ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL029

BEAM TRAJECTORY CONTROL WITH LATTICE-AGNOSTIC REINFORCEMENT LEARNING

C. Xu*, E. Bründermann, A.-S. Müller, A. Santamaria Garcia
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
J. Kaiser, A. Eichler
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

In recent work, it has been shown that reinforcement learning (RL) is capable of outperforming existing methods on accelerator tuning tasks. However, RL algorithms are difficult and time-consuming to train, and currently need to be retrained for every single task. This makes fast deployment in operation difficult and hinders collaborative efforts in this research area. At the same time, modern accelerators often reuse certain structures, such as transport lines consisting of several magnets, within or across facilities, leading to similar tuning tasks. In this contribution, we use different methods, such as domain randomization, to allow an agent trained in simulation to easily be deployed to a group of similar tasks. Preliminary results show that this training method is transferable and allows the RL agent to control the beam trajectory at similar lattice sections of two different real linear accelerators. We expect that future work in this direction will enable faster deployment of learning-based tuning routines, and lead towards the ultimate goal of autonomous operation of accelerator systems and transfer of RL methods to most accelerators.

INTRODUCTION

Recently, reinforcement learning (RL) has been successfully applied at particle accelerators for various tasks [1–6] and could perform faster and better than human operators or numerical optimizers. In each step of the RL problem, the *agent* receives some *observation* of the environment, takes an *action*, and receives a *reward* based on its behavior. The RL agent gathers experience during training and improves its policy of choosing an action that maximizes the cumulative reward. Training RL agents, however, is a time-consuming process and demands a large number of samples. Furthermore, a trained RL agent is often specialized in the task and environment that it is trained on, requiring retraining when it is applied to other accelerators or even minor changes in the hardware condition. This presents a challenge for applying RL to new tuning tasks and transferring existing RL controllers to other facilities.

To mitigate this issue, we looked into different aspects of RL including the task formulation and common interfaces, aiming to produce transferable RL controllers for different accelerator facilities [7]. Previous studies show that using techniques like domain randomization [8, 9] for the magnet misalignment and initial upstream beam condition allows the agent trained only in simulation to be applied directly at the real accelerator [10]. In this contribution, we go one step further and randomize the accelerator lattices during the training phase. We show that the resulting RL agent could perform the tuning task for multiple similar lattices. Furthermore, we show that the agent's performance can be improved after fine-tuning a small number of steps on the test lattice.

TRANSVERSE BEAM TUNING

We consider a standard transverse beam-tuning task at linear accelerators using three quadrupole magnets as a triplet, and one pair of vertical and horizontal dipole corrector magnets. The goal is to steer and focus the beam on a downstream diagnostic screen by controlling the magnet settings. The screen image is fitted with a Gaussian distributed beam using four parameters $b = \{\mu_x, \sigma_x, \mu_y, \sigma_y\}$, namely the horizontal and vertical beam position and size. In each step, the RL agent receives a 13-dimensional observation, including the parameters of the current $b^{(\text{current})}$ and target beam $b^{(\text{target})}$, the strengths of the quadrupole magnets, and the bending angles of the correctors $u = \{k_{\text{Q1}}, k_{\text{Q2}}, k_{\text{Q3}}, \theta_{\text{v}}, \theta_{\text{h}}\}$. The agent changes the accelerator setting by applying a delta action $a = \Delta u$ on the magnets, where the step size is limited to 10% of the whole range of u. We used the averaged L1-norm, i.e. the mean absolute error (MAE), as a metric of the difference between the current and target beam

$$d(b^{(\text{current})}, b^{(\text{target})}) = \frac{1}{4}\sum_{i=1}^{4}\left|b_i^{(\text{current})} - b_i^{(\text{target})}\right|. \quad (1)$$

The reward is defined as the negative MAE normalized with respect to the largest observable beam difference $d_{\max}$ limited by the screen size

$$r = -d/d_{\max}. \quad (2)$$

In this way, the agent receives a higher reward for improving the beam parameters and is penalized for each step when the beam is far from the target beam.

