Using Neural Network Control Policies For Rapid Switching Between Beam Parameters in a Free Electron Laser

Auralee L. Edelen
Fermilab and Colorado State University
Batavia, IL and Fort Collins, CO
auralee.l.morin@gmail.com

Jonathan P. Edelen
RadiaSoft, LLC
Boulder, CO

Sandra G. Biedron
University of New Mexico
Albuquerque, NM

Stephen V. Milton
Los Alamos Natioanl Laboratory
Los Alamos, NM

Peter J.M. van der Slot
University of Twente
Enschede, The Netherlands

Abstract

Free Electron Laser (FEL) facilities often must accommodate requests for a variety of electron beam parameters in order to supply scientific users with appropriate photon beam characteristics. This usually requires skilled human operators to tune the machine. In principle, a neural network control policy that is trained on a broad range of machine operating states could be used to quickly switch between these requests without substantial need for human intervention. We present preliminary results from an ongoing simulation study in which a neural network control policy is investigated for rapid switching between beam parameters in a compact THz FEL that exhibits nonlinear electron beam dynamics. To accomplish this, we first train a feed-forward neural network to mimic a physics-based simulation of the FEL. We then train a neural network control policy by first pre-training it as an inverse model (using supervised learning with a subset of the simulation data) and then training it more extensively with reinforcement learning. In this case, the reinforcement learning component consists of letting the policy network interact with the learned system model and backpropagating the cost through the model network to the controller network.

1 Introduction and Overview

Free Electron Laser (FEL) facilities support a broad range of scientific endeavors, for example elucidating protein structures [1], investigating natural processes such as photosynthesis [2], and understanding the origin of different material properties [3]. In support of this, these facilities must accommodate requests for a variety of electron beam parameters in order to supply their users with the appropriate photon beam characteristics for any given experiment. This usually requires skilled human operators to tune the machine, thus reducing the amount of useful experimental time and increasing the overall operating cost of the facility relative to its scientific output. In principle, a neural network control policy that is trained on a broad range of operating states could be used to quickly switch between these requests without substantial need for human intervention (see [4] for more general discussion on possible applications of neural networks to modeling and control of particle accelerators).

We are exploring this approach using simulations of a compact THz FEL design that is based on the Twente/Eindhoven University FEL (TEU-FEL) [5]. This an appealing system for this initial study

Workshop on Deep Learning for Physical Sciences (DLPS 2017), NIPS 2017, Long Beach, CA, USA.

because it has a small number of machine components, yet it exhibits non-trivial beam dynamics. In this case, changing the operating state consists of specifying a change in the electron beam energy and then choosing the appropriate injector and beamline settings such that the electron beam is properly matched into the undulator. Here, we first give an overview of the FEL system. This is followed by a discussion of our initial study, in which the aim was to choose the quadrupole magnet settings such that specific Twiss parameters, $\alpha_{x,y}$ and $\beta_{x,y}$ (which yield approximations of the beam ellipse in position-momentum phase space and correspond to the beam tilt and size/shape, respectively), are achieved for arbitrary requested beam energies within the range of 3–6 MeV. We then briefly discuss our current efforts on a more comprehensive study.

2 FEL Description and Physics-based Simulation

The FEL we chose to use as our template is designed to produce light with a wavelength that is tunable between 200 μm and 800 μm. It consists of a 5.5-cell, 1.3-GHz photocathode RF gun, a beam transport section that consists of 5 quadrupoles (Q1–Q5), and a fixed-gap THz undulator. The photoinjector is an RF structure that accelerates and focuses the initial electron beam. A bucking coil and solenoid provide appropriate focusing in the photoinjector. More details on this accelerator design can be found in [5], and Fig. 1 shows the relevant components for this study. In order to achieve proper matching into the undulator, the beam is focused to a waist at the entrance, which means the alpha parameter $\alpha_{x,y} = 0$. We chose to find settings for the quadrupoles such that the same beta value ($\beta_{x,y}$) at the entrance of the undulator for each electron beam energy would be achieved: $\beta_{x,y} = 0.106$ [m/rad]. This corresponds to a beam size that is within the optimal field region of the undulator for all energies.

