ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-TUPAB327

DEVELOPING ROBUST DIGITAL TWINS AND REINFORCEMENT LEARNING FOR ACCELERATOR CONTROL SYSTEMS AT THE FERMILAB BOOSTER

D. Kafkes*, Fermi National Accelerator Laboratory, Batavia, USA
M. Schram, Thomas Jefferson National Accelerator Facility, Newport News, USA

Abstract

We describe the offline machine learning (ML) development for an effort to precisely regulate the Gradient Magnet Power Supply (GMPS) at the Fermilab Booster accelerator complex via a Field-Programmable Gate Array (FPGA). As part of this effort, we created a digital twin of the Booster-GMPS control system by training a Long Short-Term Memory (LSTM) to capture its full dynamics. We outline the path we took to carefully validate our digital twin before deploying it as a reinforcement learning (RL) environment. Additionally, we demonstrate the use of a Deep Q-Network (DQN) policy model with the capability to regulate the GMPS against realistic time-varying perturbations.

BACKGROUND

Recently, the challenge and cost of hand-tuning and controlling accelerators has resulted in a push to leverage deep learning [1–4]. In this study, we present continuing work on a real-time artificial intelligence (AI) control system for precisely regulating the Gradient Magnet Power Supply (GMPS), an important subsystem of the Fermilab Booster accelerator complex.

The GMPS is realized as four power supplies, evenly distributed around the Fermilab Booster. Each powers one of four total gradient magnets, which are responsible for steering and accelerating the 400 MeV proton beam the Booster receives from the linear accelerator to 8 GeV [5, 6]. The GMPS operates on a 15 Hz cycle between the injection at minimum current and beam extraction at maximum current. Unfortunately, without any regulation, the fitted minimum of the magnetic field may vary from the set point by as much as a few percent, significantly reducing the beam flux available to experiments run at the lab [7]. This deviation trends with factors such as electrical ground movement, the operation of other nearby high-power radio-frequency systems, and even ambient temperature changes [5, 6].

In order to improve the agreement of the resulting observed minimum and maximum currents with their set points, a proportional-integral-derivative (PID) control scheme applies compensating offsets to the GMPS driving signal as a means of regulation (see Fig. 1) [8, 9]. Presently, a human operator specifies a target program for `B:VIMIN` and `B:VIMAX`, the PID-GMPS compensated minimum and maximum currents respectively, via the Fermilab Accelerator Control Network. This signal is then transmitted to the GMPS control board allowing the PID regulator to use the previous 15 Hz cycle to calculate estimates for the minimum and maximum current offset and then uses these values to adjust the power supply program in the current cycle [8, 9].

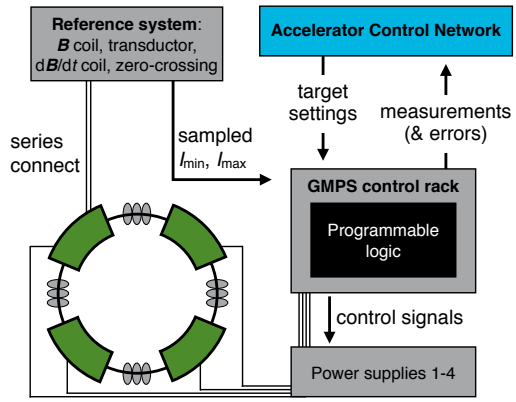

Figure 1: Overview of current PID-GMPS control system.

Presently, this PID-GMPS regulation system achieves errors corresponding to roughly 0.1% of the set value [7]. Our ultimate goal is to improve on this error by replacing the PID-GMPS system with a reinforcement learning (RL) approach. This RL-GMPS system will leverage a framework in which an artificial intelligence (AI) agent learns to achieve some end goal through feedback from interactions with its environment [10, 11]. We aim to deploy such an RL agent to control changes in `B:VIMIN` in order to minimize `B:IMINER`, the difference between the observed minimum current reading and setting. This agent will ultimately control the GMPS system via a field-programmable gate array (FPGA). However, since training the agent online involves substantial risk, we prototyped our models offline.

METHODS

We collected approximately six months of time series data from the Booster complex and ultimately selected 250,000 continuous time steps from March 10, 2020 to develop our preliminary algorithms [7]. Here algorithms refer to both the methods used to build a model that can reliably capture the dynamics of the Booster-GMPS system and to develop a reinforcement learning framework. For complete details on our data collection process and access to our full published dataset please see [12].

Digital Twin and RL Policy Models

Our offline ML development involved the training of two different neural network models: a surrogate or digital twin model and the RL agent policy model. Here we implement

* dkafkes@fnal.gov

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2021). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-TUPAB327

a simple multi-layer perceptron (MLP) as our agent policy model (see Table 1), the utility of which will be further described in the next subsection, and a stacked Long-Short Term Memory (LSTM) network to capture GMPS dynamics [7].

