ISBN: 978-3-95450-240-0 ISSN: 2673-5547 doi:10.18429/JACoW-HIAT2022-TH1I2

REINFORCEMENT LEARNING AND BAYESIAN OPTIMIZATION FOR ION LINAC OPERATIONS*

J. Martinez-Marin, B. Mustapha, B. Blomberg, E. Letcher, G. Dunn, D. Stanton, and K. Bunnell
Physics Division Argonne National Laboratory, ANL, Lemont, IL, USA

Abstract

The use of artificial intelligence can significantly reduce the time needed to tune an accelerator system such as the Argonne Tandem Linear Accelerator System (ATLAS) where a new beam is tuned once or twice a week. After establishing automatic data collection procedures and having analysed the data, machine learning models were developed and tested to tune subsections of the linac. Models based on Reinforcement Learning (RL) and Bayesian Optimization (BO) were developed, their respective results are discussed and compared. RL and BO are well known AI techniques, often used for control systems. The results were obtained for a subsection of ATLAS that contains complex elements such as the radio-frequency quadrupole (RFQ). The models will be later generalized to the whole ATLAS linac, and similar models can be developed for any accelerator with a modern control system.

INTRODUCTION

The Argonne Tandem Linear Accelerator System (ATLAS) [1] is a DOE/NP User Facility for studying low-energy nuclear physics with heavy ions. It operates ~6000 h per year. The facility (see Fig. 1), uses three ion sources and services six target areas at energies from ~1–15 MeV/u. To accommodate the total number of approved experiments and their wide range of beam-related requirements, ATLAS reconfigures once or twice per week over 40 weeks of operation per year. The start-up time varies from ~12 to 48 hours depending on the complexity of the tuning, which will increase with the upcoming Multi-User Upgrade designed to deliver beams to two experimental stations simultaneously [2].

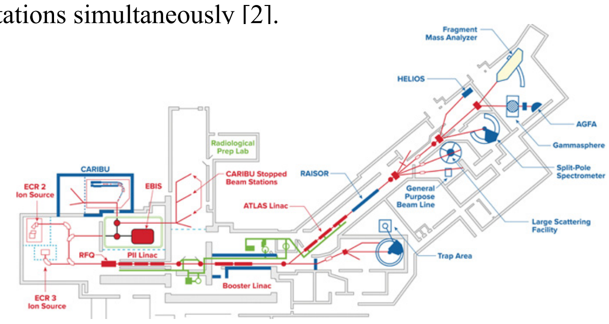

Figure 1: ATLAS Layout.

The procedure of tuning such an accelerator system is time-consuming and relies heavily on the intuition and experience of the operators. The uncertainties involved in tuning are in part due to unknown misalignments of the beamline components and the limited number of diagnostic devices to properly characterize the beam. The use of machine learning (ML) and artificial intelligence (AI) has the potential of filling the information gap and significantly reduce the time needed to tune the accelerator.

By reducing the time for beam tuning, more beam time will be available to help relieve the over-booked experimental nuclear physics program at ATLAS. In addition to beam tuning, AI/ML models can be used to improve beam quality with the installation of new diagnostics and real-time data acquisition. These improvements will increase the facility's scientific throughput and the quality of the data collected.

To support these developments, DOE/NP has approved a project to use AI/ML to support ATLAS operations. Following a description of the project objectives and future plans, the results from the most recent developments will be presented and discussed.

PROJECT OBJECTIVES & PLANS

The main project goal is to use AI/ML techniques to streamline beam tuning and help improve machine performance. The idea is to leverage artificial intelligence for linac operations, as shown in Fig. 2., with the ultimate goal of developing an AI model to tune the machine while also acquiring all kind of information from the AI model that could help improve operations.

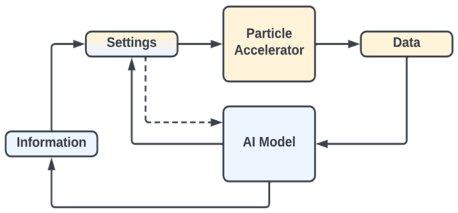

Figure 2: Basic representation of how an AI model could help particle accelerator operations.