For the RL training, we use a simulated lattice shown at the top (Training) in Fig. 1. Two sections of the real accelerator lattices are used for evaluation, namely the diagnostic section of FLUTE at KIT [11] and the experimental area of ARES at DESY [12]. As training model-free RL algorithms requires up to millions of samples, it is extremely slow or impossible to use existing codes like OCELOT [13] or ASTRA [14], where each simulation takes seconds or minutes to run. To

* chenran.xu@kit.edu

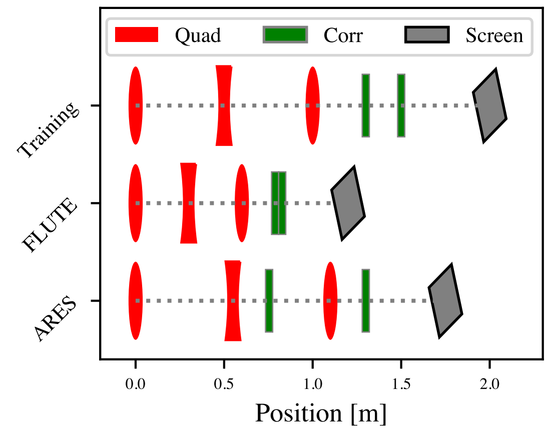

Figure 1: Lattice sections with quadrupoles (Quad) and correctors (Corr) studied for the transverse tuning task, including a dummy training lattice, the diagnostic section of FLUTE, and the experimental area at ARES.

facilitate fast training, we used the tensorized simulation code *Cheetah* [15] with transfer-matrices-based tracking, which trades off the simulation granularity for computational speed. We trained the RL agents using the benchmarked implementation of the soft actor-critic (SAC) algorithm in the *Stable-Baselines3* [16] package. Experiment tracking and Bayes hyperparameter tuning are performed with the *Weights and Biases* [17] package. The hyperparameters used in training are listed in Table 1.

Table 1: Hyperparameters for the RL Training with SAC

Hyperparameter	**Value**
Discount factor γ	0.99
Learning rate α	0.0003
Replay buffer size	10 000
Batch size	100
Gradient steps	1
Timesteps	500 000
Max episode length	50

One example of a beam tuning episode using the trained RL agent in the training lattice is shown in Fig. 2. The task is to produce a focused beam centered at the screen with random initial magnet strengths in a focusing-defocusing-focusing setting. The RL agent changed the magnet settings smoothly and successfully converged to the target beam within only 10 steps, which is significantly faster than manual tuning or numerical optimizers [10]. Additionally, it is interesting to see that the RL agent turns off the first quadrupole $Q1$ and only uses the other two as a doublet for the beam focusing.

To further investigate the learned RL policy, we calculated the correlation between the actions taken by one RL agent and the observations with an off-center incoming beam and 10 000 random magnet settings. Without loss of generality,

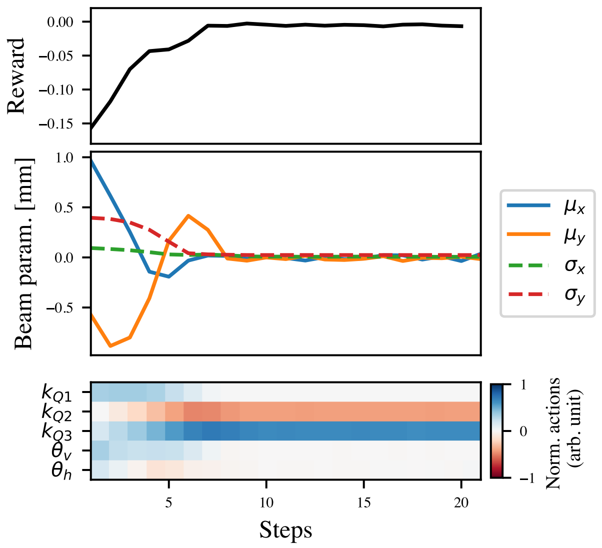

Figure 2: Example of one RL transverse tuning episode. The evolution of the beam parameters is shown in the middle, and the magnet settings are color-coded to their normalized values.

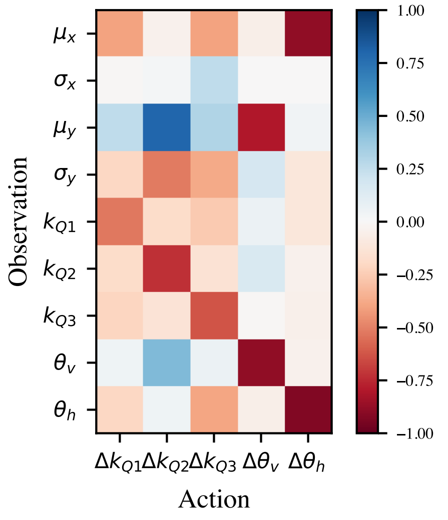

Figure 3: Correlation analysis of the RL predicted actions with respect to the observations.

we consider again the task to produce a perfectly focused and centered beam. For example, it can be seen in Fig. 3 that the agent successfully learns to use the corrector magnets $\{\Delta\theta_v, \Delta\theta_h\}$ to transversely deflect the beam as expected. Furthermore, the correlation between the corrector strength, beam positions, and changes in quadrupole strength indicates that the agent learned to use quadrupoles not only for focusing but also for compensating for the beam positions.

