

# ADDING MACHINE LEARNING TO THE ANALYSIS AND OPTIMIZATION TOOLSETS AT THE LIGHT SOURCE BESSY II

L. Vera Ramirez\*, T. Mertens, R. Mueller, J. Viehhaus, G. Hartmann  
Helmholtz-Zentrum Berlin, Germany

## Abstract

The Helmholtz Association has initiated the implementation of the Data Management and Analysis concept across its centers in Germany. At Helmholtz-Zentrum Berlin, both the beamline and the machine (accelerator) groups have started working towards setting up the infrastructure and tools to introduce modern analysis, optimization, automation and AI techniques for improving the performance of the (large scale) user facility and its experimental setups. This paper focuses on our first steps with Machine Learning (ML) techniques over the past months at BESSY II as well as organizational topics and collaborations. The presented results correspond to two complementary scenarios. The first one is based on supervised ML models trained with real accelerator data, whose target are real-time predictions for several operational goals (beam lifetime, injection efficiency, beam loss...); some of these techniques are also used for additional tasks such as outlier detection or feature importance analysis. The second scenario includes first prototypes towards self-tuning of machine parameters in different optimization cases (booster current, injection efficiency, orbit correction...) with Deep Reinforcement Learning (RL) agents.

## MOTIVATION

The integration process of ML tools with real accelerator data at BESSY II is being carried out with two main goals: modelling and prediction on the one hand and self-optimization on the other. As for today many specific prediction models for different accelerator parameters have been already constructed and analyzed - apart from the beam lifetime case presented here, different beamloss monitors along the ring as well as injection efficiency have been modeled. This is also an important preparatory step for the RL-based tuning as well as for the self-optimization of surrogate models. Besides, further significant effort is being put into beamline raytracing and the conception of *digital twins*, which can be also connected with the RL-based self-optimization. The aim of this paper is to summarize some of this application cases as a sort of *proof-of-concept* of the major possibilities opened by the incorporation of ML tools at BESSY II.

## PREDICTION OF BEAM LIFETIME

We present a representative case of beam lifetime prediction restricted to a *blind* scenario: time-series-based prediction of beam lifetime *only* with context variable readbacks, i.e., omitting the previous measurements of the target variable. This scenario allows us to identify unknown correla-

tions and patterns in the readbacks avoiding an excessive reliance on the previous target variable measurements but also to reuse the information and experience gained with the prediction models in a RL context.

The beam lifetime  $\tau$  is defined through the current decay rate  $\frac{1}{\tau} = -\frac{\dot{I}}{I}$ , where  $I$  denotes the beam current (for a study of the beam lifetime at BESSY II see, e.g., [1]). For these experiments we approximated the *instantaneous* lifetime through a piecewise linear regression with  $k$  previous measurements of the beam current  $I_t$  (usually  $k = 20$  seconds):

$$\frac{1}{\tau} \approx -\frac{1}{I_t} \frac{\sum_{i=0}^k (I_{t-i} - I_{t_0}) (t - i - t_0)}{\sum_{i=0}^k (t - i - t_0)^2}$$

The potentially beam lifetime affecting variables used as input (185 after preprocessing, see Appendix) are:

- Gap and shift of insertion devices (elliptical) undulators affecting the dynamic aperture (21 readback variables).
- Power supply currents into quadrupoles define the linear optics (58 readback variables), into sextupoles define non linear behavior (7 variables).
- Offsets to power supplies for quadrupoles define the feed forward compensations (38 variables).
- Collisions with rest gas particles, vacuum pressures measured by getter pump current (12 variables).
- Local beam loss fractions, monitored by counters close by (49 variables).

## Comparison of Methods

We have worked with the following supervised learning models:

- Extremely Randomized Trees (ExtraTrees, [2]).
- Support Vector Regression *approximated* with Random Fourier Features (SVR-RFF, [3]).
- Standard dense feed-forward Neural Networks (DNN, e.g., [4]).

Table 1 contains the results of these prediction models for two different test set elections: a random uniform set (20%) along the measurement period or the last 20% of the measurement period. The different hyperparameter configurations after grid search as well as further test settings can be found in the Appendix<sup>1</sup>.

<sup>1</sup> Other tested algorithms (traditional Random Forests and SVR with different kernels) presented similar or worse results so we excluded them from the table for the sake of clearness.

