Machine learning-based optimization of storage ring injection efficiency

Journal of Physics: Conference Series **2687** (2024) 062033 doi:10.1088/1742-6596/2687/6/062033

Machine learning-based optimization of storage ring injection efficiency

D Schirmer, A Althaus, S Hüser, S Khan, T Schüngel
Center for Synchrotron Radiation (DELTA), TU Dortmund University, Germany
E-mail: detlev.schirmer@tu-dortmund.de

Abstract. Since the DELTA accelerator facility does not use a White circuit-driven fast topping-up mode, each software-driven injection ramp cycle takes about 7 seconds. Depending on the injection efficiency, 150 to 200 ramp cycles are required to reach the maximum beam current of 130 mA in the storage ring. Thus, for fast post-injection, a high electron transfer rate is crucial. During the injection process, a large number of parameters (e. g., magnet settings, timings of pulsed elements) have to be adjusted manually. The injection efficiency depends mainly on the settings of the booster extraction elements, the T2 transfer line magnets, and the storage ring injection components. In order to automate the injection procedure and to improve the electron transfer efficiency, the application of innovative machine learning concepts was studied.

1. Introduction

The DELTA facility does not apply a classical resonating White circuit to operate the booster synchrotron (BoDo). Instead, the booster magnets and radio frequency (RF) system are ramped software-driven. Each energy ramp cycle from 70 MeV to 1.5 GeV requires approx. 7 seconds. In order to minimize the injection time, it is essential to keep the electron transfer rate (injection efficiency) from BoDo to the storage ring high. Already in 2005, first attempts were made to optimize the injection efficiency by a combination of genetic algorithms (GAs) and neural networks (NNs) [1, 2]. This idea was taken up again, but now with an expanded number of injection parameters and a significantly enlarged data pool for machine learning (ML) based supervised training.

The injection efficiency $E_{inj} = \Delta Q_{\mathrm{DELTA}}/Q_{\mathrm{BoDo}}$ is defined by the ratio of charge change $\Delta Q_{\mathrm{DELTA}}$ in the storage ring to the charge transfer from the booster Q_{BoDo}. The charge is $Q = I \cdot T$, where the beam currents I and the revolution time T are known. The beam current in BoDo is determined before extraction into the T2 transfer line (see Fig. 1) and the DELTA beam is measured approx. 0.5 seconds before and after the extraction trigger, respectively. Taking into account the revolution times of 168 ns in BoDo and 384 ns in DELTA, this gives the charge increase in the storage ring. Thus, the transfer efficiency E_{inj} is given by:

$$E_{inj} = \frac{\Delta Q_{\mathrm{DELTA}}}{Q_{\mathrm{BoDo}}} = \frac{\Delta I_{\mathrm{DELTA}}}{I_{\mathrm{BoDo}}} \cdot \frac{384}{168}.$$

The measurement error amounts to a few percent depending mainly on the percentage of transferred charge and the lifetime of the stored beam [3]. The efficiency fluctuations (see Fig. 2)

are primarily caused by jitter effects of the trigger timings for the pulsed injection components.

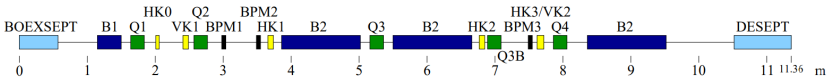

Figure 1. Schematic view of the magnet setup of the second transfer line T2, which connects the booster synchrotron BoDo (left) and the storage ring DELTA (right) ([1], modified). The main elements are bending- (B), quadrupole- (Q), and horizontal/vertical corrector (HK/VK) magnets.