While this machine is comparatively simple in terms of the number of components, the behaviour of the electron beam involves some subtleties that if not addressed properly will result in decreased performance of the FEL. In particular, the beam evolution throughout the machine is nonlinear. Because the bunch charge can be as large as 5 nC and the beam energy is low (3–6 MeV), space-charge effects (i.e. interactions between electrons within the beam) are significant throughout the system. This complicates the task of focusing the beam, which in turn complicates the task of matching the beam into the undulator.

A physics-based simulation of the machine was constructed using SUPERFISH [6] for the RF fields and PARMELA [7] for the beam dynamics. The solenoid and bucking coil assembly were modelled using PANDRIA [6] using the nominal current setting in the solenoid. The bucking coil was scaled to cancel the magnetic field on the cathode. The combined field map is then scaled in PARMELA in order to tune the space-charge compensation. The PARMELA simulations were performed using 5000 macro-particles with a 0.1° phase integration step. This was determined to be well within the stable region for reasonable estimation of bulk beam parameters [8].

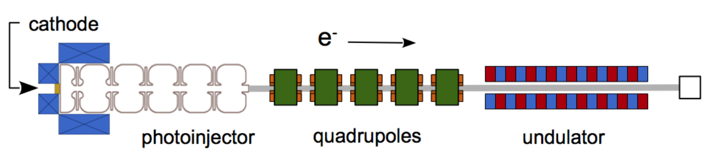

Figure 1: Layout of the accelerator showing the 5.5-cell photoinjector with its bucking coil and solenoid, the cathode, the quadrupoles, the undulator, and the beam dump.

3 Neural Network Model

First, we train a neural network model to mimic the physics-based simulation of the machine. This creates a surrogate for the full simulation that captures the relevant behaviour of the machine and can execute quickly to facilitate controller training. Similar approaches to creating fast-executing stand-ins of simulations have been used for other accelerator systems [9, 10] and in other fields, including simulation of particle detectors [11] and cosmological processes [12].

The model was trained to predict the Twiss parameters, beam energy, emittance, and transmission at the entrance of the undulator, given the RF power, the RF phase, the solenoid strength, and the quadrupole settings (see Fig. 2). Because the calculated beam parameters become inaccurate when a smaller number of particles remains in the transmitted bunch, we have two ouput layers: one for transmission, which is trained on the full data set, and one for the beam parameters, which is only trained when the transmission is above 90%. We wanted to mimic what one might find in measured data from an FEL facility (i.e. noisy data with tuning around roughly optimal settings). As such, the training set consisted of the output from each iteration of an optimization routine (in this case, Nelder-Mead [13]) to find optimal quadrupole settings for 12 different beam energies between 3.1 and 6.2 MeV, with noise added (resulting in 7195 samples total). Data for the 5.7-MeV electron beam was excluded from the training set and used as the validation set (310 samples total). Table 1 shows the range of each parameter used in the training data.

The model network consists of four hidden layers containing 50, 50, 30, and 30 nodes, respectively. Each node in the hidden layers uses a hyperbolic tangent activation function and a dropout [14] probability of 10%. The weights and biases of the network were trained using the AdaMax [15] optimization algorithm, along with subsequent fine-tuning using Møller's scaled conjugate gradient [16] algorithm. The cost was the mean squared error over the predictions for each batch of 200 samples. The network was constructed and trained using a combination of lasagne [17] and Theano [18]. The performance of the model in terms of mean absolute error (MAE) and standard deviation (STD) is shown in Table 2. A representative plot from the validation set is shown in Figure 3.

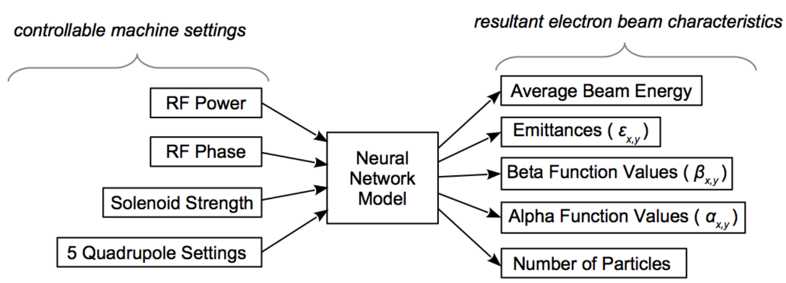

Figure 2: General setup for the neural network model. In the controller, the inputs and outputs are opposite those shown here.