\* luis.vera\_ramirez@helmholtz-berlin.de

Table 1: Beam Lifetime Prediction Experiments

Test set	Algorithm	RMSE				$R^2$		
		Avg.	Pers.	Mov. pers.	Model	Pers.	Mov. pers.	Model
Random 20%	ExtraTrees	0.201319	0.099248	0.091464	$0.068175 \pm 0.000038$	0.756961	0.79359	$0.885322 \pm 0.000128$
	SVR-RFF				$0.077432 \pm 0.000216$			$0.852064 \pm 0.000825$
	DNN				$0.069457 \pm 0.000342$			$0.880964 \pm 0.001177$
Last 20%	ExtraTrees	0.231393	0.095732	0.078776	$0.194755 \pm 0.000952$	0.828836	0.884099	$0.291586 \pm 0.006932$
	SVR-RFF				$0.121407 \pm 0.003349$			$0.724506 \pm 0.015291$
	DNN				$0.125046 \pm 0.005757$			$0.707345 \pm 0.027032$

Specially interesting for us are the results with the last 20% as test set, since they represent a major challenge for the prediction models - as we see with ExtraTrees, the random test set can be successfully predicted with probably overfitted models but perform poorly in the chronological test set. Nevertheless, DNN and SVR-RFF do manage to forecast quite accurately the trends in the ca. 3 days of completely unseen data at the end of the measurement period (Figs. 1 and 2), even keeping the same hyperparameter configuration used for the random test set in the case of SVR-RFF. In other words, these models present good results with a *time-series-based* evaluation, although they predict with no reference of the previous lifetime measurements, only with respect to the current accelerator readbacks.

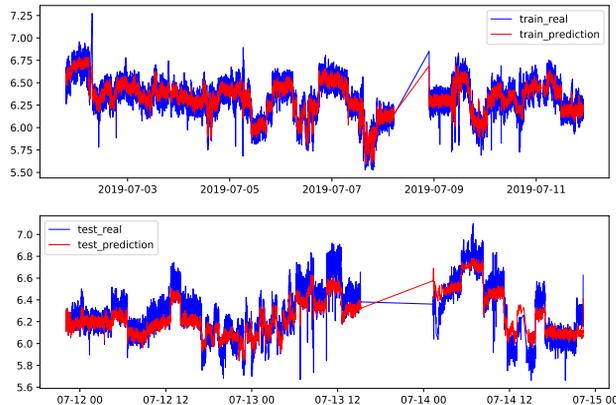


Figure 1: Prediction of beam lifetime with SVR-RFF.

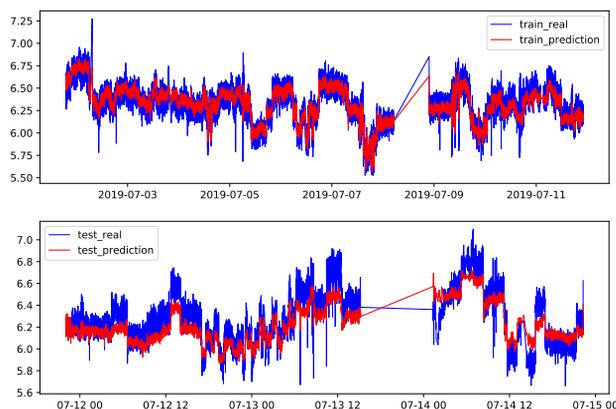


Figure 2: Prediction of beam lifetime with DNN.

## OPTIMIZATION OF BOOSTER CURRENT

The second main development area regarding ML at BESSY II is the use of Deep Reinforcement Learning (Deep-RL) for the self-optimization of accelerator parameters. For this, we are developing a new framework called *RLControl*.

The initial case tested with *RLControl* at BESSY II is the optimization of booster current. It has been observed that, after long interruptions of the machine operation, the booster current tends to be low. As for today, manual parameter tuning is required to recover acceptable current values.

The global environment description for the several tests carried out in the booster current optimization case is the following<sup>2</sup>:

- State variables:
  - High (radio) frequency - master clock.
  - Voltage in LINAC.
  - Two klystron current diagnostic measurements.
- Action variable: time phase in LINAC<sup>3</sup>
- Reward: (normalized) booster current per bunch.