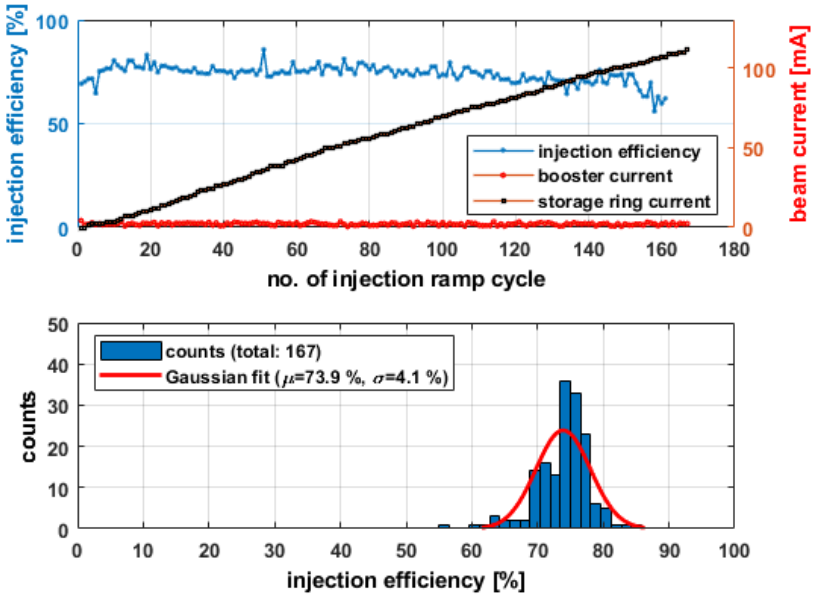

Figure 2. Example of a standard injection sequence at DELTA. Here, 167 booster injection ramp cycles were needed to fill the storage ring from 0 mA up to 111 mA. The averaged injection efficiency is $73.9\,\% \pm 4.1\,\%$. The average booster current amounts to $2.14\,\text{mA} \pm 0.74\,\text{mA}$.

2. Machine learning workflow

In this work, the measurement for ML-data acquisition (DAQ), model training, and deployment at the accelerator were carried out separately. The detailed workflow is shown in Fig. 3. To collect ML-training data, two different methods of measurements were implemented. On the one hand, each injection parameter was systematically scanned individually (see Fig. 4) whereby all other parameters remained unchanged. These data result in a sensitivity matrix for the injection efficiency. On the other hand, all parameters were randomly (Gaussian) varied at once (see Fig. 5). In this way, several thousand data sets were recorded. Subsequently, the data pool was cleaned (e. g., outliers removed), normalized and rescaled to difference values. For this purpose, the corresponding change in efficiency was calculated for each parameter variation. With these revised data records, surrogate injection models were trained by the use of different supervised machine learning algorithms [4]. Finally, the trained models serve optimizing algorithms like

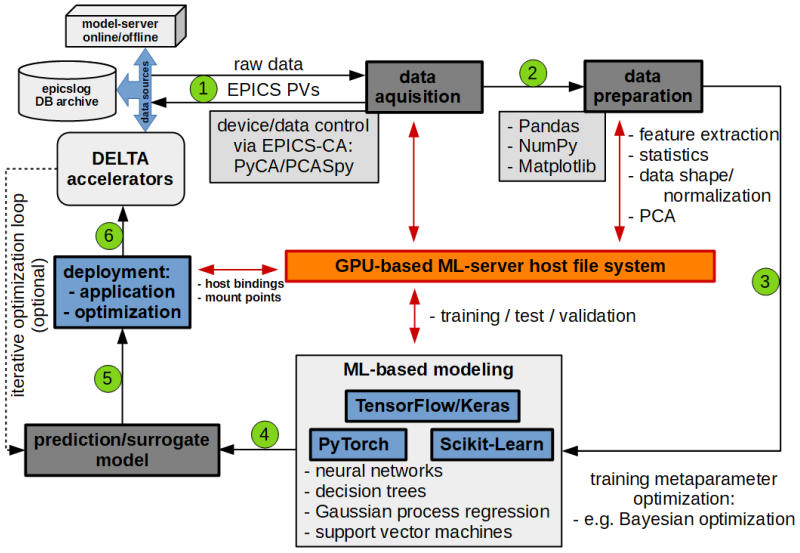

Figure 3. Concept of the EPICS/Python-based ML-workflow (steps 1 to 6). First, the ML-training data were collected from the accelerator facility (1). After data preparation (2), these data were used for offline model training (3) of surrogate injection models (4). Finally, the trained models, in turn, were loaded into the EPICS-based control system (5) and tested for injection optimization (6).