Table 1: Maximum and minimum ranges of parameters in the training data

Parameter	α_x [rad]	α_y [rad]	β_x [m/rad]	β_y [m/rad]	E [MeV]
Max Value	2.11	1.45	1.86	3.65	6.2
Min Value	-2.14	-5.76	0.06	0.07	3.1
Parameter	Q1 [T/m]	Q2 [T/m]	Q3 [T/m]	Q4 [T/m]	Q5 [T/m]
Max Value	0.83	1.98	-1.07	2.26	-0.23
Min Value	-0.98	0.65	-2.24	0.89	-1.9
Parameter	RF Power [norm.]	Solenoid [norm.]	RF Phase [deg]	Transmission	
Max Value	1.15	1.05	21.4	100%	
Min Value	0.57	0.67	10.3	0	

Table 2: Model Performance

Parameter	Train MAE	Train STD	Train Max	Val. MAE	Val. STD	Val. Max
α_x [rad]	0.018	0.042	0.590	0.067	0.091	0.482
α_y [rad]	0.022	0.037	0.845	0.070	0.079	0.345
β_x [m/rad]	0.004	0.009	0.287	0.008	0.012	0.130
β_y [m/rad]	0.005	0.011	0.357	0.012	0.017	0.189

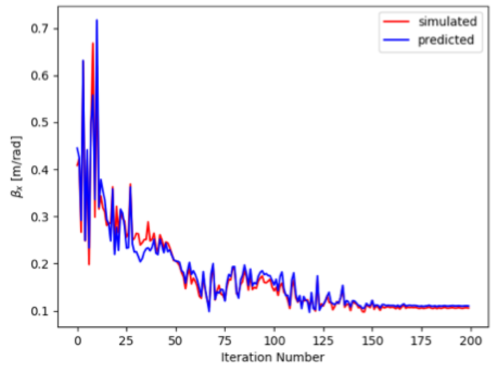

Figure 3: As an example, we show neural network model predictions and simulated values on the validation data set, with a beam energy of 5.7 MeV. This shows that we have created a reasonably accurate, fast-executing surrogate model as a stand-in for the physics-based model.

4 Neural Network Controller

The controller selects appropriate quadrupole settings for a desired beam energy. RF power, phase, and solenoid strengths were changed only to facilitate the change in beam energy, and their corresponding network outputs were ignored in this initial study. The controller was first trained as an inverse model following the same procedure as the forward model. We then trained the controller network more extensively via interaction with the model network, the output from which was used to directly calculate the cost. Here we took the naïve approach of back-propagating the cost through the model network. As such, we've trained a deterministic control policy first by pre-training with supervised learning (on our data from simulation) and then continuing training with reinforcement learning (by interacting with the learned model).

Random desired beam energy values between 3.1 and 6.2 MeV were specified, along with a target set of α and β parameters, as inputs to the controller. The cost function included the mean squared error between these target parameters and those predicted by the model. In addition, losses of full transmission and larger quadrupole settings were penalized proportionally. Requested beam energy values between 4.8 and 5.2 MeV were excluded from training and used exclusively for validation. The controller network consisted of three hidden layers, with 30, 30, 20, and 20 nodes, respectively. As before, each hidden node used a hyperbolic tangent activation function and a dropout probability of 10%, and the batch size was 200 samples. This time, only the AdaMax algorithm was used for optimizing the network weights.

Finally, interaction directly with the physics-based simulation was used to verify the performance of the controller. Given random requested energy values within 3–6 MeV, Table 3 shows the performance in reaching the desired Twiss parameters ($\alpha_{x,y}$ = 0 rad, = $\beta_{x,y}$ = 0.106 m/rad) in one iteration for energies in the range of the testing set and validation set. This shows that for a given energy, the controller will immediately reach the desired beam size to within about 10% and the beam will be close to a waist, requiring minimal further tuning to reach the target values.

