no. 01, pp. 165–183, 2010. [Online]. Available: https://doi.org/10.1142/S1793626810000427

[2] M. S. Sherwin, C. A. Schmuttenmaer, and P. H. Bucksbaum, "DOE/NSF/NIH Workshop on Opportunities in THz Science," Washington, DC (United States), 2004, Available: https://science.osti.gov/-/media/bes/pdf/reports/files/DOE-NSF-NIH_Workshop_on_Opportunities_in_THz-Science_rpt.pdf.

[3] M. Brosi, J. L. Steinmann, E. Blomley, E. Bründermann, M. Caselle, N. Hiller, B. Kehrer, Y.-L. Mathis, M. J. Nasse, L. Rota *et al.*, "Fast mapping of terahertz bursting thresholds and characteristics at synchrotron light sources," *Physical Review Accelerators and Beams*, vol. 19, no. 11, p. 110701, 2016.

[4] T. Boltz, T. Asfour, M. Brosi, E. Bründermann, B. Härer, P. Kaiser, A.-S. Müller, C. Pohl, P. Schreiber, M. Yan *et al.*, "Feedback design for control of the micro-bunching instability based on reinforcement learning," in *10th Int. Partile Accelerator Conf.(IPAC'19), Melbourne, Australia, 19-24 May 2019*. JACOW Publishing, Geneva, Switzerland, 2019, pp. 104–107.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[6] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[8] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” 2011.

[9] Y. Shoji and T. Takahashi, “Coherent synchrotron radiation burst from electron storage ring under external rf modulation,” in *Conf. Proc.*, vol. 806233, no. EPAC08-MOPC048, 2008, p. MOPC048.

[10] J. L. Steinmann, *Diagnostics of Short Electron Bunches with THz Detectors in Particle Accelerators*. KIT Scientific Publishing, 2019.

[11] Advanced Compound Semiconductor Technologies (ACST) GmbH. [Online]. Available: http://www.acst.de/

[12] Virginia Diodes, Inc. [Online]. Available: http://vadiodes.com/

[13] M. Caselle, L. A. Perez, M. Balzer, A. Kopmann, L. Rota, M. Weber, M. Brosi, J. Steinmann, E. Bründermann, and A.-S. Müller, “Kapture-2. a picosecond sampling system for individual thz pulses with high repetition rate,” *Journal of Instrumentation*, vol. 12, no. 01, p. C01040, 2017.

[14] Xilinx, “Zynq ultrascale+ mpsoc product tables and product selection guide,” 2016.

[15] M. Caselle, L. A. Perez, M. Balzer, T. Dritschler, A. Kopmann, H. Mohr, L. Rota, M. Vogelgesang, and M. Weber, “A high-speed daq framework for future high-level trigger and event building clusters,” *Journal of Instrumentation*, vol. 12, no. 03, p. C03015, 2017.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[17] F. Chollet *et al.*, “Keras,” https://github.com/fchollet/keras, 2015.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM ’14. New York, NY, USA: ACM, 2014, pp. 675–678. [Online]. Available: http://doi.acm.org/10.1145/2647868.2654889

[19] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[20] D. Teytelman, “igp12-720f,” https://www.dimtel.com/_media/support/manuals/manual12_720f_print.pdf, 2019.

[21] P. Schönfeldt, M. Brosi, M. Schwarz, J. L. Steinmann, and A.-S. Müller, “Parallelized vlasov-fokker-planck solver for desktop personal computers,” *Physical Review Accelerators and Beams*, vol. 20, no. 3, p. 030704, 2017.

[22] Y. Fu, E. Wu, A. Sirasao, S. Attia, K. Khan, and R. Wittig, “Deep learning with int8 optimization on xilinx devices,” *White Paper*, 2016.