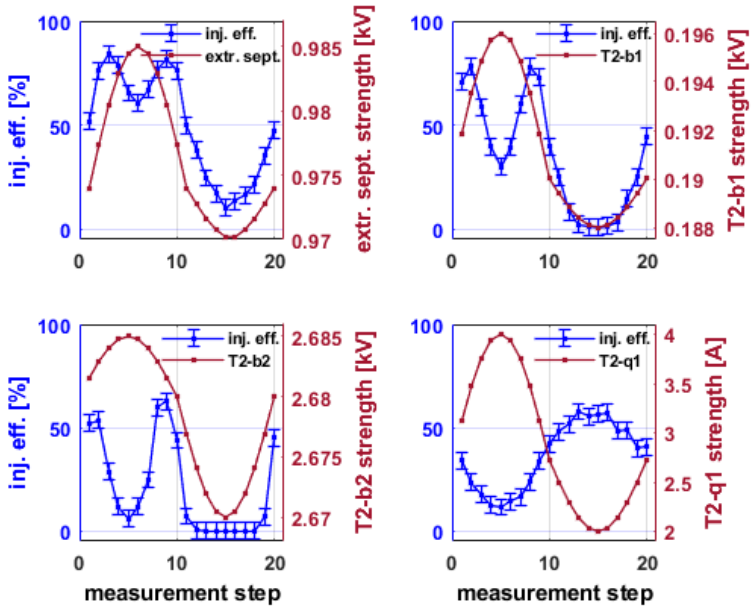

Figure 4. Step scanning of four T2 transfer line elements (exemplarily). The injection efficiency change was measured during sweeping the set values of the BoDo extraction septum (top left), the first (top right) and second (bottom left) pulsed dipole, and the first quadrupole (bottom right).

simulated annealing [5] and Bayesian optimization [6] as a predictive and computational basis for improving injection performance in real machine operation.

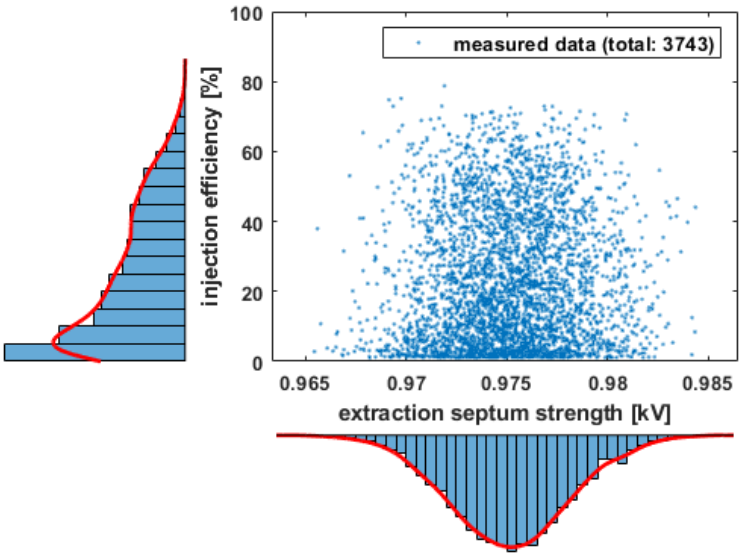

Figure 5. Example of Gaussian random variation of the BoDo extraction septum strength. At the same time, all other injection parameters were Gaussian-like changed in the same way. The resulting effect on the measured injection efficiency is depicted in the scatter plot and summarized in the histogram on the left.