Table 1: DQN-MLP Policy Model

Layer	Layer Type	Outputs	Activation	Parameters
1	Dense	128	ReLU	768
2	Dense	128	ReLU	16,512
3	Dense	128	ReLU	16,512
4	Dense	7	Linear	903
Total				34,695

MLPs are standard feedforward neural networks which take in and iteratively feedforward data through many layers of perceptrons (neurons). Each of these neurons involves a function that multiplicatively weights the input vectors, sums them together, adds a bias term; and then applies a nonlinear activation function. The chaining of these functions results in a network.

Beyond MLPs, there are many possible choices of neural network architectures. The one most relevant to capturing the Booster-GMPS system's multiple frequency modalities in our surrogate model is a type of network known as a Long-Short Term Memory or LSTM (see Table 2) [7]. Unlike standard feedforward networks, during each forward pass, LSTMs are able to learn about previous inputs through the accumulation of weights in a hidden global state variable. This mechanism is useful for modeling time series data, which is exactly what we had collected for this effort.

Table 2: Stacked LSTM Digital Twin

Layer	Layer Type	Outputs	Activation	Parameters
1	LSTM	256	Tanh	416,768
2	LSTM	256	Tanh	525,312
3	LSTM	256	Tanh	525,312
4	Dense	3	Linear	771
Total				1,468,163

DQN Reinforcement Learning

Reinforcement learning is a training framework that involves an AI agent interacting with an environment to maximize a defined reward over many fixed-iteration-length episodes [10, 11]. The agent's actions within this environment are defined by a policy model. As stated above, we used an MLP for this policy model in accordance with the deep Q-network (DQN) approach, which trains this neural network to learn the action-value function— Q-value— that maps a discrete number of agent actions to rewards [13, 14]. To keep our action space finite, we discretized the *change* of `B:VIMIN` using steps of just seven different sizes, including an option for zero-size change [7].

At each time step t, our surrogate model environment takes in the control action A_t determined by the RL agent MLP-DQN policy model as the small compensation to be applied to `B:VIMIN` based on the current state S_t. The digital twin then provides the new system state S_{t+1} along with an associated reward R_{t+1}. In our studies, the state is composed of the variables inputted to the surrogate model (discussed at length in the section below) and the reward is calculated from the `B:IMINER` output by the surrogate:

$$R_t = -|\texttt{B:IMINER}(t)|\,. \tag{1}$$

Optimizing the agent's policy actions over the training horizon is defined to mean maximizing the long-term integrated reward, which is calculated over each fixed-length episode.

DIGITAL TWIN DEVELOPMENT AND VERIFICATION

In our preliminary result [7], we formulated our stacked LSTM surrogate model to capture the dynamics of:

$$\begin{aligned}\texttt{B:VIMIN} + \texttt{B:IMINER}+\texttt{B:LINFRQ} + \texttt{I:IB} + \texttt{I:MDAT40}\\ \rightarrow \texttt{B:VIMIN}+\texttt{B:IMINER} + \texttt{B:LINFRQ}.\end{aligned} \tag{2}$$

Here `B:LINFRQ` is the measured offset from the 60 Hz line frequency, and `I:IB` and `I:MDAT40` provide measurements of the main injector bending dipole current through different communication channels. This model was trained using MinMax scaling [15] and a 150 step lookback, i.e. 150 previous timesteps of the input variables were fed into the model in order to predict the next timestep forward in the output variables [7]. Here we describe the validation process we used to verify and improve upon our initial result.

First, we distilled our surrogate model into the simplest possible combination of variables we aimed to regulate: `B:VIMIN` + `B:IMINER` → `B:VIMIN` + `B:IMINER`. From this most basic model, we explored using a much smaller lookback window of 15 timesteps as well as the use of Robust scaling [15]. Since we found the 15 timestep lookback and MinMax scaling to be performant, we ultimately decided to keep the original scaling and move forward to experiment with this much shorter lookback window.

After iterating over this most basic model, we began the forward selection process, experimenting with the inclusion of different variables. The variables we considered for these studies included the inputs from the original model: `B:VIMIN`, `B:IMINER`, `B:LINFRQ`, `I:IB`, and `I:MDAT40`; as well as `B_VIMIN` (the GMPS minimum current set point), `B:VIMAX` (the compensated maximum GMPS current), `B:VIPHAS` (the GMPS ramp phase with respect to line voltage), and `I:MXIB` (the main injector dipole bend current). `B_VIMIN` was included on the suggestion of subject-matter experts [16]; and the three additional variables were selected based on the results of a Granger Causality study [17]. For details on how this analysis was performed, please see [7].