The project objectives are threefold:

- Establish data collection, organization, and classification, towards a fully automatic and electronic data collection for both machine and beam data.
- Develop an online tuning model to optimize operations, shorten beam tuning time and make more beam time available for the experimental program.
- Develop a virtual machine model to enhance our understanding of the machine behavior, improve machine performance, optimize particular aspects and help develop new operating modes.

DATA COLLECTION

In any AI project, data collection is the first and most important step. Along with the data collection, cleaning and organizing the data are also the most time-consuming tasks. Therefore, the primary focus at the beginning of this project was on collecting the data on the state of the machine and the beam to be used for AI/ML modeling to support beam tuning and daily machine operations. Due to the challenges that come with training with a real machine and the lack of ATLAS availability, the first steps in designing and training a model should use data from beam physics simulations codes.

Before moving to the developments on the data collection, it is important to know which diagnostic devices are typically available at ATLAS and what kind of data can be obtained from them. Figure 3 shows a typical beamline at ATLAS. The elements that can be found throughout ATLAS beamline are beam profile monitors, electrostatic and electromagnetic quadrupoles that form doublets and triplets, steering magnets, dipoles, faraday cups, accelerator sections, valves, etc.

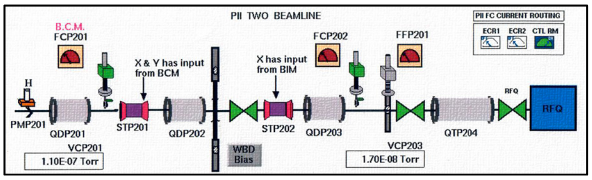

Figure 3: Beamline of a sub-section of ATLAS located between the multi-harmonic buncher and the RFQ.

Therefore, the information that can be obtained from the accelerator comes from faraday cup current readings, beam profiles and voltage and current settings of beamline elements. Currently, efforts are being made to have a functional pepper-pot device that will provide more information through beam imaging.

Prior to this project, the settings (concerning the data collection) could be saved automatically using the Control System, which uses *Vsystem* [3]. However, data from the faraday cups and beam profile monitors could not be saved and access to the devices was not automated. For example, to get a beam profile, several buttons had to be manually pressed on the control system screen. The system was capable of saving setting configurations in a data base to load old beam tunes into the accelerator but was not prepared for any kind of AI work integration.

After understanding how the Control System works, a python package was developed to communicate with it from a server that has direct access to the control system database, see Fig. 4 for a scheme of this communication between systems. The python interface was developed in a way so as to make it useful for any AI related work, thus allowing it to automatically collect and save beam profiles, faraday cup readings and beamline element settings as well as changing the settings of the accelerator.

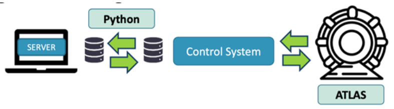

Figure 4: Scheme representing the communication with the Control System from an existing server through the new python package.

Because the idea is to train and deploy AI models with the accelerator, an API was developed that enables communication from any computer with the server, and thus the Control System, exposing only the required functionalities needed when training or deploying a model (see Fig. 5 for a representation of this new communication line). This was necessary because of the old servers that are still in place and are the only ones with direct access to the Control System. This added API layer allows the use of more powerful computers and provides the freedom needed for AI in setting up the software environment.

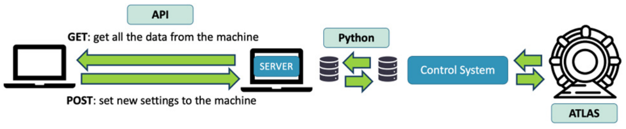
API
GET: get all the data from the machine
POST: set new settings to the machine
SERVER
Python
Control System
ATLAS

Figure 5: Scheme that represents the communication of a computer with the server and Control System through an API.