3. Optimization results

As part of a diploma thesis [7], initially 13 parameters of the transfer line T2 were randomly as well as systematically varied and the corresponding impact on the injection efficiency was measured. These records served as input for supervised training of two types of surrogate injection models based on neural networks (NNs [8,9]) and Extremely Randomized Trees (Extra Trees [10]), a special case of Decision Trees (DTs [11]), respectively. A simulated annealing-based algorithm [5] utilizes the surrogate models to optimize injection parameters during the injection phase between each ramp cycle within 7 seconds. The hyper-parameters of both model types were optimized applying Bayesian optimization [5, 6]. This results in NNs with five fully connected ReLU (Rectifier Linear Unit) layers containing 13/30/25/34/1 neurons [7]. The NNs were trained using an ADAM back-propagation method [12], a stochastic gradient-based optimization. The Extra Trees were composed of an ensemble of 104 DTs with a maximum depth to grow of 26 [7]. First case studies showed that injection models based on trained NNs generally perform better than models based on Extra Trees. Four benchmark comparisons are exemplarily presented in Fig. 6. Starting with misadjusted injection parameter setups, the NN-based methods were able to achieve the model-predicted higher efficiencies after few iteration steps, whereby the Extra-Tree-driven optimization failed to improve the injection efficiency. These preliminary studies were continued in the scope of a subsequent master's thesis [13]. In this work, the injection parameter space was extended to 18 dimensions and the database for ML-based training was enlarged by supplementary DAQ-measurements. In addition to the T2 magnet strength settings, the injection elements of the storage ring (e. g., kicker magnets, magnets of a static injection bump) as well as trigger timings of the pulsed T2 and kicker magnets

were now taken into account, too. To train the surrogate injection models, in addition to NNs [9], also Gaussian Process Regressors (GPR [14, 15]) techniques were explored. Furthermore, a

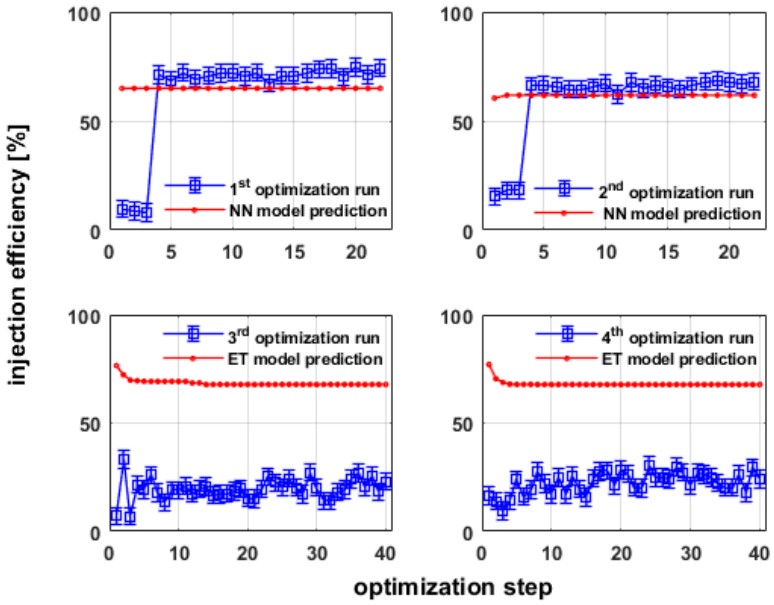

Figure 6. Four examples of injection efficiency optimization runs using neural networks (top: 1st and 2nd run) and Extra Trees (bottom: 3rd and 4th run) as surrogate models. The red lines indicate the prediction of the relating ML-trained model for each optimization step which corresponds to one injection ramp cycle.

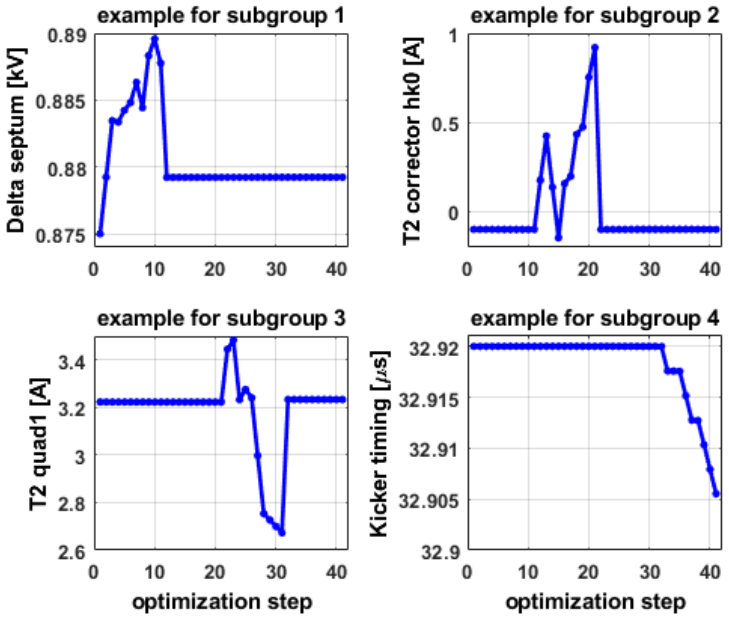

Figure 7. Variation of injection parameter set values during injection optimization scans (see Fig. 8) for one example from each of four subgroups of injection parameters. Step 1 provides a misadjusted starting point, and each subgroup is then optimized in a series of 10 steps.

Bayesian optimization using Gaussian processes (GP [13–16]) were performed for the injection optimization scans (see Fig. 8). Since GPs get by with less training data and are more efficient for lower–dimensional problems, the parameter space has been divided into subgroups of similar parameter types (e. g., quadrupole-, corrector- and kicker magnets, timings). Figure 7 depicts an example with four subgroups out of 18 available components. Figure 8 shows exemplarily the associated comparison of injection optimization scans applying the GPR- and NN-trained surrogate models, respectively. Both methods were able to find enhanced injection parameter sets to improve the efficiency significantly, generally with GPRs performing slightly better under this higher dimensional condition.

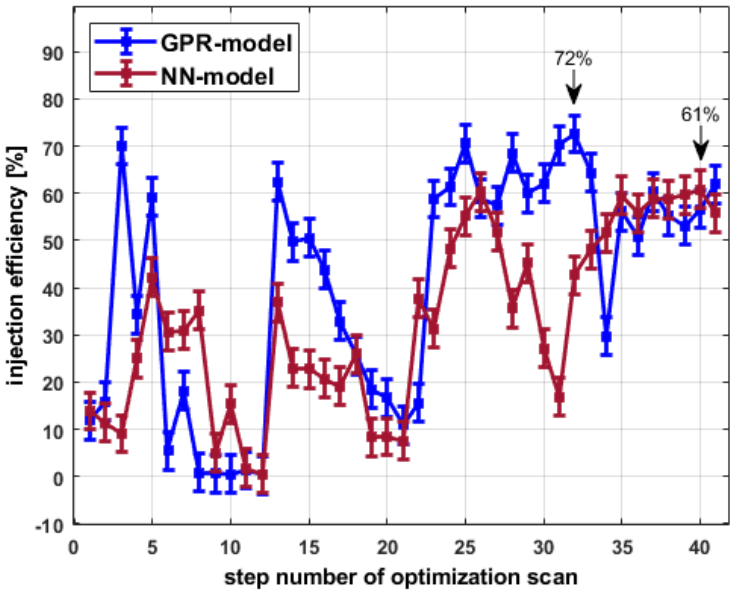

Figure 8. Injection efficiency optimization scans using Bayesian optimization and applying a Gaussian Process Regressor (GPR) as a surrogate injection model (blue) in comparison to an NN-trained model (red). The best parameter set of each scan is saved and can subsequently be applied to the machine. In this example, the GPR-based scan finds the best injection efficiency of 72% in step 32 compared to the maximum of 61% in step 40 by use of an NN-based model. Both runs started with a misadjusted efficiency setup of approx. 12%.

