ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUPA34

MODEL-BASED CALIBRATION OF CONTROL PARAMETERS AT THE ARGONNE WAKEFIELD ACCELERATOR*

I. Sugrue†[1], P. Piot[1], N. Krislock, Northern Illinois University, DeKalb, IL 60115, USA
[1] also at Argonne National Laboratory, Lemont, IL 60439, USA
B. Mustapha, J. Power, Argonne National Laboratory, Lemont, IL 60439, USA

Abstract

Particle accelerators utilize a large number of control parameters to generate and manipulate beams. Digital models and simulations are often used to find the best operating parameters to achieve a set of given beam parameters. Unfortunately, the optimized physics parameters cannot precisely be set in the control system due to, e.g., calibration uncertainties. We developed a data-driven physics-informed surrogate model using neural networks to replace digital models relying on beam-dynamics simulations. This surrogate model can then be used to perform quick diagnostics of the Argonne Wakefield accelerator in real time using nonlinear least-squares methods to find the most likely operating parameters given a measured beam distribution.

INTRODUCTION

Small-scale accelerator facilities supporting accelerator R&D have often limited diagnostics with operating parameters not precisely calibrated. The present research attempts to develop a digital twin model of the Argonne Wakefield Accelerator (AWA) where operating parameters dialed in the control system (with calibration errors) are calibrated against their physical values inferred from a Physics-informed surrogate model [1]. In this paper we discuss the development of a surrogate model using the Object-Oriented Parallel Accelerator Library (OPAL) which simulates the beam dynamics in the AWA beamline [2]. Our initial focus is to develop a digital twin of the beam-generation and acceleration section diagrammed in Figure 1. The section consists of an RF-gun followed by a linac (L1). The beamline incoporates 3 solenoid magnets (BF, M, and LS1). This required training a neural network to predict the outputs of 9 control parameters (the solenoid-magnet BF and M currents, the laser spot size on the photocathode, the phase and amplitudes of the RF gun and L1, and the transverse misalignment of L1) in the form of an image representing the transverse beam distribution captured at the scintillating screen YAG1. The goal was to gather data from OPAL, and train a network to predict the output that OPAL produced. After doing so, the same network was to be applied to the accelerator itself so that given an image one can infer the values of the 9 control parameters . Many mathematical techniques were used to aid in the creation and training of the neural network, which will be described in subsequent sections of this paper.

* This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under contract DE-AC02-06CH11357 with ANL.
† isugrue@anl.gov

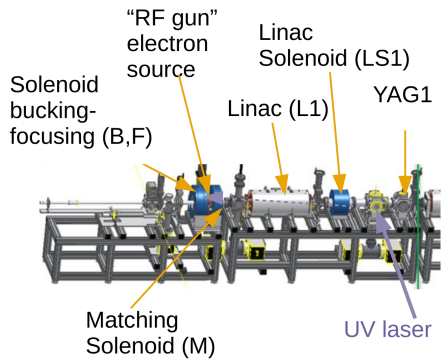

Figure 1: Overview of the AWA RF-gun and first linac considered for our surrogate model. The beam propagates from left to right.

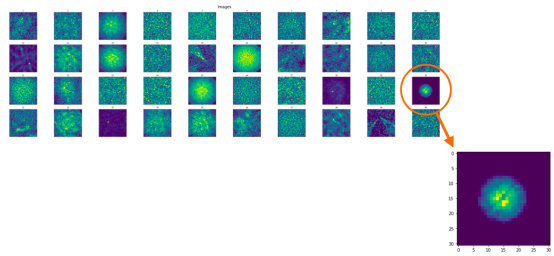

Figure 2: Unfiltered data set of YAG images. A good image was identified and isolated.

DATA FILTERING

Inputs of the Neural Network are 9 control parameters as vectors in $\mathbb{R}^9$, and our outputs are YAG images, uploaded as 32×32 real-valued matrices. The data gathered from OPAL was initially randomly generated. However, this resulted in "unusable" data, and much of it had to be discarded. The useful data was identified based on whether the beam fit entirely within the dimensions of the generated image. To sort the data based on this criteria, we displayed a collection of images and searched for one that matched this description. Once this image was identified (see Figure 2), the first 400 images that were closest in norm were selected. We also rotated each image 90 degrees and copied it to make the dataset larger (size 1600) for training.