Table 3: Ability to achieve $\alpha_{x,y}$ = 0 rad, $\beta_{x,y}$ = 0.106 m/rad for 3–6 MeV beams in one iteration

Parameter	Train MAE	Train STD	Train Max	Val. MAE	Val. STD	Val. Max
α_x [rad]	0.012	0.075	0.011	0.046	0.063	0.141
α_y [rad]	0.013	0.079	0.012	0.045	0.064	0.140
β_x [m/rad]	0.008	0.004	0.006	0.006	0.023	0.008
β_y [m/rad]	0.014	0.011	0.011	0.011	0.069	0.038

5 Current and Future Work

Based on results from the initial study, we have been conducting a more comprehensive version of it that also includes the task of minimizing the emittance at the entrance of the undulator. As this requires finer adjustment of the RF phase, RF power, and solenoid strength in conjunction with the quadrupole settings, this time the controller takes in desired beam energy, Twiss parameters, emittances, and transmission at the entrance of the undulator and sets the RF power, RF phase, solenoid strength, and quadrupole settings needed to achieve them. For this study, we are also doing more comprehensive testing (e.g. introducing drift and disturbances into the requested vs. actual machine settings, letting the controller adapt to newly observed parameter ranges and act over multiple iterations, etc.).

6 Conclusions

We have shown encouraging results from an initial simulation-based beamline tuning study, indicating that in one iteration the neural network can set the quadrupoles to achieve close to the correct Twiss parameters for arbitrary beam energies between 3–6 MeV. This is a first step toward the development of a neural network controller that will facilitate fast switching between operational parameters along with fine-tuning for the full machine. Our next step is to finish the full photoinjector and beamline study. While the true merit of the approach won't be clear until it is tested experimentally, plans are underway for this and we can optimistically say that a neural network control policy may very well be an expedient way of switching between operating states in an FEL.

References

[1] J.-P. Colletier, et al.,"De novo phasing with X-ray laser reveals mosquito larvicide BinAB structure," *Nature*, vol. 539, pp. 43–47, Sep. 2016.

[2] I. D. Young, et al., "Structure of photosystem II and substrate binding at room temperature," *Nature*, vol. 540, pp. 453–457, Nov. 2016.

[3] M. P. Jiang, et al., "The origin of incipient ferroelectricity in lead telluride," *Nature Communications*, vol. 7, no. 12291, Jul. 2016.

[4] A. L. Edelen, S.G. Biedron, B.E. Chase, D. Edstrom, S.V. Milton, P. Stabile, "Neural Networks for Modeling and Control of Particle Accelerators," *IEEE Transactions on Nuclear Science*, vol. 63, no. 2, pp. 878–897, Apr. 2016.

[5] G. J. Ernst, et al., "Status of "TEU-FEL" Project," *Nuclear Instruments and Methods A*, vol. 318, no. 1–3, 1992.

[6] J. Billen, *Poisson Superfish Codes*, Los Alamos National Laboratory, Copyright 1985–2005.

[7] L. Young, *PARMELA Codes*, Los Alamos National Laboratory, Copyright 1985–2005.

[8] J. P. Edelen, "Design and Simulation of the Colorado State Univeristy Linear Accelerator System," M.S. Thesis, Colorado State University, Fort Collins, CO, 2014.

[9] A. L. Edelen, et al., "First Steps Toward Incorporating Image-based Diagnostics Into Particle Accelerator Control Systems using Convolutional Neural Networks," *Proc. of the North American Particle Accelerator Conference 2016*, Chicago, IL, Oct. 9–11, 2016.

[10] A. Adelmann, "On Uncertainty Quantification in Particle Accelerator Modelling," *arXiv preprint*, arXiv:1509.08130, 2016.

[11] L. de Oliviera, M. Paganini, and B. Nachman, "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis," *Computing and Software for Big Science*, 1:4, 2017.

[12] H. M. Kamdar, et al., "Machine Learning and Cosmological Simulations II: Hydrodynamical Simulations," *Monthly Notices of the Royal Astronomical Society*, vol. 457, no. 2, pp. 1162–1179, 2016.

[13] J. A. Nelder and R. Mead, “A simplex method for function minimization”, *Comput. J.*, vol. 7, pp. 308–313, 1965.

[14] N. Srivastava, et al., “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[15] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint*, arXiv:1412.6980, 2014.

[16] M. Moller, “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural Networks*, vol. 6, pp. 525–533, 1993.

[17] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri et al., "Lasagne: First release," Aug. 2015, doi: 10.5281/zenodo.27878.

[18] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, arXiv: 1605.02688, 2016.