### Deep Reinforcement Learning

*RLControl* is based on Deep Deterministic Policy Gradient (DDPG, [5]). DDPG is a recent actor-critic, model free deep-RL algorithm for continuous environments. As many other approaches in RL, this algorithm is based on the update of the so-called action-value function  $Q : S \times A \rightarrow \mathbb{R}$  with deterministic target policy  $\mu : S \rightarrow A$  making use of the recursive relationship known as Bellman equation:

$$Q^\mu(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim E} [r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_t))]$$

where  $s_t \in S$  and  $a_t \in A$  denotes the state resp. action at time  $t$ ,  $s_{t+1} \in S$  denotes the next state returned by the environment  $E$  and  $r(\cdot, \cdot)$  is the *reward* function. In the DDPG agent proposed in [5] both  $Q$ -function and policy are *approximated* with neural networks, and further relevant implementation tricks are proposed: delayed target networks ( $Q_{\tilde{\phi}}, \mu_{\tilde{\theta}}$ ), replay buffer...

<sup>2</sup> Further algorithm and test settings can be found in the Appendix.

<sup>3</sup> The election of LINAC time phase as action variable is supported by previous observations showing that the modification of this parameter does not affect the injection efficiency.

Content from this work may be used under the terms of the CC BY 3.0 licence © 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

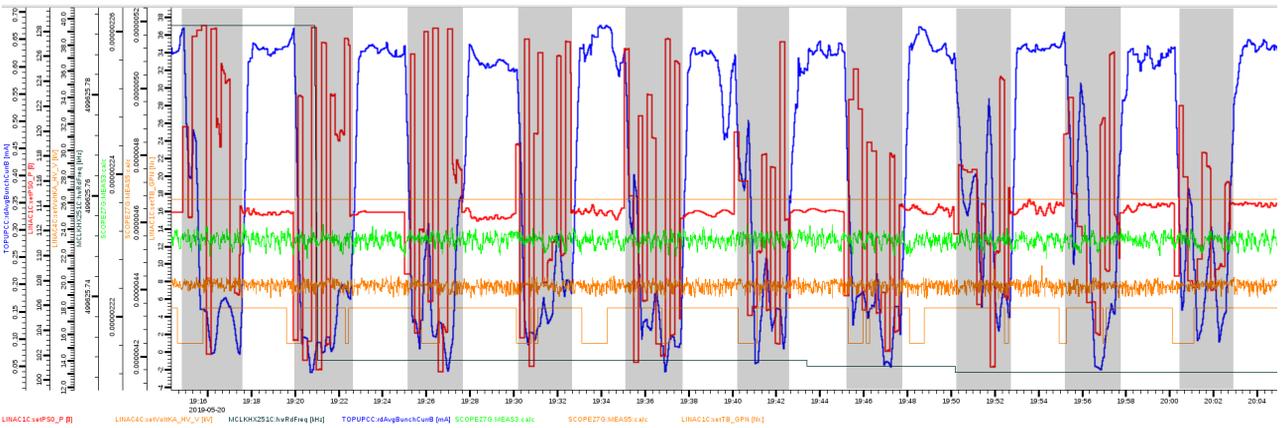


Figure 3: RLControl - Preliminary short test for booster current optimization. The reward (instant booster current per bunch) is plotted in blue, while the actions (LINAC time phase) are plotted in red - the remaining lines correspond to state variables. The agent is pre-trained with 30 days of historical data (top-up mode) as demonstration. Exploration periods are shaded.