PRINCIPAL COMPONENT ANALYSIS

The application of Principal Component Analysis (PCA) to our images was inspired in part by [3]. To begin, we have a collection of 1600 matrices (images), each with dimension 32×32. We subtract the mean image from each image to center the data, then flatten each image so that we have

ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUPA34

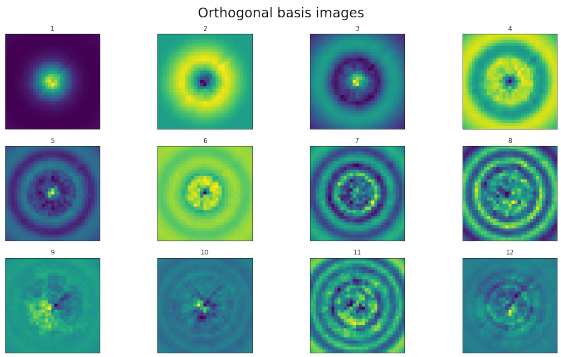

Figure 3: The first twelve orthogonal basis vectors as a result of Principal Component Analysis.

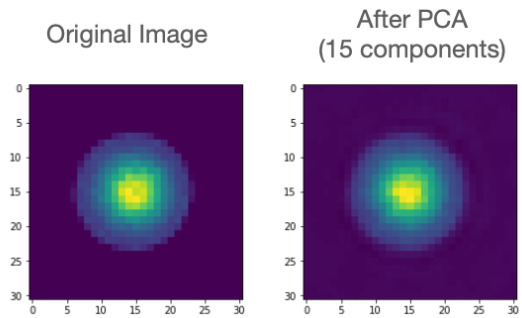

Figure 4: The original image (left) compared to the image recreated with 15 principal components (right).

a single matrix X of dimension $1600 \times (32 \cdot 32)$, where each row is a flattened image. We then apply SVD, which gives us $X = U\Sigma V^\top$, where U and $V^\top$ are orthogonal, and Σ is a rectangular diagonal matrix of singular values. The principal components (basis images) are the columns of V. These basis images describe the variation of X, and can be thought of as "building blocks" for our data.

Since SVD organizes our basis images in such a way that the most important of these come first, we can reform our data using a much smaller number of basis elements. Since V is orthogonal, we can write $XV = U\Sigma = C$, where C is a collection of coefficients c. Each c_i is the coefficient of the projection of row x of X onto $\text{span}\{v_1, \ldots, v_k\}$ for columns v_i of V.

NEURAL NETWORK RESULTS

This project originally utilized 15 principal components to train the neural network, effectively reducing the size of its target output from 984 to 15. The trained network predicts the coefficients c corresponding to the input vector of control parameters, which can then be used to recover the desired YAG image. When comparing the results of the neural network with the images recreated from PCA, the images are nearly identical, as illustrated in Figure 4.

We have seen experimentally that basic neural networks that are made up of dense, fully-connected layers (called multilayer perceptrons) can be accurately trained using only the PCA coefficients from a matrix of flattened images. More specifically, our neural network inputs a vector in $\mathbb{R}^9$, and

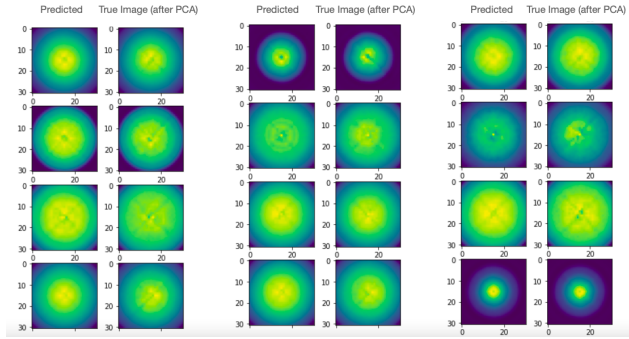

Figure 5: Predicted images compared to the original images recreated by PCA.

outputs a vector in $\mathbb{R}^{15}$, which can be reformed into an image by multiplying it to the first 15 columns of V, and reshaping it into a 32×32 image.