4. Summary and outlook

On a proof-of-principle level, it was demonstrated that ML methods are appropriate tools to optimize the injection process at the DELTA accelerator facility in an automated manner. In particular, trained NNs and GPRs were well-suited as predictive injection models in contrast to DTs techniques. In addition, it turns out that Bayesian optimization and simulated annealing algorithms using these surrogate models achieve similar improvements for injection performance in real machine operation. The results can be further improved by additional parameter optimizations of the ML algorithms and by systematic extension of the training data (e. g., T2-BPM data, T2-dipole temperatures). Furthermore, alternative optimization methods like MIDACO (Mixed Integer Distributed Ant Colony Optimization [17]), which are based on an evolutionary algorithm known as Ant Colony Optimization (ACO), will be tested. Finally, for better software maintenance, it is planned to implement the more universal ML-workflow steps (dark grey boxes in Fig. 3) into dedicated software containers [18, 19].

Acknowledgement
We would like to thank all colleagues of the DELTA team for providing sufficient DAQ time as well as for their helpful assistance in setting up the injection process.

References

[1] Büning T and Müller D 2005 *Diploma Thesis* Dortmund University Germany
[2] Schirmer D *et al* 2006 *Proc. of 10th European Particle Accelerator Conference (EPAC'06)* (Edinburgh, Scotland) pp 1948–1950
[3] Zimoch E 2003 *Ph.D. Thesis* Dortmund University Germany
[4] Schirmer D, Althaus A, Hüser S, Khan S, Schüngel T 2021 *Proc. 18th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'21)* (Shanghai, China) pp 631–35 (doi:10.18429/JACoW-ICALEPCS2021-WEPV007)
[5] scikit-optimize (Version 0.8.1) 2020 `https://scikit-optimize.github.io/stable/`
[6] Snoek J H, Larochelle H, Adams R P 2012 *Practical Bayesian Optimization of Machine Learning Algorithms* (`https://arxiv.org/abs/1206.2944`)
[7] Hüser S 2022 *Diploma Thesis* TU Dortmund University Germany
[8] Pedregosa F *et al* 2011 Scikit-learn: Machine Learning in Python *Journal of Machine Learning Research* **12** pp 2825–2830
[9] Keras/TensorFlow (Version 2.11.0) 2022 `https://www.tensorflow.org/api_docs/python/tf/keras`.
[10] Geurts P, Damien E and Wehenkel L 2006 *Extremely randomized trees* Machine Learning **63 No. 1** pp 3-42
[11] Breiman L, Friedman J, Olshen R and Stone C 1984 *Classification and regression trees* Boca Raton, USA (Florida) Chapman & Hall
[12] Kingma D P and Ba J L 2015 Adam: A method for stochastic optimization *3rd International Conference for Learning Representations (ICLR)* San Diego (`https://arxiv.org/abs/1412.6980`)
[13] Schüngel T 2022 *Master's Thesis* TU Dortmund University Germany
[14] scikit-learn (Version 0.24.2) 2021 `https://scikit-learn.org/`
[15] Rasmussen C E and Williams C 2005 *Gaussian processes for machine learning* (Cambridge, Mass.) MIT Press (ISBN 9780262182539)
[16] Xu C 2020 *Master's Thesis* Karlsruhe Institute of Technology (KIT) Germany
[17] Mixed Integer Distributed Ant Colony Optimization (MIDACO) 2018 (Version 6.0) `http://www.midaco-server.com`
[18] podman (Version 4.0.0) 2022 `https://podman.io/`
[19] docker (Version 20.10.16) 2022 `https://www.docker.com`