Apart from the work done on the data collection with the actual machine, a python wrapper was developed around a particle accelerator simulation code, TRACK, to allow the simulation of the actual machine, thereby generating a lot of data with different conditions and inputs and its integration with AI modelling. The AI training offline using a simulation code is key when much time is required for training because of the challenges that come with training directly on the actual machine and the lack of machine availability.

TUNING MODEL

For beam tuning and machine control, the most frequently used AI techniques are reinforcement learning with neural networks [4] and Bayesian optimization using Gaussian processes [5]. Many of these tools and platforms already exist and are available to implement ML models.

The ultimate goal for the tuning model is to optimize operations and shorten beam tuning time, and in order to achieve this the following steps were proposed:

- Develop a baseline model to tune/control a small section of ATLAS Linac using Simulation Data (see Fig. 6).

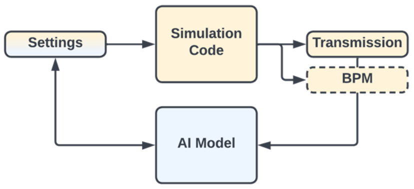

Figure 6: Scheme for model training using simulation code.

- Use alternatives approaches such as Bayesian Optimization with Gaussian Processes and Deep Reinforcement Learning.
- Test the baseline models on the real machine (see Fig. 7). Currently, the project is in this phase.

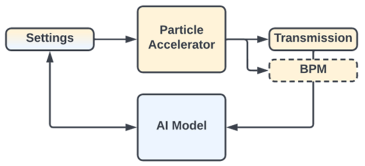

Figure 7: Scheme for model training using the accelerator.

- Improve models to enhance performance and reduce tuning time.
- Expand the model to other parts of the Linac to finally design a tuning model for the whole Linac.

Bayesian Optimization with Gaussian Processes

Bayesian optimization was chosen because of its wide use and because it combines the complementary strengths of human and numerical optimization: life-long learning, learning by experience, juggling many things at once, quick decisions, estimating its own uncertainty, reaching global optimum in a minimum number of steps, etc. This method starts with a prior belief regarding the objective function and then updates it based on samples drawn from the system in order to better approximate the objective function. It uses a probabilistic surrogate model for approximating the objective function and acquisition function that instructs where to query the system next for a more likely improvement.

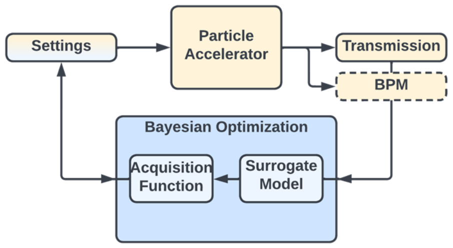

Figure 8: Scheme of how Bayesian Optimization works.

For the baseline model, Gaussian Process with a Matérn kernel, Gaussian likelihood and expected improvement as the acquisition function were used. In addition, the tools employed for BO with GP were GPyTorch and BoTorch libraries.

Simulation Data Using the beamline shown in Fig. 2 and TRACK as the simulation code, different scenarios were optimized by BO with GP. The objective is to maximize the transmission varying the settings.

1. Case for 9 quadrupoles (3 doublets and 1 triplet) and initial set of configurations randomly selected. The model was able to optimize the transmission in around 30 iterations. The initial set, although randomly generated, has a couple of configurations with around 80% transmission, see Fig. 9.
2. Case for 9 quadrupoles, RFQ and initial set of configurations randomly selected. Because the RFQ was added, the beam must match the acceptance of the RFQ; therefore, the randomly selected configurations have very low transmission through the beamline, and the model will need much more time to learn how to adjust the configurarion to match the RFQ acceptance. In the following figure less than 30% transmission is achieved after more than 200 iterations, see Fig. 10.
3. Case for 9 quadrupoles, RFQ and initial set of configurations based on historical beam tunes. The initial set of configurations was scaled from old beam tunes from a historical data base. This provided reasonable transmissions for the model to learn from, which translated into achiving the maximum transimision through the RFQ, around 80% in around 30 iterations, see Fig. 11.