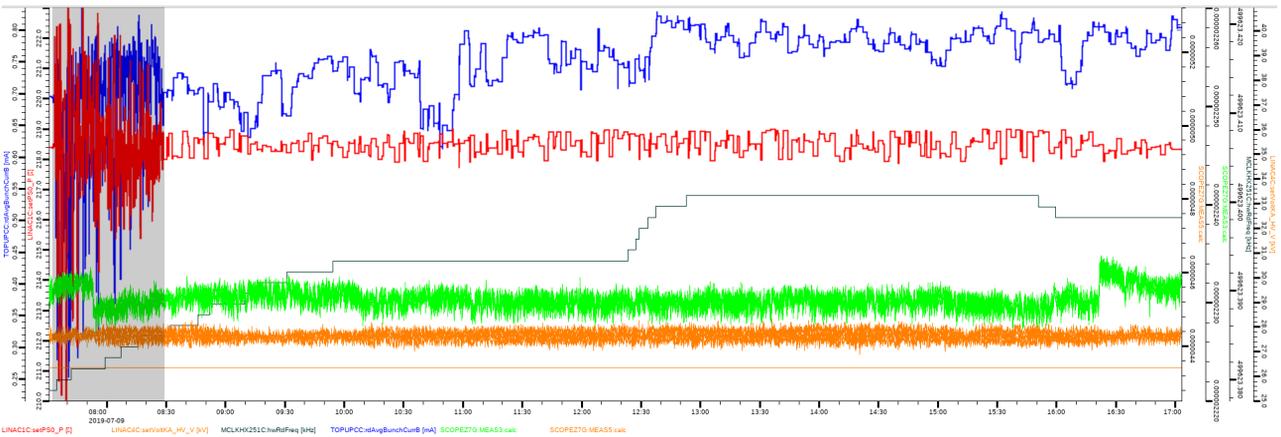


Figure 4: RLControl - Long test for booster current optimization during user operation with automatic exploration schedule. Same color configuration as Fig. 3. The agent is pre-trained with 30 days of historical data (top-up mode). Exploration with automatic schedule appears shaded.

### Tests

Figure 3 shows and describes a successful preliminary test with *RLControl* for booster current optimization during machine commissioning time. During the test it alternates periods of exploration - shaded in the plot - and optimization, i.e., learning with no exploration. Exploration is carried out with Parameter Space Noise, which brings stability and improves each optimization period. The length of the periods (ca. 2 min) approximates the average injection time interval in top-up mode, so the test suggested that fruitful exploration can be carried out with automatic scheduling between injections. Nevertheless, many issues had to be corrected during the first preliminary tests until the satisfactory agent behavior observed in Fig. 3 was achieved - for example:

- Long training time (points every 2 seconds, *brute force* synchronization) and normalization problems → improved through demonstration with historical data (inspired by the ideas of expert demonstration in [6]).

- Slow *reaction time* to reward modifications → solved by giving up average current as reward and using the instantaneous current per bunch<sup>4</sup>.
- Non-optimal exploration → solved by implementing Parameter Space Noise ([7]).

Figure 4 shows and describes the most important test so far with *RLControl* because of its duration (ca. 9.5 hours) and also because it was not carried out during pure machine commissioning but during user operation time at BESSY II. Therefore, an automatic schedule had to be conceived in order to restrict exploration to the meantime between injections.

In the test plotted in Fig. 4 exploration with automatic schedule is carried out during the first hour - although not continuously: pure exploration is scheduled to take place only in the meantime between injections during this first

<sup>4</sup> In the plots the average booster current variable is always showed, in order to improve the visibility.

hour. This scheduling allowed the agent to avoid sub-optimal booster current at the injection point that might interfere with the user activity. Optimization (learning without exploration) is activated always shortly before each injection (see Fig. 5). The agent kept optimizing (and learning) successfully during the next 8.5 hours of user operation.

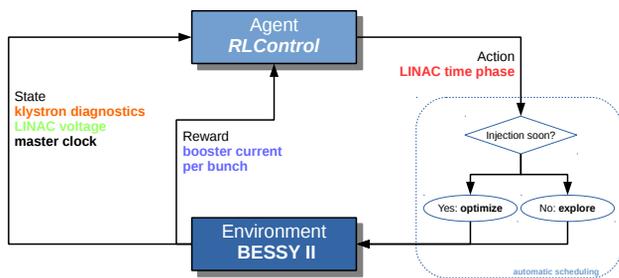


Figure 5: RL pipeline for booster current optimization with automatic scheduling.

## OPTIMIZATION OF INJECTION EFFICIENCY

The most recent application of *RLControl* at BESSY II is the injection efficiency case. Nowadays manual tuning is also required for improvement of the injection efficiency, which is observed to be affected by temperature. This motivated us to put effort into an automatized solution for this case as well. The environment description for the first injection efficiency optimization tests is the following:

- State variables:
  - Number of bunches generated by the LINAC (1, 3 or 5)
  - Injection angle mismatch, measured by the horizontal and vertical beam position in the transfer line.
  - Current measured during the booster acceleration phase (averaged per pulse).
  - Measured loss rate after extraction from the booster.
- Action variable: Deflection angle into the storage ring, generated by the 2<sup>nd</sup> septum.
- Reward: last injection efficiency, measured as fraction of current increase generated in the storage ring by the charge accelerated in the booster.

### Tests

First injection efficiency tests with *RLControl* during machine commissioning time presented some issues regarding pretraining with historical data, mainly due to the fact that septum external conditions vary along time, inducing variations in the optimal action intervals that were not reflected

in the chosen state variables. This fact slowed down enormously the learning process of these first tests, since in this case we only get one data sample per injection (in average every ca. 2 minutes in multibunch mode). Although the agents of these first tests (trained from scratch) managed to find good actions after very few training steps (ca. 100) in stable conditions and even for different number of bunches, they failed when we forced the agent to face certain unknown states - in particular when booster current was modified internally.

Figure 6 shows the first preliminary test for injection efficiency where pretraining with historical data was carried out successfully. For this, we reduced the considered period of historical data to 23 days but included also data from non-top-up mode in order to increase the range of actions observed in the pretraining phase. In this test the agent manages to find and improve the optimal action regions during the first phase of *natural* machine conditions (until ca. 18:40) but also in the second phase, where we internally modified the booster current and the number of bunches.

## UNDERSTANDING BEAMLINE PERFORMANCE

Beamline raytracing is a powerful tool to understand X-ray-beam propagation and for optimizing beam properties for the experimental requirements. However, at today's beamlines at synchrotrons and FELs one needs many components having altogether a couple of hundred parameters to fulfill these needs. This makes it impossible to map the full parameter space with traditional simulation tools. We approach this challenge with various deep learning methods (autoencoder, convolutional neural networks, tensor products, extreme gradient boosting, k-nearest neighbors, automatic-differentiation, ...) learning raytracing as well as beamline parameter prediction from photon diagnostics.

The beamline parameters cover a variety of mirror, grating, slit and source properties: misalignments, misorientations, offsets, slope errors, radii, entrance arm length, line density, fix focus constant, thermal distortion, etc. This allows, on the one hand, for predictions of X-ray footprints at specific positions and also of full traces through the entire beamline and on the other hand, the determination of the current state of a beamline becomes accessible by simple diagnostics in combination with the neural network. Additionally, tuning the beamline to specific user demands can now be handled by the ML model providing a high-dimensional solution in contrast to sequential beamline parameter tuning.

Figure 7 shows an example of the inversion of beamline raytracing. The chosen diagnostic are photon footprint screens at three different positions in the beamline. These are used from the neural network to predict 28 essential beamline parameters (100 parameters were varied in the training data).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

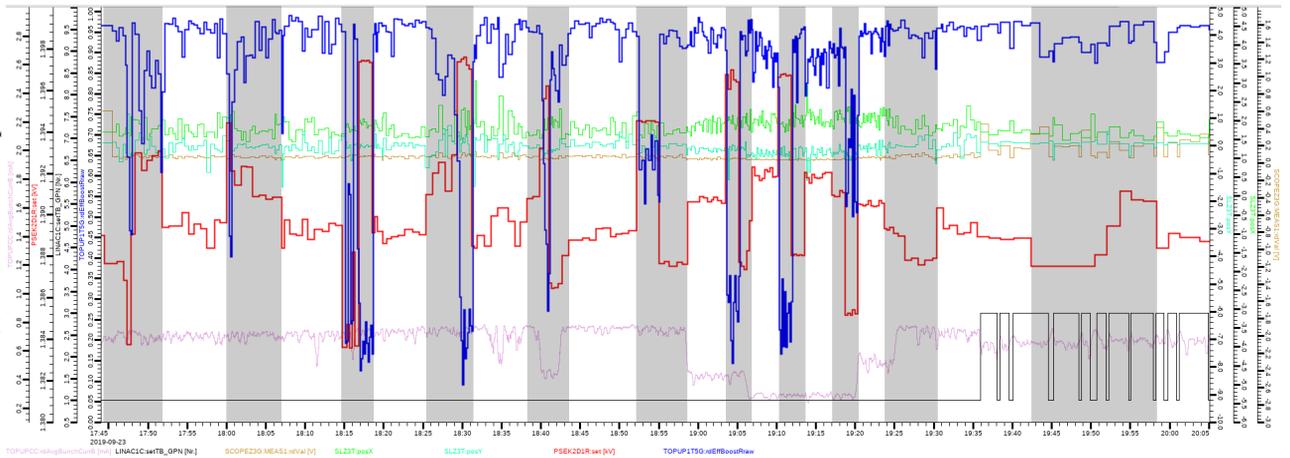


Figure 6: RLControl - Preliminary test for injection efficiency optimization. The agent is pretrained with 23 days of historical data. Reward (injection efficiency) is plotted in blue, actions (septum deflection angle) is plotted in red. *Ad-hoc* modifications of the number of pulses (in black) and booster current (in purple) are carried out during the test. Exploration periods appear shaded.