Creating a Neural Network is entirely dependent on the hyperparameters used, and the general architecture of the network itself. The number of layers, number of nodes in each layer, amount of dropout to include between each layer, batch size, learning rate, and loss function all play a huge roll in the outcome of a neural network. A hyperparameter tuning algorithm called Hyperband [4] was recently published to help find the best settings for various hyperparameters within a neural network. Hyperband uses a random search algorithm, but instead of running an entire set of epochs per random set of parameters, it only goes through a few epochs before starting on a new random set of parameters. The motivation behind this idea stems from the fact that you can often tell whether or not a given setup will be useless within the first few iterations. After going through all of the possible combinations of hyperparameters for a few epochs, the algorithm discards the hyperparameter settings it deems poor, and goes through a few more epochs of the remaining networks. Hyperband continues this process until it is down to a few combinations, and returns the best performing network settings. Based on the network architecture returned by Hyperband, a network with a small number of layers and a high number of nodes per layer yielded the best results in terms of loss in the validation data. The results are shown in Figure 5.

NONLINEAR LEAST-SQUARES

To calibrate control parameters, we include an optimization technique to correctly predict the true input parameters when the measured values have been perturbed in some way. We model the scenario as a non-linear least squares minimization problem. In particular, the problem is stated as

$$\begin{cases} \text{minimize} & \|f(x) - y\|_2^2 \\ \text{subject to} & \|x - x_0\|_2 \leq \varepsilon, \end{cases} \tag{1}$$

where f is the surrogate model as a function of the vector of input parameters x, and y represents the measured output. The problem iterates over x, starting with some perturbed vector x_0 and converging to an approximation of the true input parameters $\hat{x}$.

ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-TUPA34

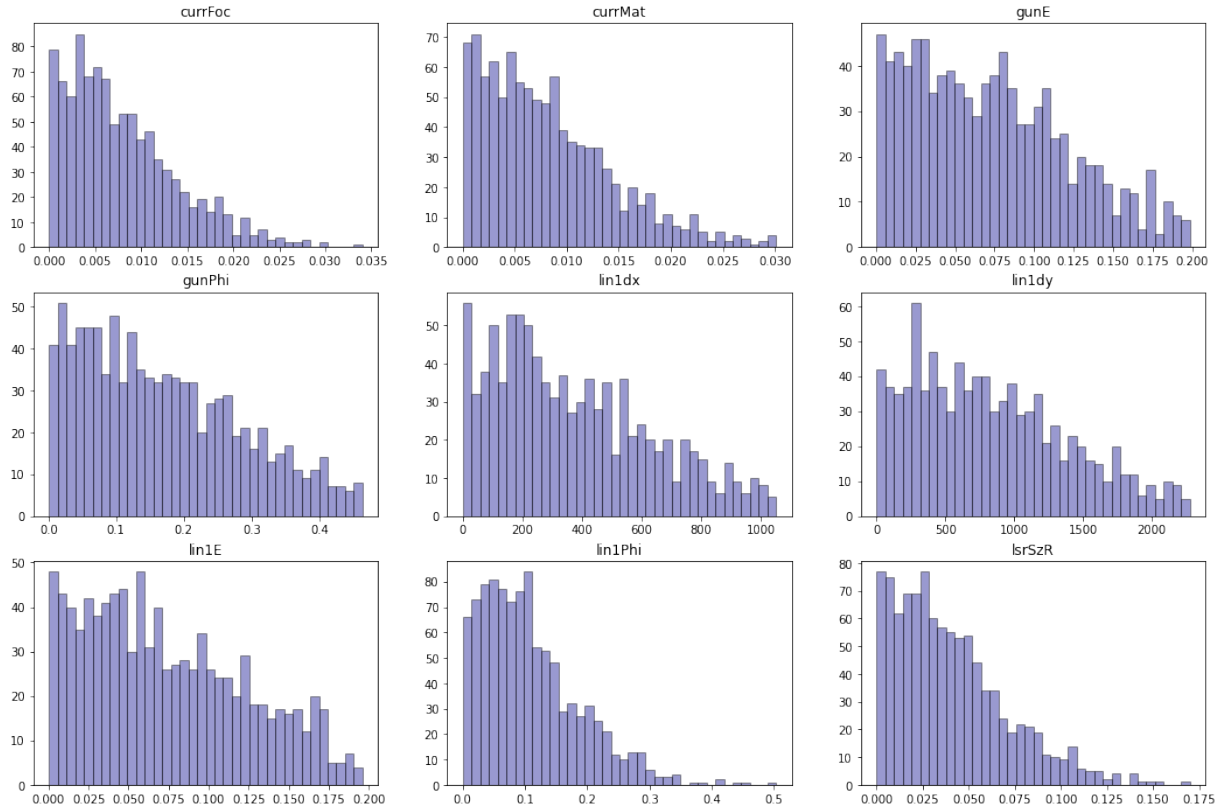

Figure 6: The result of running 1000 least squares problems with randomly perturbed initial parameters using the Levenberg-Marquardt algorithm. The horizontal axis represents the relative error, and the vertical axis represents frequency.

Rather than assume our initial guess of x_0 will be within some region of the true solution, we instead use the Levenberg-Marquardt algorithm [5] to solve the least-squares objective as an unconstrained problem. Figure 6 shows the result of running 1000 least squares problems with randomly perturbed initial parameters using the Levenberg-Marquardt algorithm. There is a histogram for every parameter. The horizontal axis is the relative error $\frac{|x_i - \hat{x}_i|}{|\hat{x}_i|}$, where $\hat{x}_i$ is the true ith parameter and x_i is the ith parameter corresponding to the solution to the least squares problem. Note that "currFoc" and "currMat" have a maximum relative error of just 3%, the parameters "gunE," "lin1E," and "lsrSzR" have a maximum relative error of about 20%, "gunPhi" and "lin1Phi" have maximum relative errors of about 50%, and "lin1dx" and "lin1dy" have a maximum relative error of over 100,000% and 200,000%, respectively.

CONCLUSION

The parameters "lin1dx" and "lin1dy" (L1 misalignment) as seen in Figure 6 are still not being accurately predicted compared to other parameters. This is likely due to a data augmentation technique used increase the size of the training data. Further work is being done to improve the results of this optimization, such as gathering more usable data to retrain the network.

The YAG images from the accelerator itself are much larger and have many other components. These include a ring toward the outer edges of the image, and a large variety in size, shape, and location of the beam. One way to combat these changes is to crop each image uniformly, centered at the center of mass of the beam. This will be implemented in the near future.

REFERENCES

[1] A. Edelen, N. Neveu, M. Frey, Y. Huber, C. Mayes, and A. Adelmann, "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems," *Phys. Rev. Accel. Beams*, vol. 23, p. 044 601, 2020. doi:10.1103/PhysRevAccelBeams.23.044601

[2] A. Adelmann *et al.*, *OPAL a versatile tool for charged particle accelerator simulations*, 2019. doi:10.48550/arXiv.1905.06654

[3] A. Scheinker, F. Cropp, S. Paiagua, and D. Filippetto, *Adaptive deep learning for time-varying systems with hidden parameters: Predicting changing input beam distributions of compact particle accelerators*, 2021. doi:10.48550/arXiv.2102.10510

[4] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, *Hyperband: A novel bandit-based approach to hyperparameter optimization*, 2018. doi:10.48550/arXiv.1603.06560

[5] K. Levenberg, "A method for the solution of certain non–linear problems in least squares," *Quart. Appl. Math.*, vol. 2, pp. 164–168, 1944. doi:10.1090/qam/10666