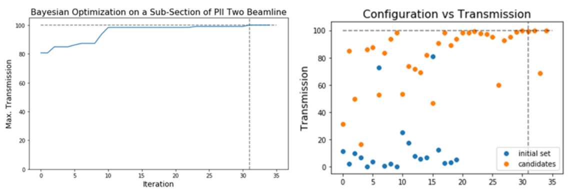

Figure 9: Results obtained for case 1. (Left) Maximum transmission achieved after a number of iterations. (Right) Transmission achieved with different configurations, in blue the initial set of points and in orange the proposed candidates by the model.

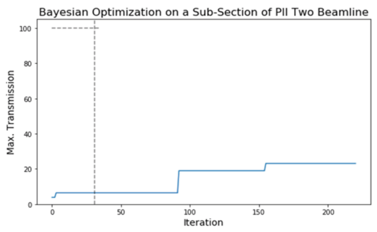

Figure 10: Results obtained for case 2. Maximum transmission achieved after a number of iterations.

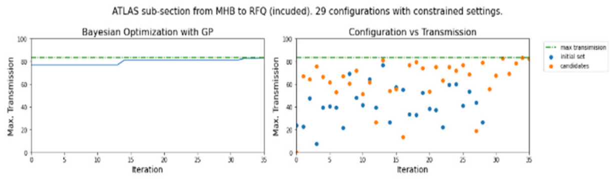

Figure 11: Results obtained for case 3. (Left) Maximum transmission achieved after a # of iterations. (Right) Transmission achieved with different configurations, in blue the initial set of points and in orange the proposed candidates by the model.

This experiment with simulation data confirms two things: first, that it is possible to optimize a beamline in a reasonable amount of time using a basic BO model, and second, the importance of good initial data in the training process when comparing case 2 against case 3.

Real Data The next step was to test the model with the real accelerator. For this experiment, 3 quadrupoles and 2 steering magnets were used (the first ones in Fig. 2). Therefore, there are 7 input parameters, since the 2 steerers have both horizontal and vertical component, and the objective remains to optimize the transmission after the RFQ.

1. Case of $^{14}N^{3+}$: the initial set was composed of 29 historical tunes plus 33 random configurations. The results are shown in Fig. 12. The model was able to converge in a few iterations (~6), although the maximum transmission was achieved around 40 iterations which was higher than the one achieved that day by the operators represented by the green dashed line in Fig. 12 left plot. The final optimum configuration was close but different from the one obtained by the operators, see Fig. 12 right plot.

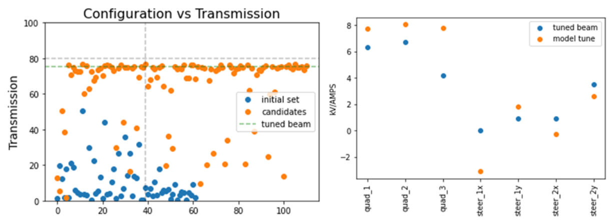

Figure 12: Results obtained for the case of $^{14}N^{3+}$. (Left) Transmission achieved with different configurations, in blue the initial set of points and in orange the candidates proposed by the model. (Right) In blue the element settings from the tuned beam by the operators and in orange the optimum settings proposed by the model.

2. Case $^{40}Ar^{9+}$: the initial set was composed of 29 historical tunes, the randomly selected configurations were not included because they did not provide an additional value owing to their very low transmissions. Similar performance is achieved in this case as in the previous one. Still showed slight improvement in transmission compared to operator configuration, see Fig. 13.

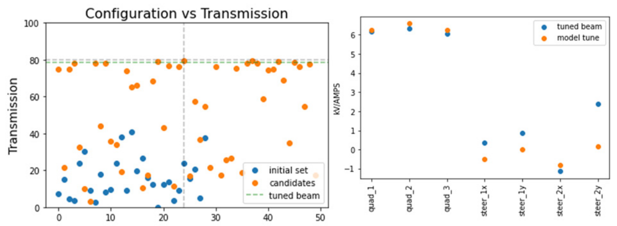

Figure 13: Results obtained for the case of $^{40}Ar^{9+}$. (Left) Transmission achieved with different configurations, in blue the initial set of points and in orange the proposed candidates by the model. (Right) In blue the settings from the tuned beam by the operators and in orange the optimum settings proposed by the model.

Reinforcement Learning

Reinforcement Learning was the other approach chosen for this study. Reinforcement learning is one of the three basic machine learning paradigms, alongside supervised learning and unsupervised learning. RL does not require labeled data because it learns from interactions between an AI agent and its environment. The idea behind using RL to tune/control a particle accelerator arises due to the complexity of a particle accelerator. Taking a look at the classic control problem (Fig. 14), it might seem like creating a single large function would be more difficult than building a control system with piecewise subcomponents; however, this is where reinforcement learning can help.

In essence, RL tries to map situations to actions in order to maximize a numerical reward. Figure 15 shows the different elements of an RL problem.

There are different kinds of algorithm that can be applied, and the one selected as a baseline model was the Deep Deterministic Policy Gradient (DDPG) [6], which is an actor-critic approach that mixes policy optimization and Q-learning method (Fig. 16). Policy optimization methods

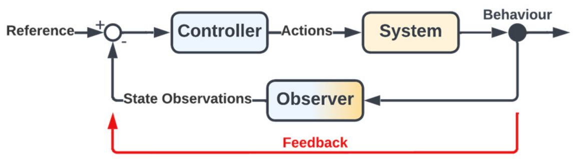

Figure 14: Classic control schema.

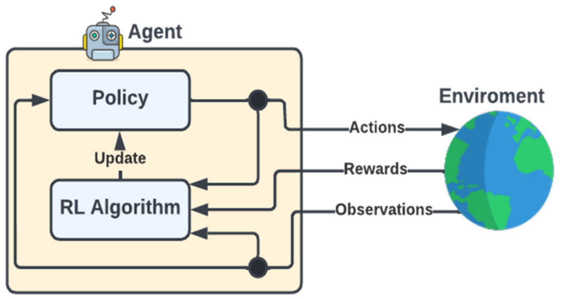

Figure 15: Scheme of how reinforcement learning works and the different elements involved.

tend to be more stable and reliable, and Q-learning methods are substantially more sample efficient, although the latter cannot work with continuous action space, which is the case for accelerators. DDPG works for continuous action spaces because the critic only needs to look at the single action that the actor took and does not need to try to find the best action by evaluating all of them.

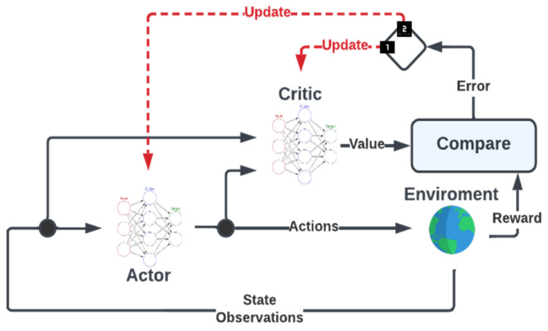

Figure 16: Scheme of how the actor-critic approach works and the different elements involved.

The *actor* is a neural network that tries to take what it thinks is the best action given the current state, as seen within the policy function method. The *critic* is a second neural network that tries to estimate the value of the state and the action that the actor took, as seen within the value function method.

Simulation Data The first scenario to test the performance of a DDPG baseline model was to train a model that minimizes the beam size by varying 3 electrostatic quadrupoles based on simulation data using the TRACK code. The rewards were defined as a logarithmic function of the beam size. Several penalties were added when the settings deviated from the given limits. The limits for the quadrupoles are from 2 kV to 10 kV and the maximum possible action is ±0.25 kV. See Fig. 17 for the training (top) and prediction (bottom) results.