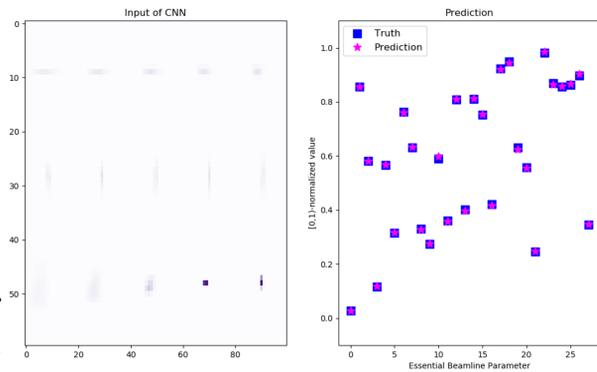


Figure 7: Inversion of beamline raytracing.

## MODELLING INTEGRATION: DIGITAL TWIN

### The Accelerator/Source Side

It is worth briefly mentioning an additional direction for deep-RL agents we are also currently investigating, which is based on the multiphysics simulation toolkit OCELOT ([8]). This framework gives us the possibility of using surrogate models for the training of deep-RL agents, complementing or replacing the pretraining with historical data. Some tests with toy-examples (emittance, orbit-correction...) in small lattices have been already carried out. Nevertheless, the major challenge to be accomplished is the *export* of a RL-agent trained with the virtual BESSY-II-lattice to the real accelerator.

### The Beamline/X-ray Side

Within Helmholtz Association and the Helmholtz-Zentrum Berlin there is a general initiative to take advantage of ML methods for process analysis and optimization and for the retrieval of scientific data from the measurements.

At the light source there is a close coordination of activities developing toolsets for the accelerator and the beamlines [9]. The available data from previous operation periods are still scarce. Nevertheless activities focus on improving the prerequisites for common ML methodologies.

## CONCLUSION

The main achievements presented in this paper can be summed up as follows:

- Beam lifetime at BESSY II can be successfully predicted in a time-series fashion through supervised learning models trained only with 185 accelerator variables readbacks - i.e., excluding previous lifetime measurements.
- The accelerator self-optimization framework *RLControl* was able to solve a first use case (booster current maximization) at BESSY II, being tested even during 8.5 hours of user operation with the help of injection-based automatic scheduling.
- Further effort has been put into the application of *RLControl* for more advanced use cases such as injection efficiency optimization, leading also to successful preliminary tests.
- Beamline performance analysis (inversion of beamline raytracing with ML tools) is being also investigated with promising results. This involves also proposal for re-commissioning plans.

Some of the next steps in the roadmap for the integration of ML tools at BESSY II are:

- Measurement prediction framework:
  - Build models for further target variables such as purity.

Table 2: Lifetime Prediction Experiments - Grid Search

Split	Algorithm	Chosen hyperparameter configuration
Random	ExtraTrees	(bootstrap, *True), (max_depth, None), (max_features, None), (n_estimators, 500)
	SVR-RFF	(batch_size, *32), (epochs, 50), (gamma, *1/n_atts), (loss, mse), (mode, rff), (n_components, 5000), (optimizer, adagrad)
	DNN	(activation, relu), (batch_size, *32), (dropout_rate, 0.1), (epochs, *20), (hidden_layers, 200+200+100+50+25+12), (intermediate_dropouts, first), (loss, mse), (optimizer, adagrad)
(activation, *tanh), (batch_size, *32), (dropout_rate, 0.05), (epochs, *20), (hidden_layers, *200+200), (intermediate_dropouts, all), (loss, mse), (optimizer, adagrad)		
Chronological		

- Classification approach.
- Surrogate models.
- *RLControl*:
  - Further tests, investigation and use cases (injection efficiency with more state and action variables, orbit correction with OCELOT pretraining...).
  - Integration of advanced data collection tools such as Bluesky ([10]).
  - User interfaces.
- Beamline adjustment and automated optimization
- Persistence: previous lifetime measurement.
- *Moving persistence*: moving average of the last 5 lifetime measurements.
- Hyperparameter optimization: grid-search with 5-folded cross validation. The chosen configurations can be found in Table 2 - fixed hyperparameters along the grid search are marked with \*. For NNs two searches (with random and chronological splits) had to be carried out; this was not necessary for the other models, since the configuration obtained from the random split performed well.
- Data preprocessing:
  - Outlier detection with Isolation Forest ([15]) with contamination 0.02.
  - [-1, 1] linear normalization.
  - PCA of the input variables (with 185 components).