The results of these surrogate model experiments are presented in Table 3 below, which displays the final loss attained by the model with the given configuration. After comparing

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2021). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-TUPAB327

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2021). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

results, we decided to move forward with the 6 to 2 model since including the other three variables in the 9 to 2 model made only a slight difference in the final loss:

$$\texttt{B:VIMIN} + \texttt{B:IMINER} + \texttt{B\_VIMIN} + \texttt{B:LINFRQ} + \texttt{I:IB} + \texttt{I:MDAT40} \rightarrow \texttt{B:VIMIN} + \texttt{B:IMINER}. \quad (3)$$

Table 3: Training Mean-Squared Error (MSE)

Model	MSE (10^{-6})
B:VIMIN + B:IMINER → B:VIMIN + B:IMINER	449.0567
B:VIMIN + B:IMINER + B_VIMIN → B:VIMIN + B:IMINER	379.5542
B:VIMIN + B:IMINER + B:LINFRQ + I:IB + I:MDAT40 → B:VIMIN + B:IMINER	346.6192
B:VIMIN + B:IMINER + B_VIMIN + B:LINFRQ + I:IB + I:MDAT40 → B:VIMIN + B:IMINER	314.3544
B:VIMIN + B:IMINER + B_VIMIN + B:LINFRQ + B:VIMAX + B:VIPHAS + I:IB + I:MDAT40 + I:MXIB → B:VIMIN + B:IMINER	294.6336

Additionally, we tried decomposing the variables from the models mentioned above into signal and noise vectors using Empirical Mode Decomposition [18]. Despite the fact that this resulted in marginally better performance, we decided that this would be too difficult to implement in real-time on the board funneling input data to the FPGA. For this reason, we omit these results here. Similarly, after completing the digital twin validation and verification process, we decided to create our own version of the MinMax scaler rather than using the transformation available to us via the scikit-learn library since using this premade scaler could not be easily implemented on the FPGA-side.

Uncertainty Quantification

In order to provide a prediction with statistical interoperability, we performed concrete dropout as a means of uncertainty quantification [19]. The concrete dropout process involves introducing tunable uncertainty into a network training process through the addition of a dropout layer, which randomly removes inputs to the following layer with some probability p at each forward pass. This causes the training of the network's other weights and biases to adjust without these "dropped out" neurons. Once this dropout layer has been added to the network, p can be adjusted to take on a different value during inference.

After training our surrogate with a dropout layer inserted after the first LSTM, we set the layer to probabilities ranging from $[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.5]$, and used the inferred outputs of our network to match the actual underlying distribution of the data. We found a value of $p = .2$ gave us our best results: reconstructing the modeled distribution of B:VIMIN at 103.3930 ± 0.0297 (underlying distribution: $103.3940 \pm .0314$); and B:IMINER at 0.0012 ± 0.2090 (underlying distribution of $.0011 \pm .2181$).

PRELIMINARY RL RESULTS

Finally, we present our most recent RL results, training and deploying our trained MLP-DQN policy model within our verified digital twin environment in Fig. 2. When comparing the DQN-GMPS system results to the PID-GMPS controller, we see a factor of 2-4x improvement.

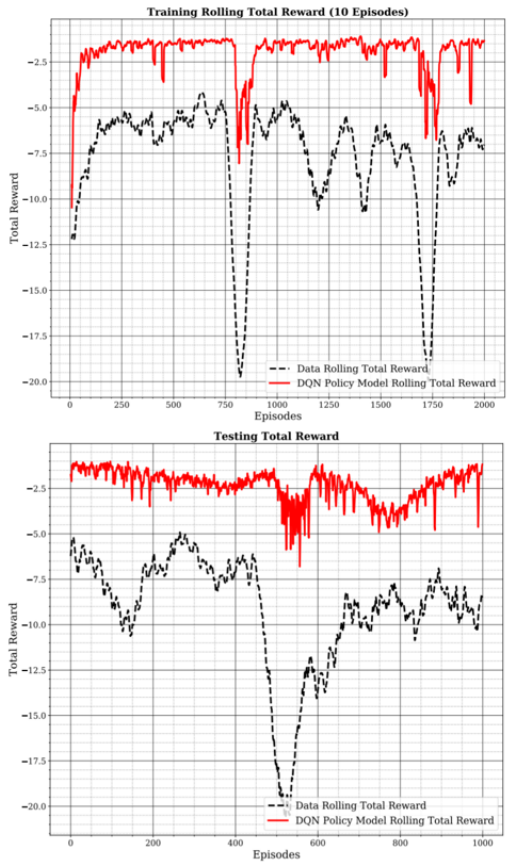

Figure 2: Results of training (top) and testing (bottom).