To increase the robustness and make the trained RL agent transferable for other accelerators, we included the domain randomization (DR) [8, 9] technique in the training phase. Instead of letting the agent repeatedly interact with the same lattice geometry, in each training episode, we randomly sample the distances between the five magnets and the screen to create a new training lattice. This approach exposes the agent to a wide range of system dynamics, forcing it to learn a more robust policy.

We evaluated the performance of the trained RL agents and show the results in Table 2, where *vanilla-SAC* denotes the agent trained on the training lattice mentioned before and *SAC-DR* denotes the agent trained with domain randomization technique. It needs to be mentioned that the agents trained with other algorithms like proximal policy optimization (PPO) achieved a similar performance, but they require many more interaction steps with the environment, taking about 5 million steps to train. As we also foresee training on real accelerators in the future, the more sample-efficient SAC algorithm is preferred. For each lattice configuration, the RL agents are started with 100 different initial magnet settings and allowed to tune the beam up to 50 steps. The metric is chosen to be the mean value of the best obtained beam MAEs, as defined in Eq. (1), over the 100 trials, which mostly corresponds to the final beam parameter in the case of RL agents. We can see that all the trained RL agents outperform the random actions by an order of magnitude. Although the vanilla-SAC agent successfully performs the task on the lattice it has been trained on, its performance drops significantly on the FLUTE lattice. On the other hand, the agent trained with DR performs clearly better on the FLUTE setting and reached similar results for both cases. Nevertheless, the performance is slightly worse than the specialized agent trained with the same number of steps. One possible reason is that the task becomes naturally more complicated when the magnet positions are changing and it is difficult to train a policy that performs well on all the possible lattices.

In addition to the Cheetah simulation, we also tested a more realistic FLUTE model including the space charge effect using the OCELOT simulation. As expected, the agent trained on a simple simulation model could capture the system dynamics and directly be applied to a higher-fidelity simulation model without any degradation in performance. In fact, a previous study showed that the RL agent trained in simulation can also be applied to the real machine without retraining [10]. We also applied the trained agent to the experimental area at the ARES linac and the results are shown in the 4th column in Table 2. One particular feature of the ARES lattice is that the vertical corrector is positioned between the two quadrupoles $\{Q2, Q3\}$, which places it *out-of-distribution* (OOD) with respect to the DR training. Nevertheless, the DR agent is still able to tune the beam and reach the same performance as other lattice configurations. This clearly indicates that the agent is generalizing to unknown scenarios.

Lastly, we investigated the possibility of further improving the DR agents by *fine-tuning* (FT) on the test lattice, shown in the 4th row in Table 2. In this case, the agent is retrained for another 10 000 steps, corresponding to only 2 % of the original number of samples, on the lattice it is tested on respectively. The fine-tuned agent clearly improved for all

Table 2: Performance of the trained RL agents on the simulated training (Tr), FLUTE (FL), and ARES (AR) lattices. The lattices are tested using either the transfer-matrices-based simulation Cheetah (Ch) or OCELOT (Oc) including the space-charge effect. The beam MAEs are averaged over 100 tasks with random initial magnet settings.

Algorithm	**Averaged best beam MAE** [μm]			
	Tr-Ch	**FL-Ch**	**FL-Oc**	**AR-Ch**
Random	958	621	548	930
Vanilla-SAC	43	154	150	92
SAC-DR	86	63	60	67
SAC-DR+FT	52	31	31	36

the lattices and reached a comparable or better performance as the original agent trained only on one lattice. This indicates that the DR agents can be used as a starting point and retrained on the real accelerators within a reasonable amount of time, allowing a successful agent to be deployed at different facilities.

CONCLUSION

We trained RL agents to perform the transverse beam tuning task at linear accelerators using quadrupole and corrector magnets. Our simulation results show that the RL agents trained in low-fidelity simulations can be successfully applied to simulations including more complex dynamics like collective effects. In addition, aided by techniques such as domain randomization, RL agents trained for specific tuning tasks can be transferred between different particle accelerators with similar characteristics. It can even be generalized for out-of-distribution lattices to some extent, which makes it applicable to new lattice configurations, which were not considered in the training phase. The DR agent can be fine-tuned further on the real accelerator with a small number of steps. This approach could achieve comparable performance to that of training the RL agent completely on a specific lattice, which significantly reduces the required beam time. It is expected that the adaptation steps on the accelerators can be further reduced using, for example, meta-reinforcement learning and model-based approaches.