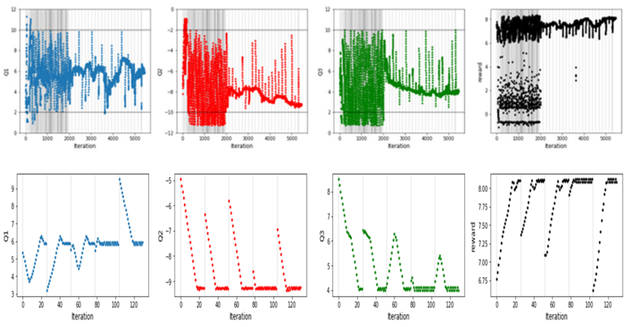

Figure 17: Results obtained for a baseline RL model using DDPG and the TRACK Code. The model optimized the beam size by varying 3 electrostatic quadrupoles. (Top) The results of the training. (Bottom) The results for 5 different predictions starting from 5 different random configurations.

Based on the training results, Fig. 17 top, the model learned first the limits of each quadrupole (up to iteration #2000), and after that, it learned how to optimize the beam size. Regarding the prediction, Fig. 17 bottom, it can be seen how the reward converges in each different scenario to a value corresponding to a minimum beam size.

Real Data After developing the simulation RL model, the next step was to test the baseline model in the real machine. In this case, the objective was to maximize the transmission, which was the reward function, through the selected beamline, see Fig. 2 for more detail. The selected beamline was composed of 4 electrostatic quadrupoles and 2 steering magnets. This gives 8 input parameters in total. The electrostatic quadrupoles were limited from 3 kV to 10 kV with a maximum action of ±0.25 kV and the steering magnets from −1 A to 1 A and a maximum action of ±0.25 A. Although the training was never completed because of limited experiment time with the real machine, some interesting conclusions were drawn. See Fig. 18 for the training results, in the figure it can be seen that the model was be able to learn and identify the limits for almost every element of the selected beamline; it needed more time to finish learning all the elements' limits and optimize the transmission.

Moreover, this experiment confirmed the need to train the RL model offline using simulation code such as TRACK in order to perform only a fine-tune online with the machine. However, there are elements, such as the RFQ, whose simulations are considerably time consuming because of multiple accelerating cells using 3D fields calculations.

Surrogate Model

Following the need of an offline training for some approaches like RL, alongside with the time-consuming simulation of some elements like the RFQ, the idea of developing surrogate models to speed up the simulations was raised. In recent years it has successfully been shown how surrogate models can help in speeding up particle accelerator optimization [7, 8].

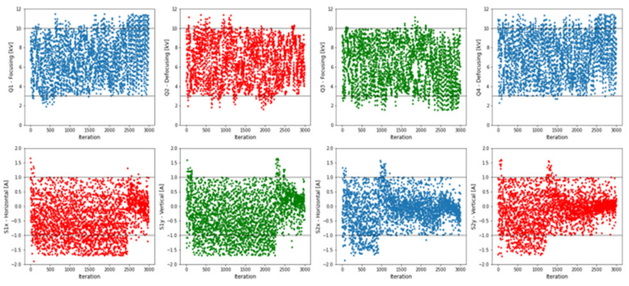

Figure 18: Results obtained for the training of a baseline RL model using DDPG in a subsection of ATLAS. This subsection was composed of 4 electrostatic quadrupoles (top) and 2 steering magnets with both vertical and horizontal components (bottom).

A surrogate model can be trained on beam simulation data to reliably reproduce the physics results in very short time; then it could be enhanced with experimental data. ML Surrogate Model can be used for virtual diagnostics, offline experiment planning, design of new setups, control and tuning.

In order to speed up the RL offline training through the RFQ, a surrogate model for the ATLAS RFQ can be used. A surrogate model was developed previously using the TRACK code [9] and could be used for RL offline training. The model was based on neural network architecture. – specifically, it was composed of two hidden layers and two residual blocks. The main objective was to predict beam transmission and output beam Twiss parameters as functions of input beam parameters which include the beam emittances and input Twiss parameters. The agreement between the simulation code and the surrogate model was excellent and is consistent with the comparison of two beam dynamics codes. Therefore, the surrogate model can be considered reliable and capable of reproducing the physical results, with the big advantage of being ~ 30,000 faster than the 3D model in this case. This is exactly what is needed for speeding up simulations.

CONCLUSIONS AND NEXT STEPS

Bayesian Optimization and Reinforcement Learning can be considered as analogous concepts with different terminology and often different settings. BO and RL both are useful for high-level tuning and control but excel in different regimes. BO seems to be more exploratory – ideal for optimizing new settings and situations in the limited data regime or slow measurements, while RL needs significantly more data to be trained and focusing more on continuous control.

From these preliminary results it seems that BO would be more suitable for new tuning configurations and RL for continuous control after being pre-trained offline.

The project has reached several milestones such as automated data collection, integration of new devices as the pepper-pot, integration of AI modelling with the accelerator and the successful training and deployment of a BO with GP on a subsection of ATLAS. In addition, the first step for a successful RL model has been made in demonstrating the model's learning the limits of the elements.

However, there is still a lot to do, and the next steps will be focused on testing the pepper-pot integration with the control system and the modelling, getting more useful data from the machine, fine tuning the RL with the machine after training offline, and improving existing models. To do so, other architectures, new type of data such as beam profiles and pepper-pot images, the use of surrogate models and the incorporation of more Physics information into the systems will be considered and included. And last, but not least, one of the current challenges encountered during the experiments was the possible damage to some devices when the beam is lost during model training. A solution is being investigating.

REFERENCES

[1] P. N. Ostroumov *et al.*, "Completion of Efficiency and Intensity Upgrade of the ATLAS Facility", in *Proc. LINAC'14*, Geneva, Switzerland, Aug-Sep. 2014, paper TUPP005, pp. 449-451.

[2] B. Mustapha *et al.*, "The ATLAS multi-user upgrade and potential applications", *J. Instrum.*, vol. 12, p. T12002, Dec. 2017. doi:10.1088/1748-0221/12/12/t12002

[3] P. Clout, "The status of Vsystem", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 352, pp. 442-446, 1994. doi:10.1016/0168-9002(94)91565-2

[4] V. Kain, S. Hirlander, B. Goddard, F. M. Velotti, G. Z. Della Porta, N. Bruchon, and G. Valentino, "Sample- efficient reinforcement learning for CERN accelerator control," *Phys. Rev Accel. Beams*, vol. 23, p. 124801, Dec 2020. doi:10.1103/PhysRevAccelBeams.23.124801

[5] M. W. McIntire, T. M. Cope, S. Ermon, and D. F. Ratner, "Bayesian Optimization of FEL Performance at LCLS", in *Proc. IPAC'16*, Busan, Korea, May 2016, pp. 2972-2975. doi:10.18429/JACoW-IPAC2016-WEPOW055

[6] 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", 2016. doi:10.48550/arXiv.1509.02971

[7] M. Kranjčević, B. Riemann, A. Adelmann, and A. Streun, "Multiobjective Optimization of the Dynamic Aperture Using Surrogate Models Based on Artificial Neural Networks", *Phys. Rev. Accel. Beams*, vol. 24, p. 014601, 2021. doi:10.1103/PhysRevAccelBeams.24.014601

[8] L. Gupta, A. Edelen, N. Neveu, A. Mishra, C. Mayes, and Y.-K. Kim,. "Improving surrogate model accuracy for the LCLS-II injector frontend using convolutional neural networks and transfer learning", *Mach. Learn.: Sci. Technol.* vol. 2, p. 045025, 2021. doi:10.1088/2632-2153/ac27ff

[9] B. Mustapha, B. Blomberg, C. Dickerson, J. M. Marin, and C. Peters. "AI-ML Developments for the ATLAS Ion Linac Facility", in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 4122-4125. doi:10.18429/JACoW-IPAC2021-THPAB181