## ACKNOWLEDGEMENTS

The *RLControl* experiments would not have been possible without the cooperation and feedback of Andreas Schällicke, Thomas Birke, Markus Ries, Paul Goslawski and Terry Atkinson.

## APPENDIX

### Implementation

- Supervised learning for measurement prediction: `scikit-learn` ([11]) and `keras` ([12]) with `tensorflow 1.12` backend ([13]).
- *RLControl*: our implementation relies essentially on `keras-rl` ([14]), but we have incorporated some extensions such as Layer Normalization, Parameter Space Noise and pretraining with historical data.

### Beam Lifetime Prediction - Settings

- Measurements period from 2019-07-01 19:00:00 until 2019-07-16 19:00:00, restricted to top-up and multi-bunch. From this, 80% (31631 samples) is used for training and 20% for test (7908 samples). In both cases of test set election (random 20% and last 20%) the following baselines are used:
  - Test set lifetime average.

### RLControl - Settings

- Neural networks: in both cases, `relu` used as inner activation function and `adam` as optimizer ( $lr = 0.001$ ).
  - Critic network: five hidden layers (25+50+25+10+5 neurons) and concatenates actions at the first hidden layer. Linear activation at the output layer.
  - Actor network: three hidden layers (25+10+5 neurons), all of them with layer normalization ([16]). `tanh` used as activation for the output layer.
- Data preprocessing: [-1, 1] linear normalization, historical data downsampled to 60 seconds in the booster current case.
- Parameter Space Noise:  $\delta = 0.01$ ,  $\alpha = 1.1$ , initial  $\sigma = 0.2$ .
- Training parameters:  $\gamma = 0.2$ , pretraining with 10000 steps (2000 before actor training in the booster current case), warm-up with 32 steps, target model update rate: 0.01.

## REFERENCES

- [1] S. Khan, "Study of the BESSY II beam lifetime," in *Proceedings of the 1999 Particle Accelerator Conference (Cat. No. 99CH36366)*, vol. 4, Mar. 1999, 2831–2833 vol.4. doi: 10.1109/PAC.1999.792953.
- [2] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr. 2006, ISSN: 1573-0565. doi: 10.1007/s10994-006-6226-1. <https://doi.org/10.1007/s10994-006-6226-1>
- [3] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., Curran Associates, Inc., 2008, pp. 1177–1184. <http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf>
- [4] R. Rojas, *Neural networks : A systematic introduction*. Berlin, New York: Springer-Verlag, 1996, ISBN: 978-3-540-60505-8.
- [5] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. <http://arxiv.org/abs/1509.02971>
- [6] X. Zhang and H. Ma, "Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations," *CoRR*, vol. abs/1801.10459, 2018. arXiv: 1801.10459. <http://arxiv.org/abs/1801.10459>
- [7] M. Plappert *et al.*, "Parameter space noise for exploration," *CoRR*, vol. abs/1706.01905, 2017. arXiv: 1706.01905. <http://arxiv.org/abs/1706.01905>
- [8] I. Agapov, G. Geloni, S. Tomin, and I. Zagorodnov, "OCELOT: A software framework for synchrotron light source and FEL studies," *Nuclear instruments & methods in physics research/A*, vol. 768, pp. 151–156, 2014, (c) Elsevier B.V., ISSN: 0168-9002. doi: 10.1016/j.nima.2014.09.057. <http://bib-pubdb1.desy.de/record/192826>
- [9] R. Müller *et al.*, "Modernization of experimental data taking at BESSY II," MOCPL02, Proceedings of this conference, 2019.
- [10] Bluesky, <https://github.com/bluesky/bluesky>.
- [11] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [12] F. Chollet, *Keras*, <https://github.com/fchollet/keras>, 2015.
- [13] Martin Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. <https://www.tensorflow.org/>
- [14] M. Plappert, *Keras-rl*, <https://github.com/keras-rl/keras-rl>, 2016.
- [15] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08, Washington, DC, USA: IEEE Computer Society, 2008, pp. 413–422, ISBN: 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.17. <http://dx.doi.org/10.1109/ICDM.2008.17>
- [16] J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv*, vol. abs/1607.06450, 2016.