ACKNOWLEDGMENTS

This work is partly funded by the Initiative and Networking Fund by the Helmholtz Association (Autonomous Accelerator, ZT-I-PF-5-6) and the IVF project InternLabs-0011 (HIR3X). C. Xu acknowledges the support by the DFG-funded Doctoral School "Karlsruhe School of Elementary and Astroparticle Physics: Science and Technology". In addition, the authors acknowledge support from KIT (Karlsruhe, Germany) and DESY (Hamburg, Germany), members of the Helmholtz Association HGF.

REFERENCES

[1] F. H. O'Shea, N. Bruchon, and G. Gaio, "Policy gradient methods for free-electron laser and terahertz source opti-

mization and stabilization at the FERMI free-electron laser at Elettra," *Phys. Rev. Accel. Beams*, vol. 23, p. 122 802, 2020. doi:10.1103/PhysRevAccelBeams.23.122802

[2] S. Hirlaender and N. Bruchon, "Model-free and Bayesian Ensembling Model-based Deep Reinforcement Learning for Particle Accelerator Control Demonstrated on the FERMI FEL," *arXiv*, 2020. doi:10.48550/arXiv.2012.09737

[3] V. Kain *et al.*, "Sample-efficient reinforcement learning for cern accelerator control," *Phys. Rev. Accel. Beams*, vol. 23, p. 124 801, 2020. doi:10.1103/PhysRevAccelBeams.23.124801

[4] J. St. John *et al.*, "Real-time artificial intelligence for accelerator control: A study at the Fermilab Booster," *Phys. Rev. Accel. Beams*, vol. 24, p. 104 601, 2021. doi:10.1103/PhysRevAccelBeams.24.104601

[5] T. Boltz, "Micro-Bunching Control at Electron Storage Rings with Reinforcement Learning," Ph.D. dissertation, 2021.

[6] N. Bruchon *et al.*, "Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser," *Electronics 2020, Vol. 9, Page 781*, vol. 9, p. 781, 2020. doi:10.3390/ELECTRONICS9050781

[7] A. Eichler *et al.*, "First Steps Toward an Autonomous Accelerator, a Common Project Between DESY and KIT," in *Proc. IPAC'21*, Campinas, Brazil, May 2021, pp. 2182–2185. doi:10.18429/JACoW-IPAC2021-TUPAB298

[8] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30. doi:10.1109/IROS.2017.8202133

[9] OpenAI *et al.*, "Solving Rubik's cube with a robot hand," *arXiv*, 2019. doi:10.48550/arXiv.1910.07113

[10] J. Kaiser, O. Stein, and A. Eichler, "Learning-based Optimisation of Particle Accelerators Under Partial Observability Without Real-World Training," in *Proc. of the 39th International Conference on Machine Learning*, vol. 162, 2022, pp. 10 575–10 585. https://proceedings.mlr.press/v162/kaiser22a.html

[11] M. J. Nasse *et al.*, "FLUTE: A versatile linac-based THz source," *Review of Scientific Instruments*, vol. 84, no. 2, p. 022 705, 2013. doi:10.1063/1.4790431

[12] F. Burkart *et al.*, "The ARES Linac at DESY," in *Proc. 31st International Linear Accelerator Conference (LINAC'22)*, Liverpool, UK, 2022, paper THPOJO01, pp. 691–694. doi:10.18429/JACoW-LINAC2022-THPOJO01

[13] I. Agapov, G. Geloni, S. Tomin, and I. Zagorodnov, "Ocelot: A software framework for synchrotron light source and fel studies," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 768, pp. 151–156, 2014. doi:10.1016/j.nima.2014.09.057

[14] K. Flöttemann, *ASTRA: A Space Charge Tracking Algorithm*. https://www.desy.de/~mpyflo/

[15] O. Stein, I. V. Agapov, A. Eichler, and J. Kaiser, "Accelerating Linear Beam Dynamics Simulations for Machine Learning Applications," in *Proc. IPAC'22*, Bangkok, Thailand, 2022, pp. 2330–2333. doi:10.18429/JACoW-IPAC2022-WEPOMS036

[16] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021. http://jmlr.org/papers/v22/20-1364.html

[17] L. Biewald, *Experiment Tracking with Weights and Biases*, Software available from wandb.com, 2020. https://www.wandb.com/