CONCLUSION

We outlined the steps we took to carefully validate our digital twin of the Booster-GMPS system— perhaps the most important aspect of our offline machine learning development. After all, without a robust surrogate model to support training, we would not be able to trust the deployment of the trained agent on the live system via an FPGA in the future.

ACKNOWLEDGEMENTS

This research is conducted through the Fermilab Directed Research and Development Program "Accelerator Control with Artificial Intelligence" Project (Project ID *FNAL-LDRD-2019-027*, under Contract No. DE-AC02-07CH11359), and is registered at Fermilab as Technical Report Number *FERMILAB-CONF-21-230-AD-SCD*.

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-TUPAB327

REFERENCES

[1] A. Edelen *et al.*, "Neural Networks for Modeling and Control of Particle Accelerators", *IEEE Trans. Nucl. Sci.*, vol. 63, no. 2, p. 878, Apr. 2016. doi:10.1109/TNS.2016.2543203

[2] A. L. Edelen, S. Biedron, S. V. Milton, and J. P. Edelen, "First Steps Toward Incorporating Image Based Diagnostics into Particle Accelerator Control Systems Using Convolutional Neural Networks", in *Proc. NAPAC'16*, Chicago, IL, USA, Oct. 2016, pp. 390-393. doi:10.18429/JACoW-NAPAC2016-TUPOA51

[3] A. L. Edelen, S. Biedron, S. V. Milton, and P. J. M. van der Slot, "Using A Neural Network Control Policy For Rapid Switching Between Beam Parameters in an FEL", in *Proc. FEL'17*, Santa Fe, NM, USA, Aug. 2017, pp. 488-491. doi:10.18429/JACoW-FEL2017-WEP031

[4] J. Duris *et al.*, "Bayesian optimization of a free-electron laser", *Phys. Rev. Lett.*, vol. 124, no. 12, p. 124801, 2020. doi:10.1103/PhysRevLett.124.124801

[5] *Booster Rookie Book Manual v4.1*, J. Crawford *et al.*, Fermilab, Batavia, IL, USA, 2009. https://operations.fnal.gov/rookie_books/Booster_V4.1.pdf

[6] J. Ryk, "Gradient Magnet Power Supply for the Fermilab 8-GeV Proton Synchrotron", Fermilab, Batavia, IL, USA, Rep. FERMILAB-PUB-74-085, Aug. 1984.

[7] J. St. John *et al.*, "Real-time Artificial Intelligence for Accelerator Control: A Study at the Fermilab Booster", submitted for publication.

[8] N. Minorsky, "Directional Stability of Automatically Steered Bodies", *J. Am. Soc. Nav. Engineers*, vol. 34, no. 2, p. 280, 1922. doi:10.1111/j.1559-3584.1922.tb04958.x

[9] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers", *J. Dyn. Sys., Meas., Control.*, vol. 64, p. 759, 1942.

[10] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA, USA: MIT Press, 2018.

[11] V. François-Lavet *et al.*, "An Introduction to Deep Reinforcement Learning", *Found. Trends Mach. Learn.*, vol. 11, p. 219, 2018. doi:10.1561/2200000071

[12] D. Kafkes and J. St. John, "BOOSTR: A Dataset for Accelerator Control Systems", *Data*, vol. 24, no. 6, p. 124801, 2021. doi:10.3390/data6040042

[13] V. Mnih *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, p. 219, 2015. doi:10.1038/nature14236

[14] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning", present at the NIPS Deep Learning Workshop, Lake Tahoe, NV, USA, Dec. 2013, unpublished.

[15] F. Pedregosa, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, p. 2825, 2011. doi:10.5555/1953048.2078195

[16] R. Keller, "Controlling Currents", Fermilab, Batavia, IL USA, Aug. 2019.

[17] C. Granger, "Investigating Causal Relations By Econometric Models and Cross-spectral methods", *Econometrica*, vol. 37, p. 424, 1969. doi:10.2307/1912791

[18] J. Gao, S. Haghighi, and D. Hatzinakos, "Reference empirical mode decomposition", in *2014 IEEE 27th Canadian Conf. on Electrical and Computer Engineering (CCECE)*, Toronto, Canada, May. 2014, pp. 1–4. doi:10.1109/CCECE.2014.6900938

[19] Y. Gal, J. Hron, and A. Kendall, "Concrete Dropout", in *Proc. 31st Conf. on Neural Information Processing Systems (NIPS'17)*, Long Beach, CA, USA, Dec. 2017, pp. 1-10.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2021). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI