Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems

Auralee Edelen [1,*] Nicole Neveu,[1] Matthias Frey,[2] Yannick Huber [2]
Christopher Mayes,[1] and Andreas Adelmann[2,†]

[1]*SLAC National Laboratory, Menlo Park, 94025 California, USA*
[2]*Paul Scherrer Institut, 5232 Villigen, Switzerland*

High-fidelity physics simulations are powerful tools in the design and optimization of charged particle accelerators. However, the computational burden of these simulations often limits their use in practice for design optimization and experiment planning. It also precludes their use as on-line models tied directly to accelerator operation. We introduce an approach based on machine learning to create nonlinear, fast-executing surrogate models that are informed by a sparse sampling of the physics simulation. The models are $\mathcal{O}(10^6)$–$\mathcal{O}(10^7)$ times more computationally efficient to execute. We also demonstrate that these models can be reliably used with multiobjective optimization to obtain orders-of-magnitude speedup in initial design studies and experiment planning. For example, we required 132 times fewer simulation evaluations to obtain an equivalent solution for our main test case, and initial studies suggest that between 330–550 times fewer simulation evaluations are needed when using an iterative retraining process. Our approach enables new ways for high-fidelity particle accelerator simulations to be used, at comparatively little computational cost.

I. INTRODUCTION

Physics simulations are essential tools for the initial design of modern particle accelerator systems, as well as for the subsequent optimization of new operating configurations. However, there is generally a trade-off between simulation speed and accuracy in terms of the represented physics effects. Standard codes for simulating accelerator systems can be computationally intensive to run, particularly when complex beam behavior must be taken into account (e.g., instabilities, collective effects, beam self-fields). Exacerbating this computational burden, accelerator systems often consist of many components that can be used to accelerate and manipulate the beam (e.g., accelerating cavities, bending and focusing magnets, collimators). Each of these components has controllable variables that can be independently adjusted to achieve specific beam characteristics. In many cases, the subtle interactions between all variables must be considered, and modeling these systems from "start to end" (i.e., from the beginning of the accelerator to a final point of interest) is critical for

obtaining realistic predictions. To support this, design and optimization studies for particle accelerator systems often require the use of thousands of cores at high performance computing (HPC) facilities. While in principle many large accelerator facilities have access to such resources, in practice this computational burden significantly hampers efforts to conduct comprehensive optimization studies.

Optimization studies are important in the initial design of particle accelerator systems, when many trade-offs between possible setting combinations have to be explored. In practice, multiobjective optimization with genetic algorithms (GAs) [1,2] is frequently used for finding optimal setting combinations (see [3–7] for accelerator-specific examples). One advantage of using multiobjective optimization is that it enables one to examine optimal trade-offs between achievable beam parameters. This is done via examination of the estimated Pareto fronts, which delineate the limit at which one can no longer improve a particular parameter without negatively impacting another parameter. These trade-offs drive the selection of machine working points, which in turn guide the rest of the design process (such as selection of rf equipment with appropriate specifications).

For accelerators that are already in operation, off-line optimization is also used to aid in experiment planning and setup. This is especially the case for facilities that require frequent retuning of settings. For example, at free electron laser (FEL) facilities like the Linac Coherent Light Source

---

*edelen@slac.stanford.edu
†andreas.adelmann@psi.ch

(LCLS) and Swiss Free Electron Laser (SwissFEL), a variety of scientific user requests for specific beam parameters need to be accomodated, and new configurations (e.g., novel FEL schemes) are often developed during limited blocks of time between the scheduled user experiments. Simulations can be used to aid both.

Even though high-fidelity physics simulations are often created as part of the initial design process for a new accelerator, they are often not fully utilized during machine operation (i.e., as "on-line models") for on-the-fly optimization and control. On-line models can be used in model-based control and can provide estimates of normally inaccessible beam parameters. They can also be used to perform rapid analysis in the control room and to plan out new courses of action as goals during an experimental shift may change. In addition, on-line models can be used to help identify when anomalous conditions have arisen (for example, if model predictions suddenly show a sharp increase in error). High-fidelity physics simulations are typically not used on-line due to their computational expense: the execution speed is often too slow to aid operation. Instead, on-line models tend to rely on greatly simplified representations of the machine physics (e.g., see [8–10]), and as a result trade accuracy for speed.

In light of these limitations, improving the execution speed and scalability of particle accelerator simulations is an area that has seen considerable effort in recent years [11,12]. Approaches to do this have focused on parallelization (e.g., see [13,14]) and hardware-based acceleration of existing simulation codes (e.g., using graphical processing units, GPUs) [15,16]. In a few exceptional cases, computationally expensive models have been used to aid live operation when on-site HPC resources are available [17]. Improvements to underlying modeling algorithms, such as using the Lorentz boosted frame [18] and spectral solvers [19,20], have also provided orders of magnitude increases in computation speed. All of these efforts are highly successful. However, it remains the case that the computational expense of these simulations prevents them from being fully utilized.

Here we explore a different, but complementary, approach that immediately enables new capabilities in how these existing high-fidelity physics simulations can be used by the particle accelerator community. We show that one can create machine learning (ML) based surrogate models to obtain accurate, fast-executing representations of the relevant beam dynamics from a sparse sampling of the physics simulation of interest. In contrast to the physics simulation, the ML models can execute in fractions of a second on a laptop with comparable accuracy in predicting the resultant beam parameters. We also show that these models are useful for multiobjective optimization in two important ways: (1) they can accurately reproduce optimization results obtained from the physics simulation, meaning they can be reliably used in experiment planning

and live optimization during accelerator operation, and (2) they can be used to substantially speed up the initial design process by eliminating the need to run an optimization algorithm entirely on the simulation. In addition, although we do not address it in this work, these models can in principle be updated with machine measurements (e.g., see [21,22] for an initial example) to help improve model fidelity with respect to the real machine behavior.

We have used ML models in several previous instances to create fast-executing surrogates for computationally intensive accelerator simulations [21–25]. Here, we build on those works and take a substantial step forward by evaluating such models for use in optimization (and multiobjective optimization in particular). Aiding optimization is one of the main anticipated use cases for ML in particle accelerator applications [26,27], and as such this work represents an important contribution to the particle accelerator community. We also evaluate how many training samples are needed to obtain an accurate model when used for optimization and make a brief comparison between different classes of ML models.

We demonstrate the proposed approach considering two different types of accelerator systems: the injector at the
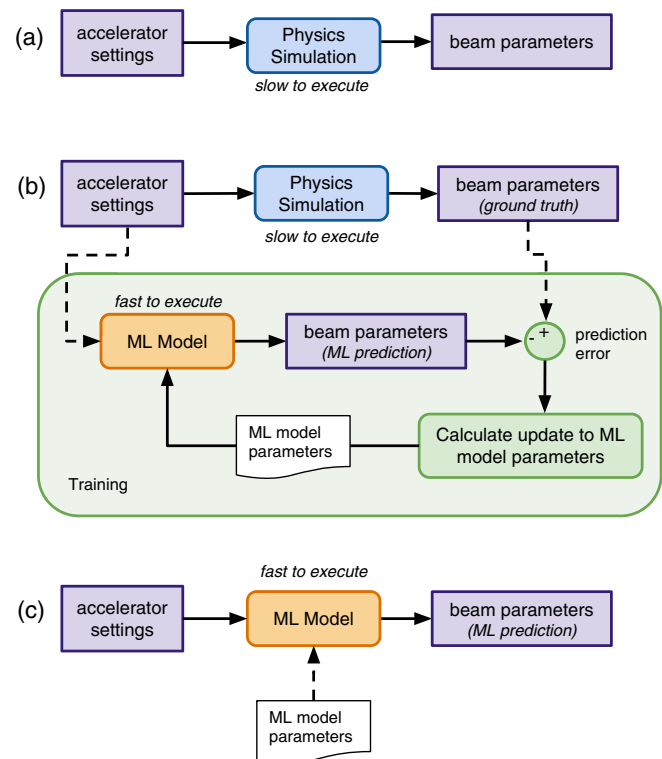

FIG. 1. Initially, we have a computationally expensive physics simulation (a). We then use the physics simulation to generate a sparse set of training data for the ML model that covers a wide range of input settings. The ML model parameters are then optimized until the predictions of the beam parameters match those from the physics simulation (b). The result is a fast-executing representation of the physics simulation that can be used for optimization and on-line modeling (c).

Argonne Wakefield Accelerator (AWA) Facility [28] (a linear accelerator) and a high-intensity cyclotron proposed for the search for sterile neutrinos. The latter system is based on the Isotope Decay At Rest (IsoDAR) design, as detailed in [29]. The AWA injector has a simple layout that is very similar to that used by other accelerator facilities, whereas the cyclotron is a substantially more complex machine (see [30] for more discussion on the dynamics of various types of machine designs). These two cases were chosen specifically to show the generality of the proposed approach. For simulating both accelerators, the OPAL simulation framework is used. OPAL [31] is a parallel, particle-in-cell (PIC) code that handles nonlinear and collective beam effects (e.g., coherent synchrotron radiation, 3D space charge).

It is common practice in the accelerator community to use GAs for multiobjective optimization, although alternatives exist, such as particle swarm optimization [32,33]. Because of its ubiquity, we chose to run the popular NSGA-II [34] algorithm with the ML models as our standard for assessing their performance. For brevity, in the subsequent text we refer to the OPAL simulation of the AWA and

IsoDAR as the "physics simulation," and we refer to NSGA-II as the "GA."

II. RESULTS

A. Description of ML approach and validation procedure

The general procedure for creating the ML surrogate models is shown in Fig. 1, and the procedure for using these models to improve the speed of optimization of particle accelerator systems is shown in Fig. 8. An ML model is trained on a sparse random sample of the accelerator input variables and the resulting beam parameters. The ML model can then be used as a fast-executing representation of the physics simulation. In this work, to assess the performance of the ML model when used with an optimization algorithm (see Fig. 2), we run a GA with the physics simulation to optimize settings (e.g., rf cavity phases, rf cavity gradients, solenoid strengths). We then run a GA with the ML model and compare the resultant estimated Pareto fronts. Good agreement between the estimated Pareto fronts indicates that the ML model can
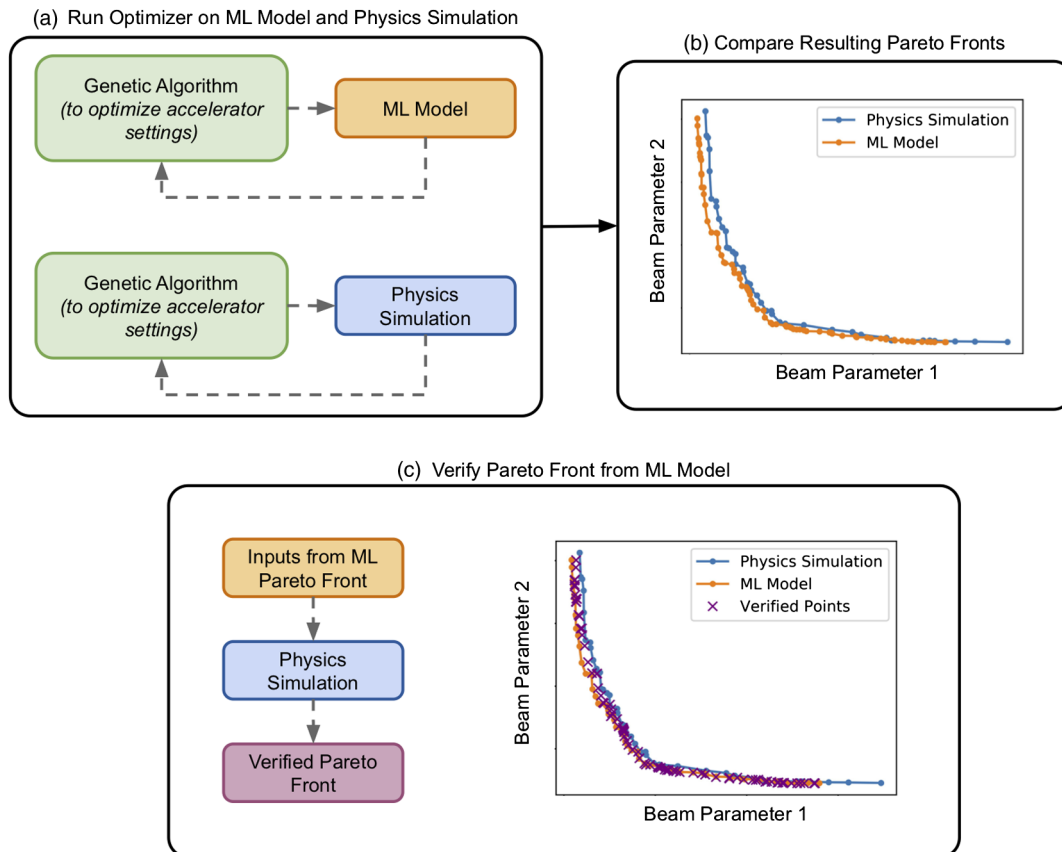

FIG. 2. Approach for assessing the reliability of ML-based model when used for optimization of beam parameters. We run a GA with the physics simulation to find accelerator settings (e.g., rf cavity phases, rf cavity gradients, solenoid strengths) that optimize the beam parameters. The optimization is repeated using the ML model instead of the physics simulation. We then compare the estimated Pareto fronts for key beam parameters. Finally, we run the inputs corresponding to the estimated Pareto front predicted by the ML model through the simulation to verify the accuracy of the prediction.
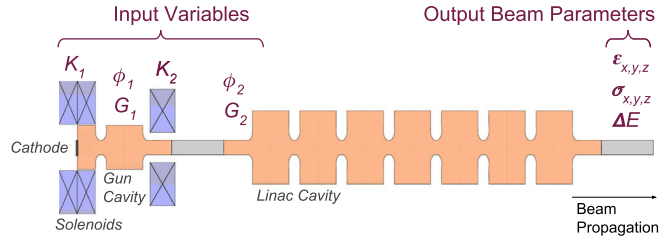
FIG. 3. Schematic of the AWA linac, together with the controllable accelerator settings and predicted beam parameters. The randomly varied inputs include the injector rf phase ϕ_1 and accelerating gradient G_1, the linac cavity rf phase ϕ_2 and accelerating gradient G_2, and two solenoid strengths K_1 and K_2. The output electron beam parameters are the transverse spot sizes σ_x and σ_y, the bunch length σ_z, the transverse projected emittance values ε_x and ε_y, the longitudinal projected emittance ε_z, and the energy spread ΔE. The input variable ranges are determined by typical operating ranges at the AWA and are shown in Table I. We examined this setup for 40 and 1 nC bunch charges.

be used as an accurate replacement for the physics simulation in multiobjective optimization. We also take the input points that correspond to the estimated Pareto front from the ML model and run these through the physics simulation to verify the accuracy of the predictions.

TABLE I. Range of the AWA input variables.

Name	Abbreviation	Minimum value	Maximum value	Unit
Solenoid 1 strength	K_1	400	550	m^{-1}
Solenoid 2 strength	K_2	180	280	m^{-1}
Injector phase	ϕ_1	−10	0	deg
Cavity phase	ϕ_2	−10	0	deg
Injector accelerating gradient	G_1	60	75	MVm^{-1}
Cavity accelerating gradient	G_2	15	25	MVm^{-1}

B. Validation of ML surrogate modeling approach for optimization

We chose to assess this approach first with the AWA linear accelerator. Research at the AWA is focused on advanced accelerator concepts, which generally include efforts to improve control, diagnostic instrumentation, and components (e.g., accelerating structures) for future accelerators. Much effort is also dedicated to developing and testing beam line configurations that could be used for beam shaping [35], or future linear colliders [36]. Often, the accelerator settings (e.g., focusing fields for all magnets, cavity phases, cavity accelerating gradients) are adjusted prior to each experiment to achieve custom beam characteristics (e.g., bunch length and transverse sizes). The accelerator also regularly operates at bunch charges where nonlinear effects are important (e.g., 40 nC), and the cavity fields contain asymmetries. Overall, this results in a challenging optimization problem, and 3D PIC simulations are required to accurately predict the beam behavior. A fast-executing, accurate model of the machine could be useful for supporting the research program of the AWA. Taken together, these factors make the AWA a good test case.

We demonstrate the efficacy of the ML approach by training models on a sparse random sample of six adjustable input variables for the AWA and seven of the resultant beam parameters (see Fig. 3). The inputs were varied uniformly over a relevant operating range of the accelerator (see Table I), and the same range of input variables was allowed for the GA-based optimization. Definitions of the output beam parameters can be found in Table II. While the main focus was on an accelerator configuration with a bunch charge of 40 nC, we also examined a case with 1 nC bunch charge (where nonlinear effects are less important). Details on the datasets, training procedures, implementations of the ML models and the GA, and the details of the physics simulations can be found in the Appendix. We first focus on artificial neural networks (NNs) to demonstrate the technique, and later briefly compare the results with those obtained from polynomial chaos expansion (PCE) [25,37] and support vector regression (SVR) models.

TABLE II. Definitions of beam parameters referred to in the text.

Name	Symbol	Unit	Definition
Position	x, y, z	m	Position of each particle
Momentum	p_x, p_y, p_z	1	$\gamma\beta_x, \gamma\beta_y, \gamma\beta_z$
Normalized emittance	ϵ_x, ϵ_y	m-rad	$\sqrt{\langle x^2 \rangle \cdot \langle p_x^2 \rangle - \langle x \cdot p_x \rangle^2}$, similar for y.
Bunch length	σ_z, σ_s	m	rms bunch size in z or s
Transverse beam size	σ_x, σ_y	m	rms bunch size in x or y
Energy	E	MeV	Mean bunch energy
Energy spread	ΔE	MeV	Energy spread of the bunch
Halo	h_x	1	Particles outside the core, $\frac{\langle x^4 \rangle}{\langle x^2 \rangle^2}$
Particles lost	P_L	%	Percentage of initial particles that travel outside the accelerator aperture

The estimated Pareto fronts obtained using the NN models and the physics simulation are in good agreement (see Fig. 4). Only 500 random sample points were needed to train the NN in the 40 nC case and still generate a set of estimated Pareto fronts that is very close to those obtained with the physics simulation. In contrast, obtaining the same result with the physics simulation required just under 66,000 simulation evaluations. Furthermore, only a small amount of fine-tuning of the NN architecture was done in this case, as the initial topology and hyperparameters were chosen based on previous experience of the authors with similar types of injector modeling problems [21–24]. This highlights the generality of this approach for common kinds of accelerator components and hints at the possibility for doing transfer learning with the produced models (in which a pretrained model can be applied to a new system with relatively little retraining).

It is important to note that verifying the estimated Pareto front that was obtained with the NN model by running these points through the physics simulation does not verify that we have reached the actual Pareto front for the problem. Similarly, although we examined the convergence of the GA that we ran on the physics simulation to ensure it had converged to a stable solution, this is also not necessarily the true Pareto front for the problem. It is simply the one that the GA was able to converge to. Hence, in our assessment we only claim that the *estimated* Pareto front found with the NN model matches the estimated Pareto front obtained by running the GA on the physics simulation (i.e., our ground truth for this comparison).

In order to visualize the extent to which the NN is generalizing to new regions of the parameter space (as opposed to just learning the estimated Pareto front directly from the training data), we compared the training data with the final estimated Pareto fronts obtained with the NN (see Fig. 5). From this we infer that for some beam parameter combinations, the estimated Pareto front is in a region of the parameter space that is not sampled in the training data. This indicates that the NN is able to interpolate in the input parameter space (six input dimensions, seven output dimensions) to find optimal combinations of output beam parameters that are outside of the convex hull of those observed in the training data. In other words, these correspond to more optimal combinations of output beam parameters than were observed during training. Here "more optimal" means that for a given value of an output beam parameter that one would like to minimize, a solution was found where a competing output beam parameter that one also would like to minimize is at a lower value than was obtained with the previous solution.
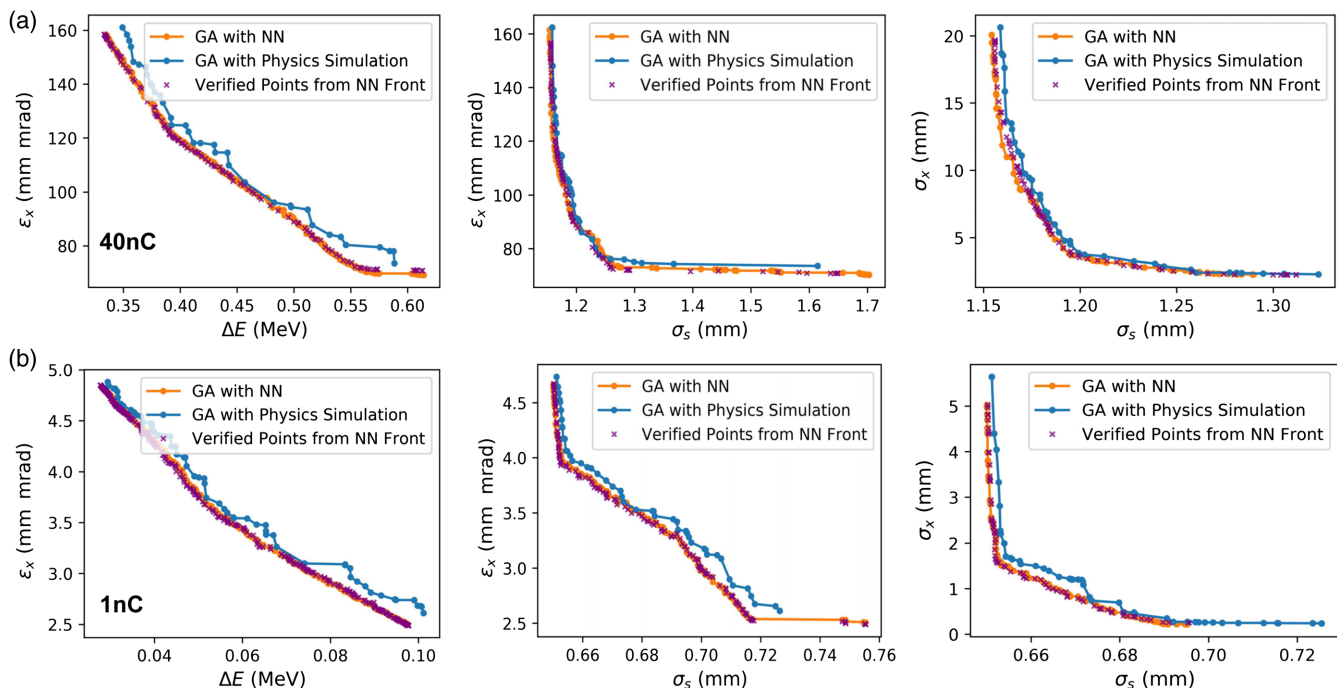

FIG. 4. Comparison between estimated Pareto fronts obtained from the NN and the physics simulation for three sets of beam parameters. We show results at the 40 nC bunch charge (a) and at the 1 nC bunch charge (b) configurations, and we find excellent agreement between the estimated Pareto fronts. In total seven beam parameters have been optimized, and we show the 2D projections from this larger front for the parameters that are most critical for optimization of the AWA. The other projections show similar agreement.
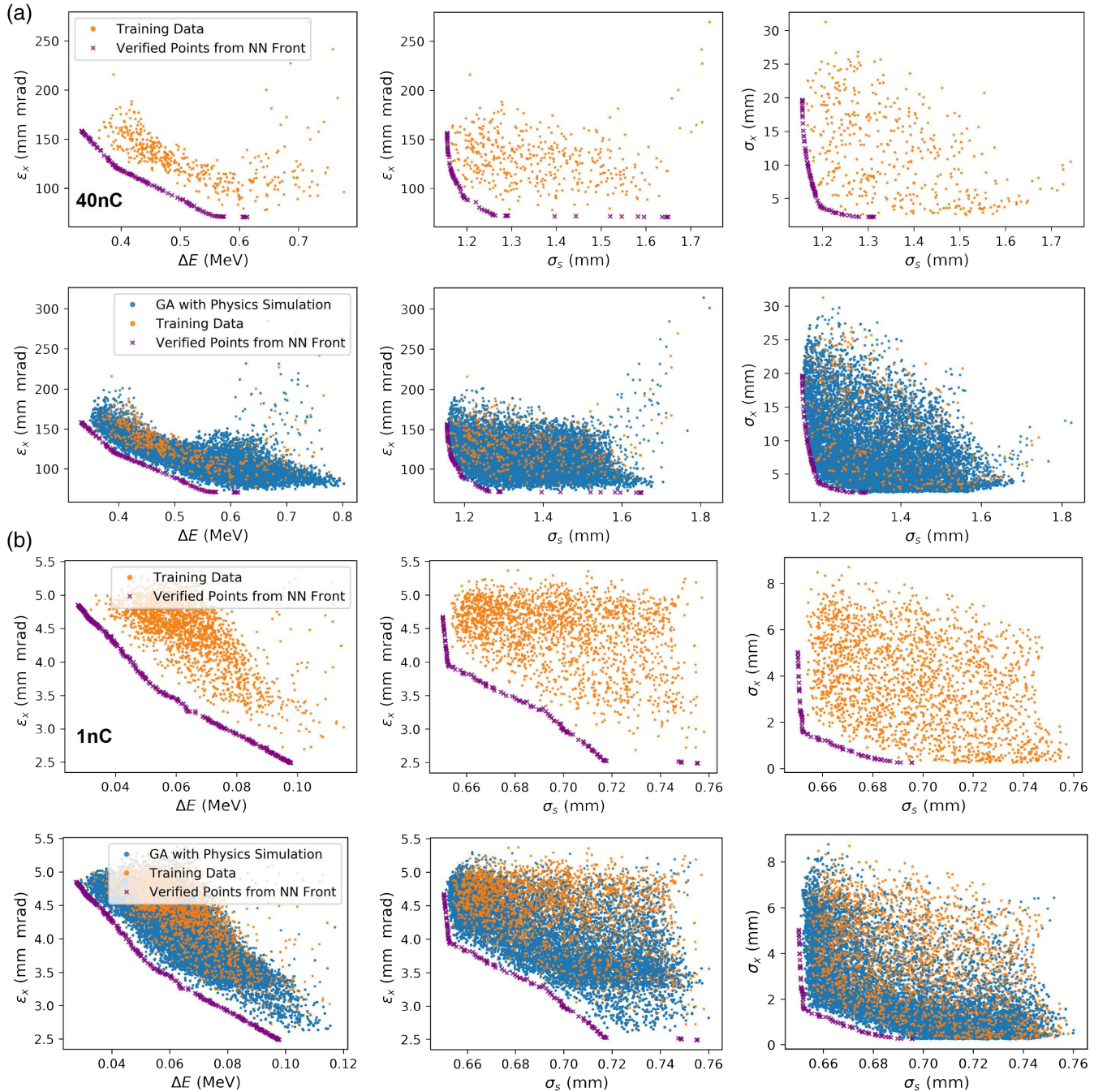
FIG. 5.   Visualization of the solution found by the GA run on the NN model, as compared with both the training set and the points sampled with the GA that was run on the physics simulation. The position of the front indicates that the NN is able to interpolate in the input space to produce better combinations of output parameters than are observed in the training data. We show the sampled points in the training set and the verified points from the estimated Pareto front from the NN for 40 nC (a) and 1 nC (b).

C. Reducing training sample size and iterative retraining

1. Reducing random sample size

Producing training data using the physics simulation is computationally expensive. In light of this, one question which arises is how the accuracy of the ML model will change with the number of samples used in training. This is important for estimating the corresponding trade-off between computation time needed to generate the training set and the resultant model accuracy (the requirements for which may vary depending on the application).

To address this, we trained models using 5000, 500, 200, and 100 randomly sampled points for the 40 nC case and compared the resulting estimated Pareto fronts (see Fig. 6). Note that when using only 500 points, we do not see
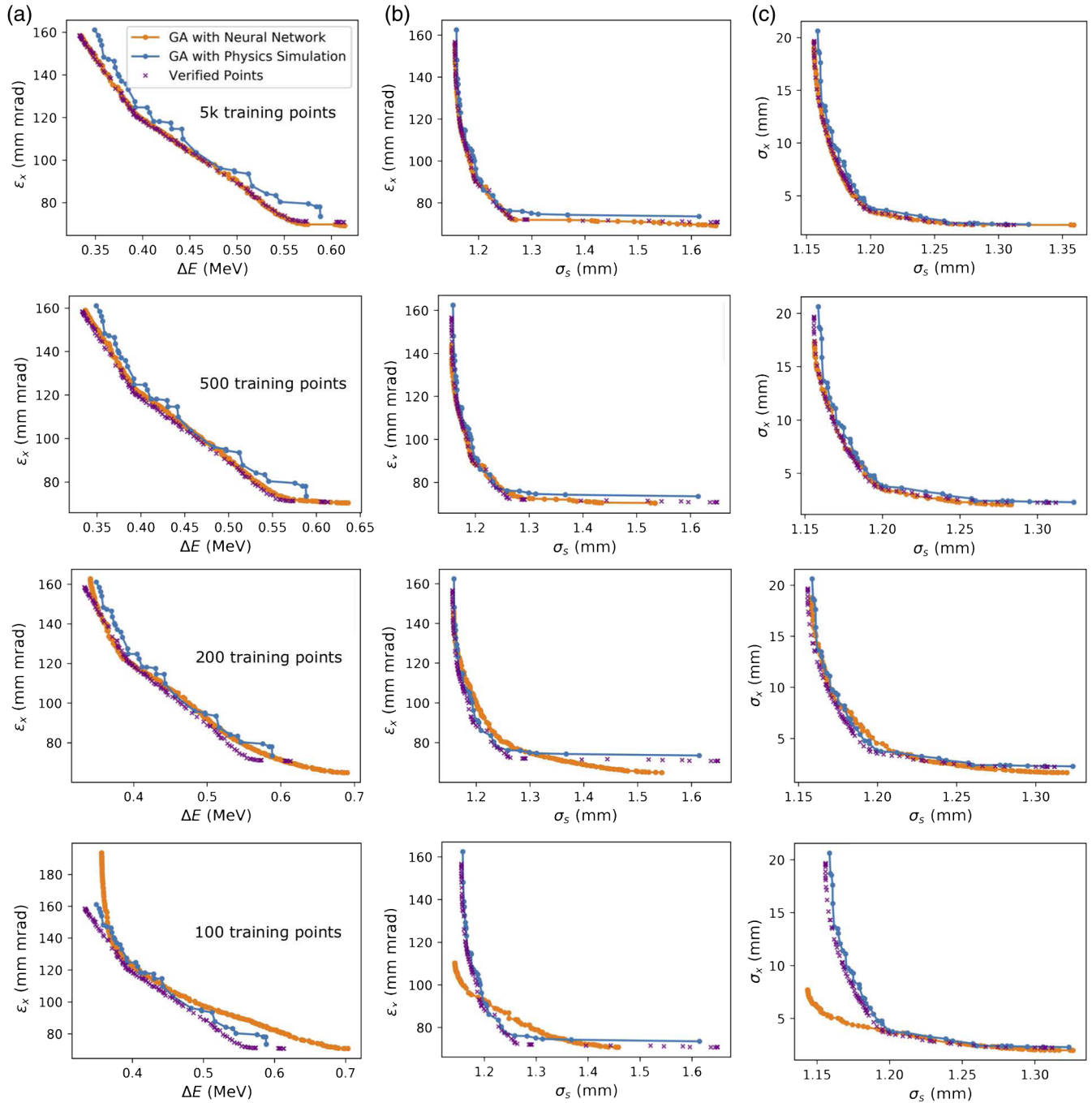
FIG. 6.   Impact of training sample size on the quality of the estimated Pareto front solutions. We show a comparison between estimated Pareto fronts obtained from the NN for three sets parameters in the 40 nC case: ΔE vs ε_x (a), σ_z vs ε_x (b), and σ_z vs σ_x (c). Cases with 5k, 500, 200, and 100 training points are shown from top to bottom. 500 randomly sampled training points are sufficient in this case for obtaining an accurate estimated Pareto front with the NN model. For 200 training points, the estimated Pareto front is quite a bit less accurate but still could provide an initial population for subsequent fine-tuning with a GA run on the physics simulation.

substantial reduction in the estimated Pareto front accuracy, compared with the 5000 point case. For training with 200 points, the prediction starts to deviate substantially from the ground truth, but the solution could still be used as an initial guess (or "warm start") for subsequent fine-tuning with the physics simulation (i.e., by including new points from the estimated front in the training set and iteratively retraining).

We also trained the NN models on a larger range of training sample sizes and evaluated their performance in predicting the points obtained from running the GA with the physics simulation. We quickly see diminishing returns in improvement for the prediction task after the number of samples increases beyond a few thousand. In Fig. 7 we show the impact of changing the training set size on the prediction performance.
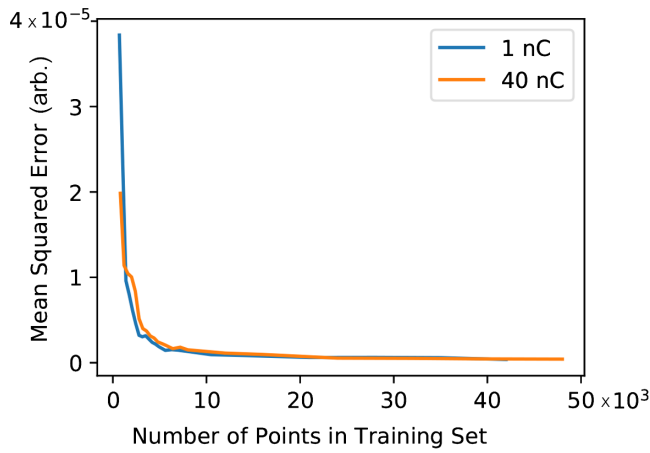
FIG. 7. Prediction error in normalized units for the AWA model trained on various training set sizes. After a few thousand points, we see diminishing returns.

2. Iterative retraining

Note that the uniform random sampling of the input space does not map onto the hypervolume of the output space evenly, and instead we see clusters of points in the output space (for example, see the ΔE vs ε_x plots in Fig. 5). The consequence is that having just a few points for some large parts of the output space results in a model that does not represent these regions as well. It also suggests that the space could be sampled more efficiently (e.g., to avoid oversampling some regions and undersampling others). This points to the potential utility of a more intelligent sampling strategy. To see whether iterative retraining might be a viable approach to reduce the number of required samples, we conducted a preliminary study. For this we take a small initial sample, train the model, run a GA on the NN model, evaluate a small set of the resultant solutions, add these to the training set, and then repeat these steps until convergence. This process is shown in Fig. 8.

To explore the iterative retraining approach, we leveraged the fast execution of the NN model that was trained on all the random sample data (referred to as the "NN surrogate" in the following text) to provide a proxy for the true function. We did this because it would be too computationally intensive for us to explore this approach with the physics simulation directly, particularly when taking into account the number of trials that we anticipated would be needed to refine the method. It also highlights an interesting use of the NN surrogate model: prototyping optimization algorithms. To ensure that it is acceptable to use the NN surrogate as the proxy function in this case, we compared point predictions from it with those from the physics simulation and find that they are in excellent agreement (for example, see Fig. 9). The convergence of the GA run on the NN surrogate also matches the convergence of the GA run on the physics simulation. Beyond this, we also trained a model on actual input and output data from the physics simulation and another model on output data generated by the NN surrogate.
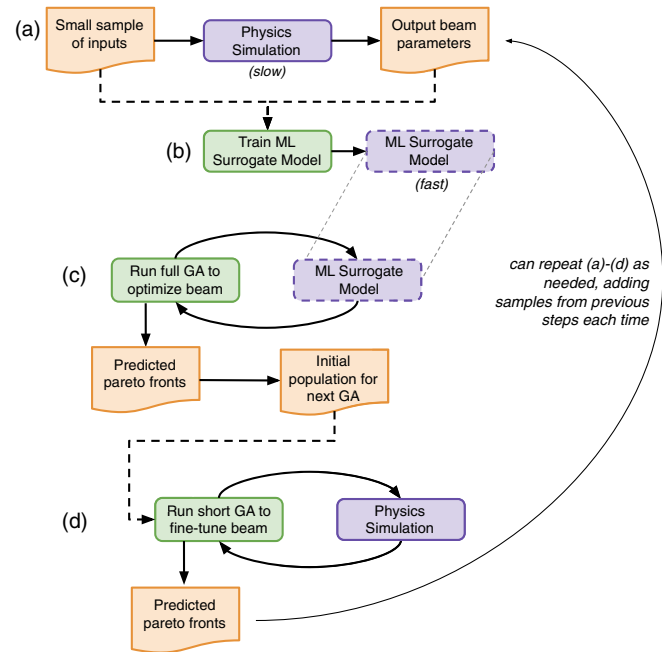

FIG. 8. Workflow for a method to obtain orders of magnitude faster GA optimization of particle accelerators. First, run a small random sample of inputs through the physics simulation (e.g., a few hundred points in our case) (a). Train an ML model on the random sample (b). Run the GA with the ML model to obtain predictions of the estimated Pareto front and corresponding optimal input settings on the machine (c). Use the predicted optimal input settings as the initial population in a second, shorter GA optimization over the physics simulation to fine-tune the result (d). One can also incorporate the new data into a second training of the ML model and repeat these steps as needed until convergence (i.e., iterative retraining and optimization).

In this case we used a 350-point sample so that we were below the threshold that was needed to obtain an accurate front. We then compared the convergence of the GA on each of these models and found that they were the same. Taken together, this provides assurance that we can use the NN surrogate as a proxy for the physics simulation in the iterative retraining study.

For the study, we conducted iterative updates to a second, randomly initialized NN model and observed how many function evaluations were needed to obtain an estimated Pareto front that matches the one obtained from the physics simulation. Note that here we are not directly picking points on the front to add to retraining (which could also be done), but rather, we are selecting a set of additional points using the NSGA-II selection algorithm (i.e., the best-performing solutions with respect to all seven output objectives). We start with a random sample of 20 points, train the NN model, run a GA with 500 generations and 100 individuals in each generation, select 20 points from the final population, run these points through the full NN surrogate (which provides an accurate proxy for the physics simulation in this case), add these new evaluated
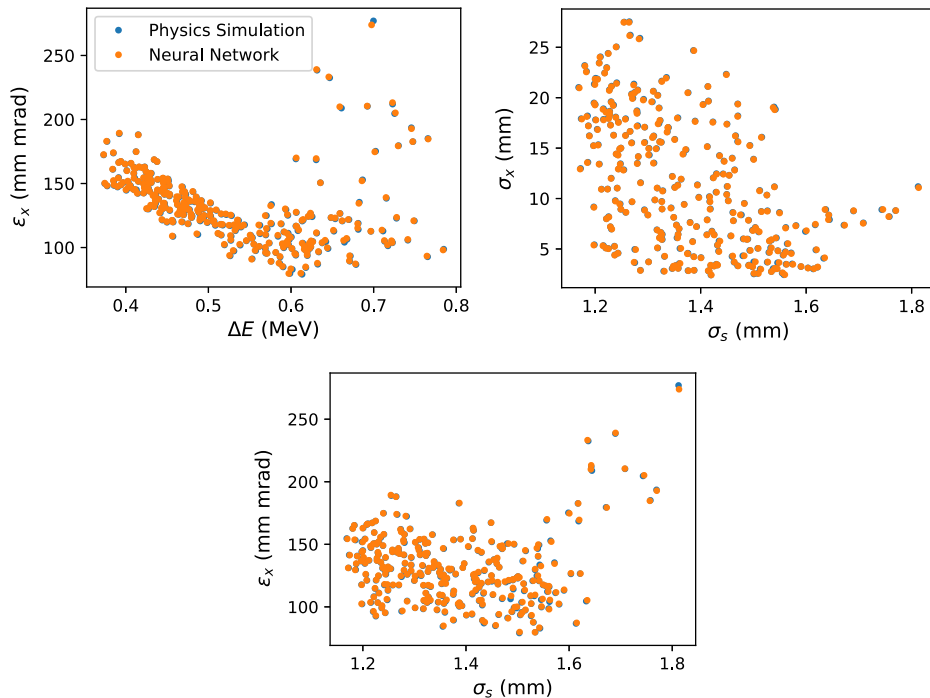
FIG. 9.   Point predictions from the NN surrogate model and the corresponding points from the physics simulation, showing that the two are in excellent agreement. These samples are drawn from the test set.

points the training and validation sets (16 to the former and four to the latter), and then repeat the steps. We repeated this process with ten random sets of initial points and found that for 40 nC we were typically able to reduce the number of required points to obtain an accurate front to between 120 and 200 in total (i.e., 6–10 iterations of the algorithm). An example of one of the slower-to-converge runs is shown in Fig. 10. The variance is due to differences in where the 20 initial randomly sampled points are located (i.e., some of these random initializations will happen to map the output space more evenly than others).

The number of generations and individuals to use for the GA were selected by doing a small search of the GA hyperparameters. Investigating this in more detail (e.g., rigorously assessing the impact of hyperparameters for both the GA optimization and the NN training) is outside the scope of this paper. However, these results affirm the intuition that iterative retraining is a viable option for substantially reducing the number of required training points.

D. Improvement in computational efficiency by using ML model

Once training is completed, one NN model evaluation can be computed in < 1 ms on one core of a *laptop computer*, compared with 590 seconds for one physics simulation on eight cores for the 40 nC simulation. In Table III, the time-to-solution and computing resources

needed for the GA optimization with the physics simulation are compared with those needed for GA optimization with the NN surrogate model. The NN-based GA takes about 2 minutes on a laptop computer, which corresponds to $\mathcal{O}(10^6)$ times fewer computing resources (in terms of core-hours) than the same optimization when conducted with the physics simulation directly.

Importantly, the overall improvement is still substantial when considering the computation time required to generate the training data and to train the NN. This makes the approach a viable way of speeding up initial design optimization. To generate enough training data to produce an accurate estimated Pareto front in the 40 nC case, 132 times fewer simulation evaluations and 144 times fewer total core-hours were required than if one were to use the physics simulation alone. Finally, the NN training itself takes approximately ten minutes on one core of a laptop. Furthermore, for a given problem, this step of generating the data and training the model only needs to be done once, and the NN model can be used for subsequent modeling and optimization tasks.

Note that although we showed results in the previous section that suggest there would be further reduction of the computational resources by conducting iterative retraining, we do not include results for this in Table III because we did not repeat this with the physics simulation itself. On average, the cost of running each iteration of iterative retraining included 33.5 seconds for training and 43.0 seconds for running the GA (with standard deviations
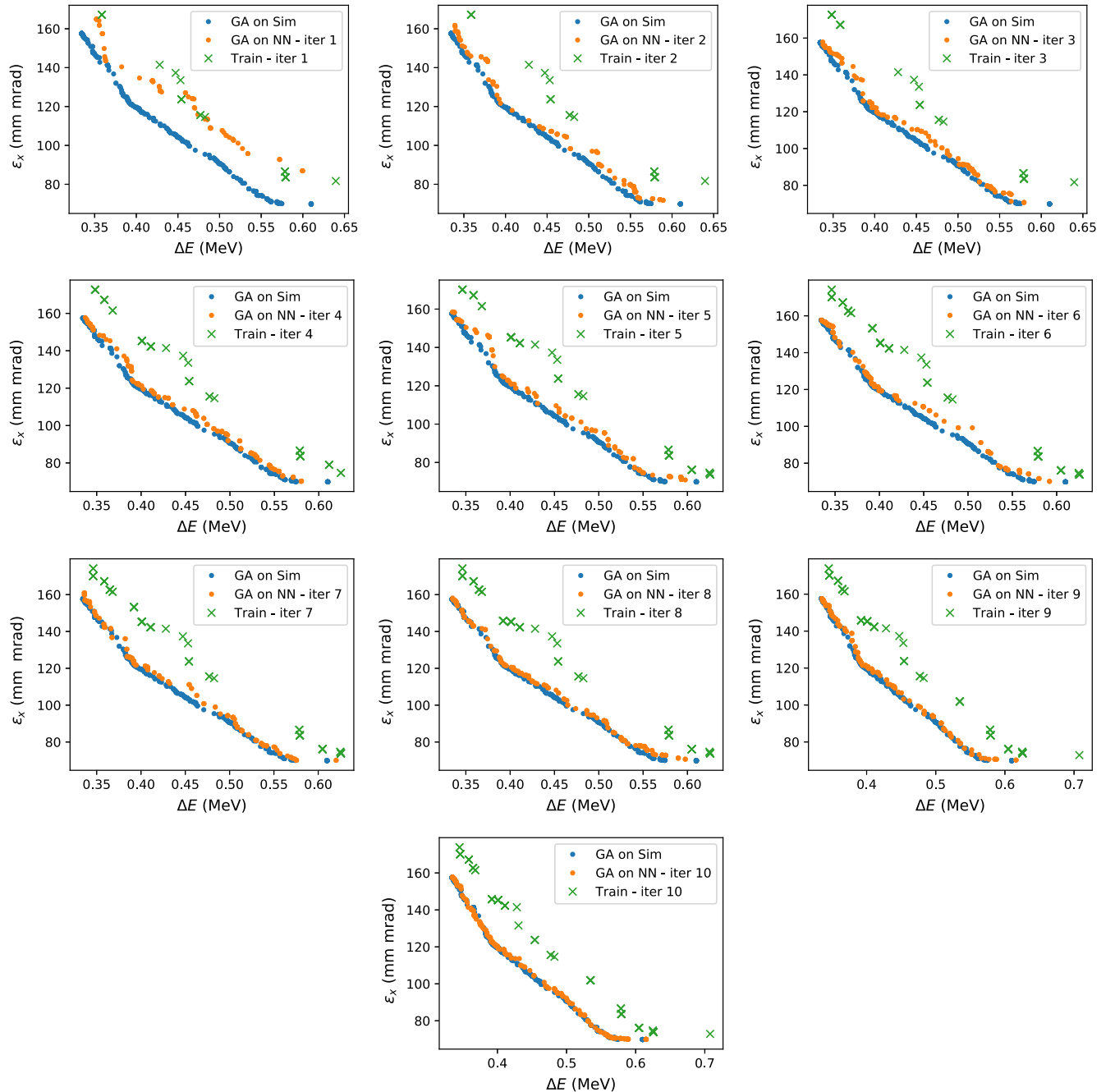
FIG. 10. Iterative retraining: Example of convergence of the estimated front for ΔE vs ε_x during iterative retraining. Here we show ten iterations from one of the trials and display the estimated Pareto front projections from the physics simulation, from the NN model on each iteration of the retraining process, and from the training data used in that iteration. Substantially fewer points (e.g., 200 in this run) are needed to obtain an estimated front that matches the one obtained with the physics simulation, as compared with using a single random sample for training. Note that this example is one of the slower-to-converge runs, which we deliberately chose as a conservative demonstration of the method (i.e., other runs had a fewer number of required points, but that relies on lucky placement of the initial random sample points).

of 34.7 and 24.9 respectively). For a case with ten iterations of convergence, this would add 0.21 core-hours on top of the cost of running the simulation and allow a lower number of simulation samples to be used (likely about 264 core-hours of simulation time instead of 660). This corresponds to a factor of 360 times fewer core-hours than

were needed to obtain the equivalent solution with the physics simulation alone. The NN surrogate is an accurate proxy for the physics simulation (and thus this result is encouraging), but the actual computation time should be verified in future work by running the same iterative retraining procedure with the physics simulation directly.

TABLE III.   Comparison of computing resources: core-hours, wall time, and number of simulation evaluations required for running the GA with the physics simulation and the GA with the NN. Here we show only results for the 40 nC bunch charge, but details on 1 nC bunch charge can be found in the Appendix. When running the same GA with the NN, 3×10^6 times fewer computing resources are required. When including the resources needed to generate the training data (as might be done for initial design optimization), we still have a factor improvement of 144 in terms of core-hours required. Note that although we showed results in the previous section that suggest there would be further reduction of the computational resources by conducting iterative retraining (e.g., 360 times fewer core-hours for ten iterations), we do not include results for this in the table because we did not repeat this with the physics simulation itself.

Method	Calculation	Core-hours	Wall time (hours)	Simulation Evaluations
Physics simulation	GA on OPAL	95 000	36	65 929
ML-based	Generate training data	660	0.33	500
	Train NN	0.17	0.17	not applicable
	GA on NN	0.03	0.03	65 600
	Speedup—training included	144×	109×	132×
	Speedup—training excluded	$3 \times 10^6 \times$	1200×	not applicable

When considering the reduction in computational resources used, it is important to note that the physics simulation had already been tuned for a high level of computational efficiency and we ran the simulations in parallel. As such, the results represent improvement over the state-of-the-art. In cases where simulation codes are less efficient or higher complexity (e.g., plasma simulations, FEL simulations), the ML-based approach may actually enable larger optimization studies to be conducted than would have been feasible using the physics simulation alone.

E. Comparison with different ML models

As a check to see whether a linear model would be sufficient for this problem, we trained a support vector regression (SVR) model [38] with a linear kernel. For the error metric, we used the mean squared error (MSE), defined as $\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$, where N is the number of samples, y_i is the true output for sample i, and $\hat{y}_i$ is the predicted output for sample i. The MSE over all predicted beam parameters using the SVR model was 5.5×10^{-6}, in comparison to 3.5×10^{-10} for the NN
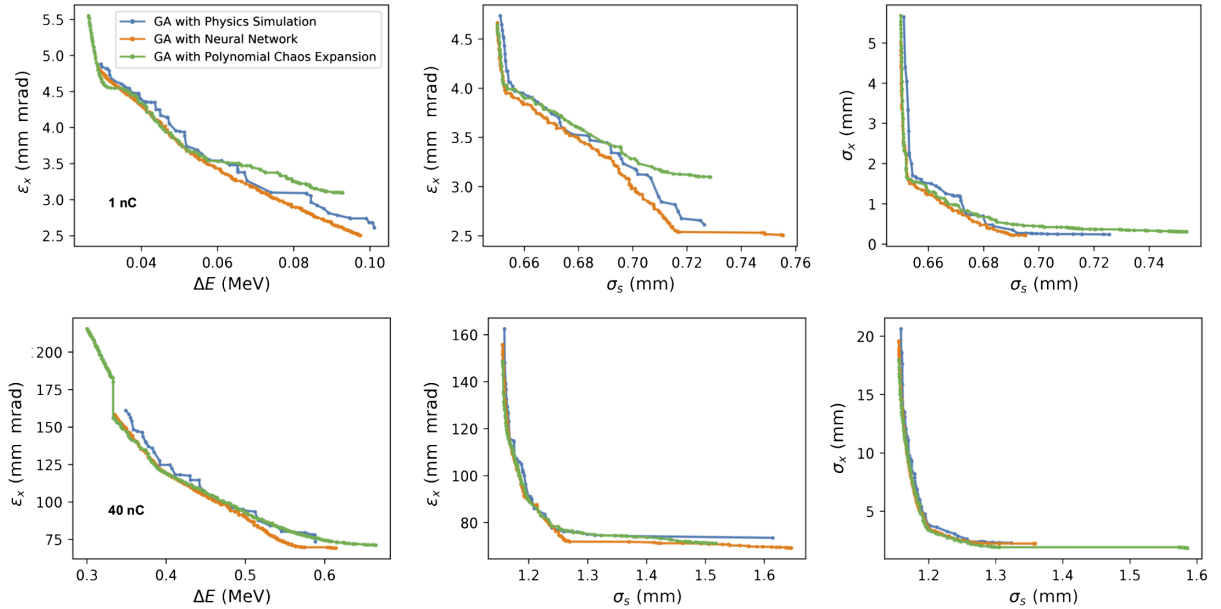

FIG. 11.   Comparison of the estimated Pareto front for a PCE model and a NN model. Note that in this case, we use a random sample of 42,000 points (this was from the large initial random sample, which we ran before attempting to scale down to as few sampled points as possible during training). To make a fair comparison in this case, we use the same number of sample points for the NN as we use for the PCE. Note that although the PCE model does not perform as well, it is more straightforward to train, has fewer hyperparameters to tune, and can be used to provide an uncertainty estimate. Although the estimated fronts are less accurate, the points could be used as an initial starting point for a subsequent GA run on the physics simulation.

model, indicating that we do gain accuracy by using a nonlinear model. Note that although these differences in error may at first strike readers as small, we are dealing with quantities like emittances that have raw values on the order of 10^{-6} m-rad and rms beam sizes that have raw values on the order of 10^{-3} m. For example, for an error in emittance of 0.1×10^{-6} m-rad we would expect a squared error on the order of 10^{-14}. Considering this, the difference between the MSE of the SVR and NN model is substantial.

We also compared performance of NN and PCE surrogate models, both in parameter prediction and in the optimization task. In Fig. 11, we show a comparison between the estimated Pareto fronts obtained with the PCE and the NN models. The MSE of the PCE model for 40 nC was 1.6×10^{-7} and for 1 nC was 4.5×10^{-6}. In contrast, the NN models were more accurate with a MSE of 3.5×10^{-10} and 2.9×10^{-8} on the 40 and 1 nC cases, respectively. However, one downside of using a simple NN model is that it does not inherently give an estimate of prediction uncertainty and model sensitivity without additional analysis. In contrast, the PCE model has the benefit of providing straightforward estimates of prediction uncertainty and sensitivity via the Sobol' indices [25,39]. The PCE model also has fewer hyperparameters to tune (i.e., polynomial order, type of polynomial used). Although the estimated fronts are less accurate, the points from the PCE could be used as an initial starting point for a subsequent GA run on the physics simulation.

III. EXTENSION TO A CHALLENGING CYCLOTRON EXAMPLE

To assess this approach further, we applied it to a higher-dimensional and generally more complex accelerator problem: the IsoDAR cyclotron, shown in Fig. 12. The IsoDAR cyclotron will provide 60 MeV protons for sterile neutrino search [29]. It has also been proposed to use this machine to produce medical isotopes with high efficiency [40]. In high-power proton machines, a major challenge is to minimize the number of uncontrolled, lost particles (thus maximizing the number that are sent to the relevant experiment and also minimizing damage to the machine). An indication for losses are large halos around the core of the particle beam. The halo can be quantified by a set of halo parameters and is proportional to the kurtosis, which is a measure of the "tailedness" of the probability distribution. A detailed description of this problem can be found in [25].

The IsoDAR cyclotron simulation is implemented in OPAL and includes nonlinear beam collective effects such as the beam self-fields. This simulation is an order of magnitude more expensive to evaluate than the AWA due to the long path the particles travel and correspondingly long integration time. We also attempt to predict a more difficult set of beam parameters than we did for the AWA. The most difficult parameter to predict is the beam halo, which is notoriously challenging to simulate accurately.
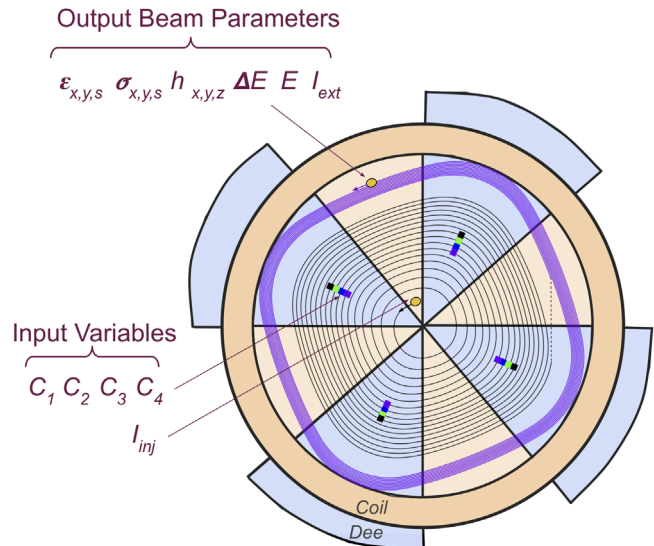

FIG. 12. The IsoDAR schematic, showing an overlay of the magnetic field configuration (coil and dee) and central particle trajectories from the simulation. The rf system used for acceleration is not shown. In black, the centroid trajectories of 18 turns are shown. A particle bunch of intensity I_{inj} is injected at the center and passes through four families of collimators (indicated with the colored rectangles starting at turn 6). Each collimator family has one parameter $C_1 \dots C_4$ defining the collimator aperture. The collimator aperture and the initial intensity are input variables used by the optimizer. In this schematic, the extraction system is not shown, nor are all 100 turns of the beam. The last ten turns before extraction are indicated in purple, together with the output beam parameters: the halo parameters $h_{x,y,z}$, the beam size $\sigma_{x,y,s}$, the projected emittance $\epsilon_{x,y,s}$, beam energy E, and energy spread δ E, and the extracted beam current I_{ext}. Particle losses $(I_{\text{inj}} - I_{\text{ext}})$ in the collimators have to be minimized, and a small h is desired. This is one of the most important considerations in the design phase.

Because the simulation is so computationally intensive, we use only a small number N of particles (i.e., $N = 1.33 \times 10^5$). As a result, we expect that the actual prediction of the halo is not fully converged (which should show up as noisy estimates of the halo).

Using a 2500-point random sample, we trained a NN to predict 12 beam parameter outputs: the total number of lost particles (P_L), the beam energy (E), energy spread (ΔE), transverse and longitudinal beam sizes $(\sigma_x, \sigma_y, \sigma_s)$, transverse and longitudinal emittances $(\epsilon_x, \epsilon_y, \epsilon_s)$, and the beam halo (h_x, h_y, h_z). Table IV shows the ranges of the input variables used in the random sample. Here we use the definition of the halo parameter found in [41], formula 12. Again, definitions of the beam parameters can be found in Table II. The 5 input parameters are the initial proton beam current and the positions of four families of collimators (corresponding to 16 collimators in total). For the GA run on the NN, we use 1000 generations with 300 individuals. Other than this, the implementation is the same as that used for the AWA.

TABLE IV.   Range of the IsoDAR input variables.

Name	Abbreviation	Minimum value	Maximum value	Unit
Initial beam current	I_{inj}	5.5	7.5	mA
Collimator 1	C_1	2.37	2.63	Unitless
Collimator 2	C_2	2.37	2.63	Unitless
Collimator 3	C_3	7.60	8.40	Unitless
Collimator 4	C_4	7.60	8.40	Unitless

As expected, upon verification by running the predicted points through the simulation we find that the estimates (see Fig. 13) are not as accurate as those obtained for the AWA. However, the points still roughly map out the region of parameter space near the boundary sampled by the physics simulation. It is unclear how much of this is a contribution from the simulation itself (e.g. due to numerical noise), since it has not been tuned to ensure that the mesh size and number of particles is sufficient to produce a reliable result. Fine-tuning of the simulation was not conducted because the associated computational intensity of conducting this process for the broad range of
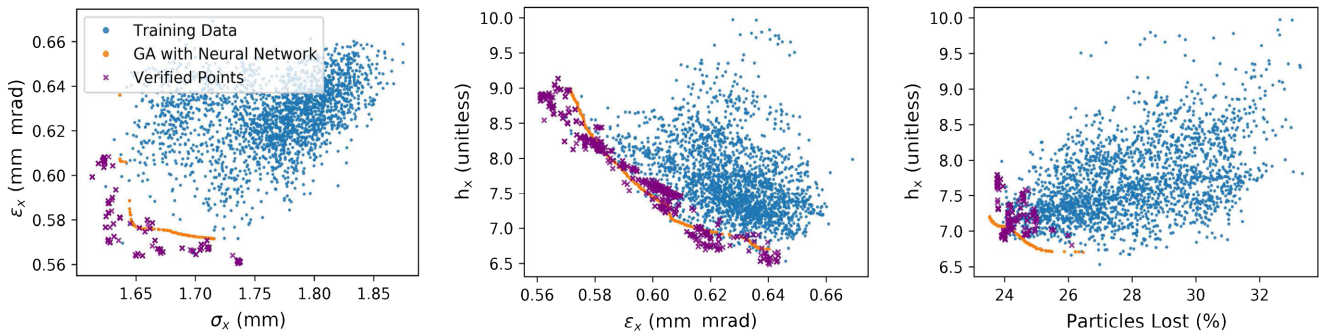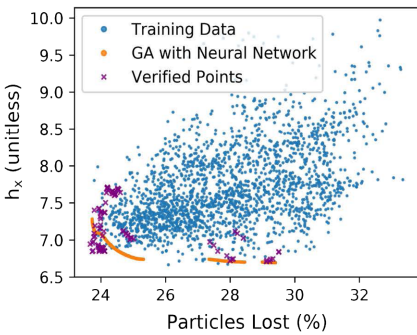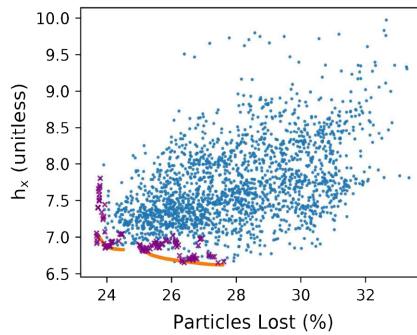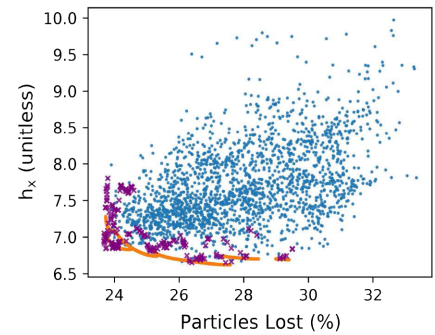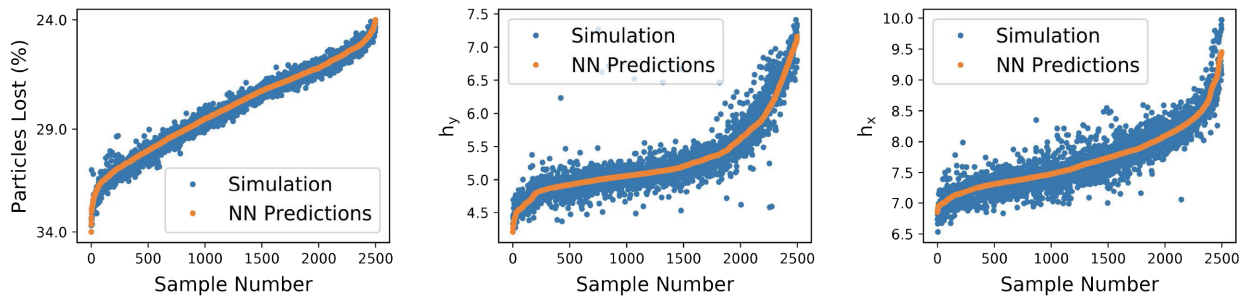

FIG. 13.   Example of estimated Pareto fronts from IsoDAR NN model and corresponding verified points from the physics simulation. We first show results from training a model to predict and optimize 12 beam parameter outputs (a). We then show corresponding results from training only on the two most challenging beam parameter outputs, h_x and P_L (b). We then include the verified front from the previous model in the training set, retrain the model, repeat the optimization, and check the new Pareto points (iterative retraining) (c). Although the fronts do not match as well as they do for the AWA, for this challenging problem it is encouraging that we can predict roughly accurate fronts. Finally, we show the prediction on sorted values of the random sample set.
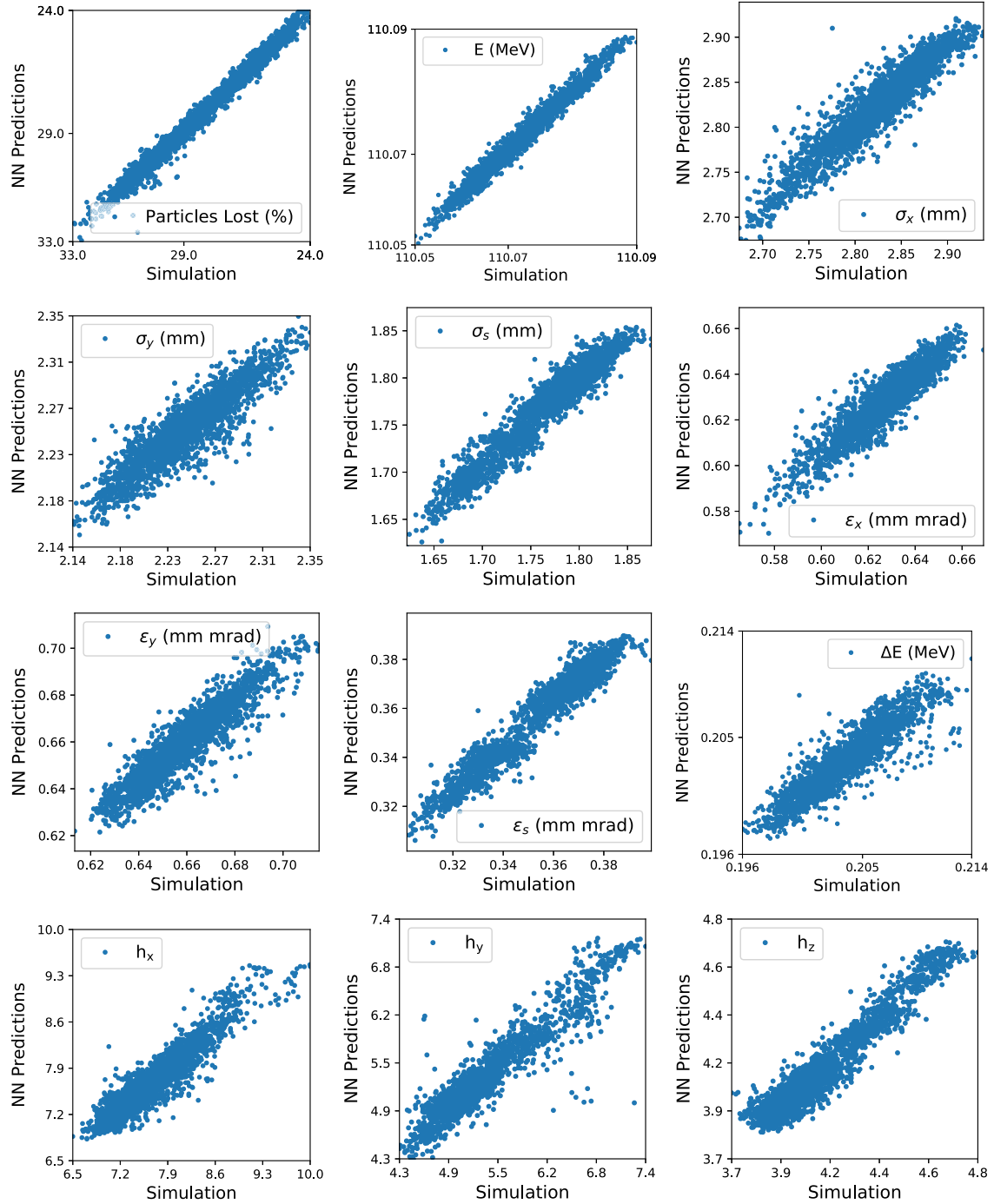
FIG. 14.   Example predictions from IsoDAR model trained on 12 outputs vs true values. Perfect prediction would correspond to a straight diagonal line.

parameters we are varying was too high to be feasible given the compute time available to us. Note also that in this case, we do not have an estimated Pareto front from running the GA on the physics simulation. This is because the IsoDAR simulation is so computationally expensive that we were unable to run a multiobjective optimization on all parameters to convergence. All this taken into account, we consider finding verified points that are in close proximity

to the Pareto front estimated by the NN to be a successful use of the approach, particularly with regard to creating a fast-executing representation of the simulation.

For the trade-off between halo parameters and losses, we obtain results that more closely outline the extent of the observed hypervolume projection when we produce a model that only predicts those direct outputs (as opposed to predicting and then optimizing all 12 objectives).
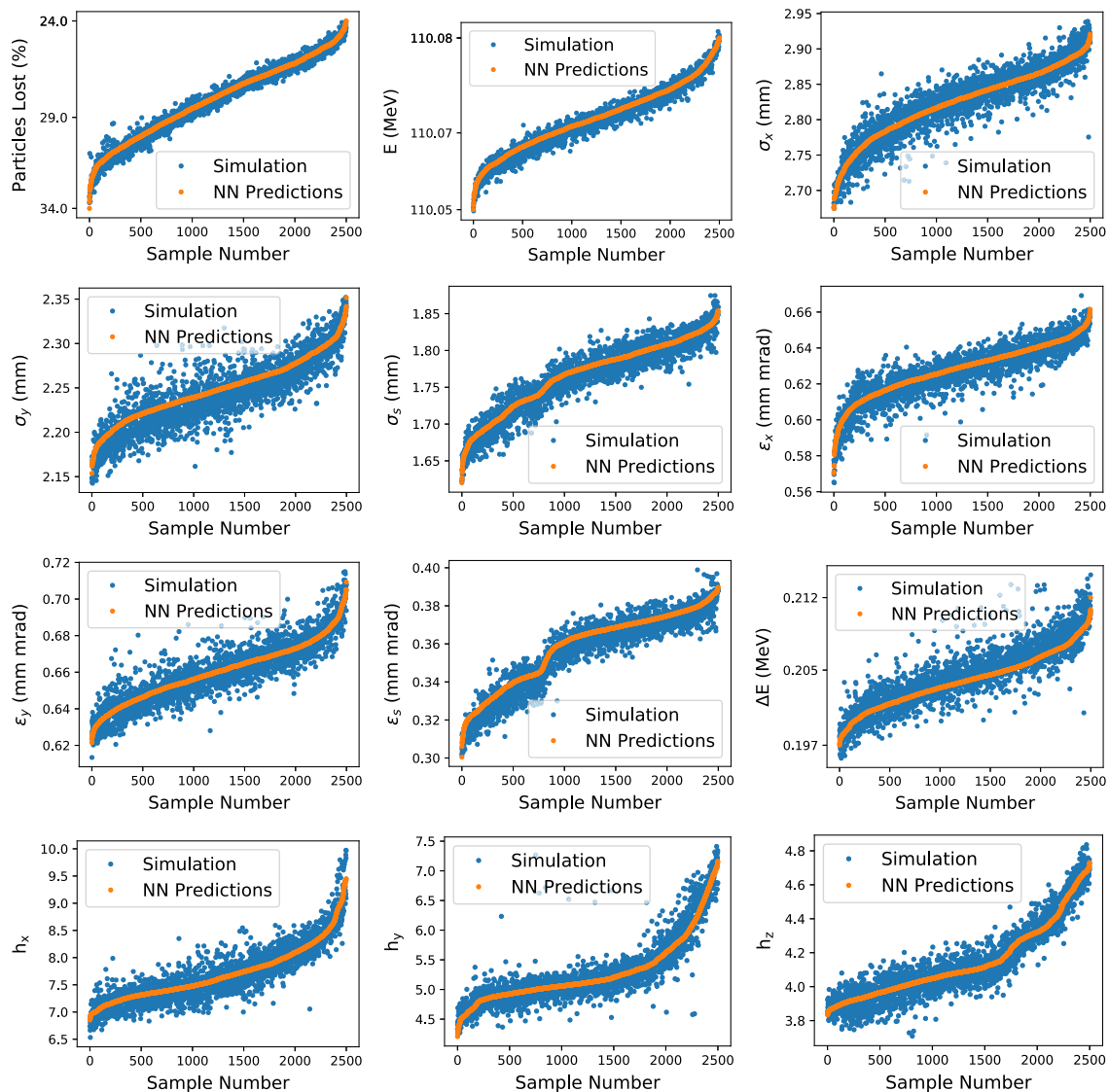
FIG. 15.   Example predictions from IsoDAR model trained on 12 outputs and corresponding values from the simulation.

In Fig. 13, we show this for h_x and P_L. This is also more in line with how GAs are typically used by accelerator physicists at present: often, only two important parameters will be optimized as competing objectives. We also find that by iteratively retraining the model with the verified points and repeating the process with the new model, we obtain a model that fills out more of the front (also shown in Fig. 13).

As a result, we obtain a surrogate model that is $\mathcal{O}(10^7)$ times faster to execute than the original physics simulation and can predict the 12 beam parameter outputs with reasonable accuracy. To illustrate this, in Figs. 14 and 15, we show some examples of the prediction performance of the IsoDAR model on the random sample dataset. We show the prediction on the raw sorted values and plot the expected vs predicted values against one another. As discussed, the results are likely also impacted by numerical

noise, and the outliers in the dataset also illustrate the need for uncertainty estimates along with the prediction.

IV. CONCLUSIONS AND DISCUSSION

In practice, physics simulations of particle accelerators are often too computationally expensive for full exploration of the parameter space during optimization. The computational expense also limits (and in many cases prohibits) their use during machine operation to aid in prediction and control. In this work, we have shown that machine learning (and NNs in particular) can be used to obtain a fast-executing representation of computationally intensive accelerator physics simulations, and these models can be reliably used in multiobjective optimization. We have also shown that in some cases relatively little data is needed to achieve a high degree of fidelity relative to the original physics simulation.

The NN surrogate models are more accurate than the simplified physics models that are presently used when fast execution is needed (e.g., fast envelope codes, simple tracking codes used with a small number of particles, etc.). This approach thus provides one avenue toward creating fast-executing representations of high-fidelity physics simulations for use in machine operation, which would not otherwise be possible. It also enables faster start-to-end optimization, which could in turn help facilitate more extensive off-line design optimization and experiment planning, as well as aid the prototyping of novel accelerator operating modes. Finally, as we demonstrate, these models can be used to quickly prototype new optimization algorithms.

Our results also suggest a new procedure for doing GA optimization of particle accelerators. Instead of running a GA with a physics simulation, one can run a small random sample spanning the parameter space, train a surrogate model on this sample, and run a full GA using the surrogate model very quickly. The estimated Pareto points obtained from the surrogate model could then be used as an initial population in a subsequent short GA with the physics simulation to verify and fine-tune the result (see Fig. 8). The process can also be repeated until convergence to a good solution is reached (i.e., using iterative retraining and reoptimizing).

We found that optimization with the NN model requires substantially fewer simulation evaluations than a purely simulation-based optimization (e.g., 132 times fewer simulations with one random sample in the case of the 40 nC setup for the AWA, with indications that this could be further reduced to 330–550 times fewer simulations with iterative retraining).

The computational cost of the parameter space exploration with the GA before it gets close to convergence is high. Using the NN model enables one to skip these early stages of convergence and form a model that can be used to interpolate across the parameter space. Assuming the physical function to be modeled is smooth and that the NN model is learning a good representation of the underlying physical system, running the GA with the NN model can thus provide an estimate of the Pareto front in a fraction of the time needed to run the GA on the physics simulation. This opens up the possibility to do more extensive optimization than might otherwise have been feasible using computationally intensive physics simulations alone.

Using NN models to aid the optimization process could also be useful in cases where one cannot in practice run a GA for a sufficient number of generations or with a large enough population to converge. This can be the case when HPC resources are limited and/or the optimization problem is too high dimensional or computationally intensive. Related to this, a larger generation and population size can be used with the NN than would normally be feasible with the physics simulation, thus potentially enabling better

solutions to be found during the optimization process (e.g., in contrast to doing only a short GA over a limited range with the physics simulation).

Overall, while training a NN as an intermediate step in the optimization process may seem cumbersome, GAs do have their own hyperparameters that need to be tuned (e.g., population size, number of generations, crossover and mutation probabilities). The risk of wasting computational resources while tuning these parameters is high, and in practice accelerator physicists often just pick a number of generations to run based on experience.

During preparation of this manuscript, a similar approach that employs iterative retraining but conducts the optimization of the model and selection of new points slightly differently was demonstrated on the problem of nonlinear dynamics optimization of the SPEAR3 storage ring [42], showing similar promise. This is an encouraging indicator that NN-based iterative retraining can be applied to different kinds of machines to speed up design optimization.

We anticipate that the approach presented in this work will be useful for a variety of applications, including design optimization, prototyping of optimization routines, off-line experiment planning, and on-line optimization of machine settings. Overall, the approach is quite general, and as many beam dynamics problems in accelerators are similar, it is also reasonable to expect that these results could provide good starting points for applying this approach to other kinds of accelerator systems. For example, injector systems that are similar in scale and complexity to the AWA are extremely common, and the results we show in this work should provide good guidance for those wishing to use this method on similar components. The fact that we used a similar NN architecture to other injector modeling problems (e.g., [21–24,43,44]) also hints at the possibility of doing transfer learning between models (e.g., training on one injector system and then reusing the model with small updates for a similar injector system). While the IsoDAR cyclotron is a more unique design, the performance of the ML approach on that case shows it can also be used in cases with much more complicated beam dynamics.

To move from proof-of-concept demonstrations to regular deployment in accelerator applications, several areas of future work are apparent. These include addressing how best to scale these methods up to larger or more computationally intensive systems, how to obtain accurate estimates of prediction uncertainty, how best to combine simulation and measured data (i.e., to make efficient use of both, and to improve the model accuracy with respect to the real machine), how best to incorporate prior physics knowledge into these models, and how best to account for machine drift and keep the model updated over time. We discuss these directions of future work in more detail below.

A. Incorporation into on-line modeling and model-based control

First, one can begin to incorporate these surrogate models directly in machine operation. In some cases, high-fidelity simulations have been made to match the machine very closely and can be used to provide suggested machine settings (for a comprehensive example, see [45]). If the simulation matches the machine closely enough, the ML model trained on the simulations can immediately be used to aid operation.

Machine operators could use these models to check the potential impact of setting changes before trying them out on the actual machine, or to assess a new course of action as goals change during an operating shift. These models could also be used as a diagnostic tool to provide predictions about unmeasured beam parameters (i.e., as a virtual diagnostic [21,26,46]) or to flag when the system has changed substantially (i.e., model-based anomaly detection). They could also be exploited in model-based control and model-guided optimization routines (i.e., using the model to help guide the search for optimal settings, as is done in model predictive control and Bayesian optimization [47]). Bayesian optimization with Gaussian process models has been successfully used in on-line optimization of operational accelerators [48–52], and a pretrained NN model with prediction uncertainties could in principle be used in a similar fashion.

The NN surrogate models can also be used to provide a warm start to a local, feedback-based optimization algorithm (i.e., by reading unvaried inputs from the machine to get an estimate of the present system state, running an optimization algorithm on the model for the variable settings to get an initial solution, and then refining this with on-line feedback with the machine directly). This is similar to the approaches described and tested in [22,23,26,43,53,54], but it uses optimization around the forward model to obtain the initial suggested settings rather than using an inverse model (i.e. directly mapping desired beam outputs to suggested settings). The advantage of the combined approach (model-based methods combined with model-free local feedback) is that it can help compensate for inevitable discrepancies between the model and the machine (e.g., due to drift, hidden variables, etc.) without necessitating retraining. As was demonstrated experimentally using an inverse model in [53], the suggested settings from the model only need to be close enough to the basin of a good minimum to allow a local optimizer to converge. Forward models can also be used to help train inverse models (e.g. for accelerator-specific applications see [23,44]).

In contrast to relying on local optimization methods or hand tuning only, these methods could be exploited to reduce the time spent switching between custom beam requests. However, all of these possible applications need to be explored in practice and rigorously assessed before they will be ready for dedicated on-line deployment.

B. Updating models with measured data

Second, getting the physics simulation to match the measured machine behavior can be very difficult and usually requires substantial effort. Even simulations with the main expected physics effects included often deviate substantially from the observed machine behavior. As a result, many accelerator facilities do not prioritize creating accurate physics simulations (particularly since high-fidelity physics simulations could typically not be used directly in operation anyway, at least not prior to the introduction of the approach discussed in this paper).

With a surrogate model, one can instead update the model learned in simulation with measurements from the machine to account for deviations between the simulation and the real machine behavior. This was shown to be viable in [21,22], but more rigorous study is needed to address how best to preserve information gained from the simulation while also updating the model with respect to measured data.

By creating learned models that are trained at least in part on measured data, subtle statistical correlations across the machine that may otherwise go unnoticed and unutilized by human operators can be exploited. Training on simulations prior to this reduces the need to rely only on machine time and data available in the archive, thus potentially enabling regions of parameter space that otherwise would go unseen to be included. For some use cases, such as identifying the source of discrepancies between the machine and the simulation, it may also be more useful to learn to predict the error between the measured data and the model that was trained in simulation rather than updating the model directly. Finding ways to effectively combine measured and simulated data is an important direction of future work.

C. Accounting for drift and unseen operating conditions

Third, another challenge concerns how best to update the models with measured data in a reliable fashion so that one may maintain prediction accuracy despite drift in the machine response and exposure to new regions of parameter space. First, one needs to be able to reliably identify when the model is no longer accurate for the present operating condition. Raw prediction errors for available signals or uncertainty predictions can be used to help decide when the model needs to be retrained and how much to trust a given prediction (e.g., to help decide whether to use that prediction in control or analysis, or rely on alternative methods instead).

Then the model must be updated in a reliable fashion without manual intervention. Online retraining has been demonstrated for a feed-forward correction scheme to improve source size stability at the ALS [55]. However, for accelerators that frequently switch operating conditions and have very large operating ranges automatic retraining when a high prediction error is observed could result in a loss of valuable information about previously visited machine states. This is due to the well-known problem of "catastrophic forgetting" in NNs [56–58], and it was also

observed in practice in early tests at the FAST injector [59] when attempting to do on-line retraining of the NN model described in [27,60] for different rf power levels.

All in all, finding good strategies for updating the model or otherwise accounting for drift while preserving previously learned behavior from different operating configurations will be critical. Depending on the particular details of the accelerator (e.g., how much drift there is from day to day, how flexible the operating conditions are, how much noise there is, to what extent the diagnostic information captures the variables that affect the machine behavior), the strategies to do this most effectively will likely vary.

D. Inclusion of model uncertainty

Fourth, another challenge concerns how to obtain an accurate measure of the prediction uncertainty, particularly for NN-based models. Uncertainty predictions can be used to decide when the model needs to be retrained, or when the model should not be relied upon. Although this is relevant to both on-line and off-line applications (and for simulations that exhibit noise), it is particularly important for on-line deployment or cases where measured data is used in training. This is because the input-output relationships of operational accelerators are subject to a variety of sources of uncertainty, including noise, intermittent anomalous conditions (e.g., due to equipment failures), drift over time, and the influence of unobserved variables.

In the context of optimization, uncertainty predictions can also be used to help choose subsequent points to evaluate. For example, one may want to assess the expected benefit of running a computationally expensive simulation at a given operating point to fill out the parameter space of a model, or one may want to weigh the risk of moving to a new operating point on the live machine against the likely performance improvement from that new operating point. While this approach is well established for Bayesian optimization techniques (especially using Gaussian process models), obtaining reliable uncertainty estimates from NNs is an open area of research [61–63]. Developing expressive models that also include uncertainty estimates (e.g., model ensembles, Bayesian NNs, Gaussian process models with NN-based kernels [64], deep GPs [65]) is a reasonable next step.

E. Efficient sampling strategies

Sampling of simulations to produce an initial model can itself be time consuming in cases where the cost of obtaining a sample is extremely high or when many variables must be included. In some cases, the random sampling and iterative retraining strategies we used in this work may not be sufficient, and more intelligent sampling strategies will likely be needed in order to map the parameter space fully while minimizing the number of simulation evaluations. Even in the AWA case, we observed that for the random sample, some areas of the output space were overrepresented and others were underrepresented.

F. Scaling to higher dimension and complexity

While NNs in principle can be used to model high-dimensional, complex systems, determining how best to scale this method to accelerator systems with a much greater number of input/output variables, wider ranges of variables, or more complex beam dynamics is an important question that will also need to be examined in future work.

G. Including prior physics information

Present approaches discard our rich knowledge of accelerator physics. Methods should be developed which incorporate this physics knowledge into these models so that fewer training examples are required and improved performance in unseen regions of parameter space can be obtained. There are elegant examples of this already for GP models. For example, in [51] and [52], it was demonstrated that using physics models to directly inform the GP kernel hyperparameters resulted in faster convergence in Bayesian optimization of accelerator settings.

ACKNOWLEDGMENTS

APPENDIX

1. Datasets for the surrogate models

We generated uniformly distributed random samples from the physics simulation. For this we used the OPAL-based interface for creating such datasets, which was developed in part to support this effort. This feature allows the submission of massively parallel jobs using an OPAL input file.

For the AWA, the randomly varied inputs include the injector phase ϕ_1 and gradient G_1, the linac cavity phase ϕ_2 and gradient G_2, and two solenoid strengths K_1 and K_2. The output parameters are the transverse spot sizes σ_x and

σ_y, the longitudinal beam size σ_s, the transverse projected emittances values ε_x and ε_y, the longitudinal projected emittance ε_s, and the energy spread ΔE. The input variable ranges are informed by the operating ranges at the AWA (see Table I). Random samples for two bunch charges were generated (1 and 40 nC, with the corresponding laser radius being 2 and 9 mm). In the 1 nC case, we generated 70k samples, and in the 40 nC case, we generated 80k samples. However, we only use a small subset of these during training of the models.

For the IsoDAR cyclotron, the input parameters varied are the initial beam current I_{inj} and four collimator settings C_1–C_4. The ranges of the collimators and the beam current are determined by technical design considerations. A random sample of 2500 points was generated for the initial training dataset. For retraining, 100 points from the previous estimated Pareto front were selected randomly to add to the training set. Table IV shows the ranges of the input variables used in the random sample.

For running the GA on the AWA simulation, we choose an initial population of 656 individuals and subsequently evolve the population over 200 generations, while retaining the same number of individuals in each generation. The following hyperparameters were used: gene mutation probability $P_g = 0.8$, mutation probability $P_m = 0.8$, and recombination probability $P_r = 0.2$. Specific descriptions of the hyperparameters can be found in Sec. 1.4.2 of the OPAL manual [31]. The specific values used in this case were chosen based on previous optimization work for the AWA that involved a hyperparameter scan [5]. The constraints of the problem were set such that the variables stayed within the operating ranges of the AWA (see Table I). For each generation, the input and output parameters from the simulation are saved. In the 1 and 40 nC cases respectively, 59,285 and 65,929 final samples were obtained. See Table V for an overview of the computational resources required to make the physics simulation datasets.

Note that for the IsoDAR cyclotron, we do not run a GA on the physics simulation. This is because the IsoDAR simulation is so computationally expensive that we were unable to successfully run a multiobjective optimization on all parameters to convergence.

The implementation of NSGA-II [34] that is provided within OPAL was used for the GA with the physics simulation of the AWA. Details related to the algorithm can be found in [66]. The implementation of the GA used with OPAL is slightly different than the implementation in DEAP (used with the ML models) because some of the adjustable hyperparameters are defined differently. Originally, we had intended to use DEAP for both the physics simulation and the ML model. However, we ran into practical limitations in being able to run the OPAL simulation in parallel using DEAP. OPAL had ostensibly the same algorithm programmed in a way that allowed efficient parallelism (making it feasible to run the GA on the physics simulation in a reasonable amount of time). We did not at the time have a way to run DEAP in parallel on the HPC systems that we had access to. Thus, we used the same nominal algorithm in each case (NSGA-II), but the implementation is slightly different in DEAP and OPAL.

After setting up LIBENSEMBLE [67] to do the parallel simulations in conjunction with DEAP, we were later able to do a direct comparison between DEAP's version of NSGA-II and OPAL's version by running both of these with the physics simulation. When comparing the two algorithms on OPAL simulations of the AWA, the fronts produced by each method are in reasonable agreement, as shown in Fig. 16. The main source of the discrepancy between the OPAL-GA solution and the DEAP solution is the slight difference between the algorithm implementation. Note that this does not alter the conclusions of the paper. Specifically, two points need to be considered: (1) the solution from the NN model is in agreement with both the OPAL and DEAP solutions (which themselves have deviations between one another but are in general agreement) after running 200 generations with 656 individuals in each generation, and (2) the beam parameters found with the NN model are more optimal than are seen in the training set (which is a separate consideration from the DEAP or OPAL GA solutions).

2. OPAL simulation

OPAL is an open-source, parallel library for electrostatic PIC simulations of charged particle accelerators. More details can be found in the OPAL manual (see [31]). For

TABLE V. Overview of the datasets from the physics simulations and the computational resources used for generating them. Note that we found that to make accurate surrogate models we needed many fewer randomly sampled points than were initially generated. For the GAs, 2624 cores were used. For the 1 nC random sample, 16k cores were used, and for the 40 nC random sample, 15k cores were used.

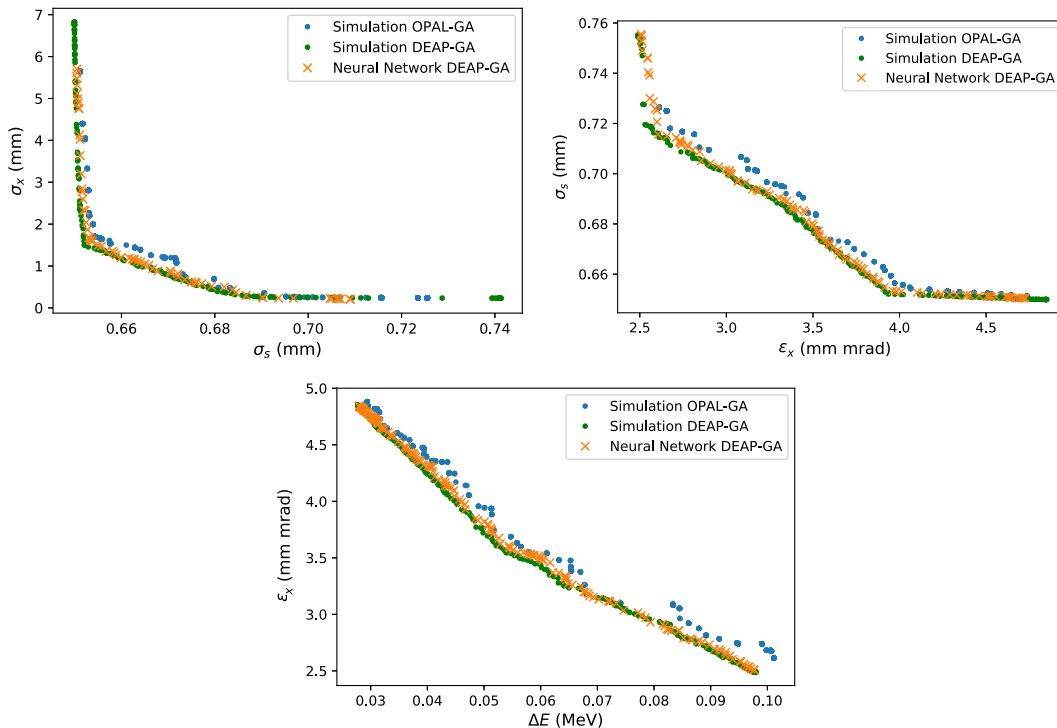| Case	Calculation	Core-hours	Wall time (hours)	Simulation evaluations
AWA 1 nC	Genetic algorithm (200 gen)	43 500	16.56	65 929
	Random sample	60 600	4.12	70 000
	500 points	283	0.14	500
AWA 40 nC	Genetic algorithm (200 gen)	95 000	36	65 928
	Random sample	115 700	7.23	80 000
	500 points	660	0.33	500
IsoDAR	Random sample	13 335	1.7	2 500

FIG. 16.   Comparison between estimated Pareto fronts obtained with the OPAL implementation of NSGA-II and the DEAP implementation of NSGA-II after 200 generations, with 656 individuals in each generation. We also show the result from running DEAP on the neural network surrogate model. We consider these to be in sufficiently close agreement; please refer to the text for more discussion.

simulations of the AWA, 3D space charge forces are calculated throughout the time evolution of the beam, which is important for realistically capturing the nonlinear impact of the beam self-fields. The particle generation at the CsTe photocathode is modeled using a uniform emission model, assuming a planar ideal surface. The laser profile used for emission is uniform transversely and a flattop longitudinally, with Gaussian tails. Convergence studies were previous done to determine an appropriate time step, mesh size, and number of particles to use (here, 1×10^{-11} seconds for the time step, $16 \times 16 \times 32$ grid cells for the space charge mesh, and 10k macroparticles). The full-width-half-maximum of the laser in the longitudinal direction was 6 ps for both cases. The laser radius was set to 2 mm for the 1 nC simulations. Due to large nonlinear space charge forces at 40 nC, the laser radius was increased to 9 mm. These are typical operating conditions at the AWA.

Field maps generated in POISSON [68] were used to model the solenoid magnets. Two types of rf field maps were used to model the gun and accelerating cavities. 2D maps were generated in SUPERFISH [69] and used in the 1 nC case. 3D maps were generated in ACE3P [70] and used in the 40 nC simulations. While 3D field maps are computationally more expensive to evaluate, they are more accurate and capture asymmetries that are present in the AWA rf cavities.

The IsoDAR simulation is described in Sec. II of [71]. In addition, in this work we added four collimators to clean up

the beam (i.e., reduce halo). The collimators are placed in the central region of the cyclotron where the energy is low and the activation is negligible.

3. Implementation of machine learning based surrogate models

The NNs were implemented in KERAS [72], with TensorFlow [73] as the backend. For general demonstration of the technique, we used a topology and set of hyperparameters that roughly correspond to those the authors had previously found to work well for similar problems in accelerators [21–24,43]. This consisted of a fully connected, feed-forward NN with four hidden layers, each with 20 nodes and hyperbolic tangent activation functions. No regularization penalties (e.g., L_1 or L_2 norm) were used on the weights. The NNs were trained for 10k epochs with a batch size of 500 points. The Adam optimization algorithm [74] was used for training, with an initial learning rate of 0.001 and hyperparameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$. For training, the random sample data was randomly split into training (60%), validation (20%), and testing (20%) sets. All datasets were scaled to fit within an appropriate range. For example, in our case the data was scaled to be within the range of $[-1, 1]$. For the IsoDAR problem, the setup is the same, except we use a neural network with a slightly different number of nodes in each hidden layer: $10 - 20 - 20 - 15$ nodes in each layer respectively.

The surrogate model based on polynomial chaos expansion (PCE) is constructed using the Uncertainty Quantification Toolkit (UQTk) [75,76]. This library provides functionalities to perform an intrusive as well as a non-intrusive UQ in C++ and PYTHON. In contrast to the projection method of [25], we used the regression method [76,77] with Legendre polynomials, and we associate a uniform distribution to all input variables. In this work, we closely follow [25] in regard to the PCE surrogate model. Furthermore, choosing a polynomial order of $p = 4$ and 60% of the random sample for training matches the performance of the NN model most closely.

The GA optimization with the surrogate models is done using the PYTHON package DEAP [78] and its standard implementation of NSGA-II. We picked hyperparameters that were as close as possible to those used with the OPAL GA.

4. Code availability

For this research only open source software is used. This includes the accelerator simulation framework OPAL [31] and PYTHON-based software tools: DEAP [78], KERAS [72], TensorFlow [73], UQTk [75], and SCI-KIT LEARN [79]. For access to additional materials contact authors Auralee Edelen and Andreas Adelmann.

————————

[1] M. Mitchell, *An Introduction to Genetic Algorithms* (MIT Press, Cambridge, MA, 1996).

[2] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford University Press, Inc., New York, 1996).

[3] I. V. Bazarov and C. K. Sinclair, Multivariate optimization of a high brightness dc gun photoinjector, Phys. Rev. ST Accel. Beams **8**, 034202 (2005).

[4] A. Hofler, B. Terzić, M. Kramer, A. Zvezdin, V. Morozov, Y. Roblin, F. Lin, and C. Jarvis, Innovative applications of genetic algorithms to problems in accelerator physics. Phys. Rev. ST Accel. Beams **16**, 010101 (2013).

[5] N. Neveu, L. Spentzouris, A. Adelmann, Y. Ineichen, A. Kolano, C. Metzger-Kraus, C. Bekas, A. Curioni, and P. Arbenz, Parallel general purpose multiobjective optimization framework with application to electron beam dynamics, Phys. Rev. Accel. Beams **22**, 054602 (2019).

[6] Y. Sun, Multiobjective online optimization of beam lifetime at APS, in *Proceedings of NAPAC 2016, Chicago, IL, USA* (JACoW, 2016), WEPOB12, http://accelconf.web.cern.ch/AccelConf/napac2016/papers/wepob12.pdf.

[7] W. F. Bergan *et al.*, Online storage ring optimization using dimension-reduction and genetic algorithms, https://arxiv.org/abs/1807.10720.

[8] M. Woodley, R. Iverson, P. Krejcik, G. White, and J. Wu, XAL-based applications and online model for LCLS, in *Proceedings of PAC, Vancouver, CA* (2010), FR5REP022.

[9] T. Pelaia, Open XAL Status Report 2013, in *Proceedings of the 4th International Particle Accelerator Conference,* *IPAC-2013, Shanghai, China, 2013* (JACoW, Shanghai, China, 2013), MOPWO086.

[10] A. Scheinker and S. Gessner, Adaptive method for electron bunch profile prediction, Phys. Rev. ST Accel. Beams **18**, 102801 (2015).

[11] R. Ryne *et al.*, Scidac advances and applications in computational beam dynamics, J. Phys. Conf. Ser. **16**, 210 (2005).

[12] SciDAC: Scientific discovery through advanced computing, https://scidac.gov/.

[13] A. Adelmann, U. Locans, and A. Suter, The dynamic kernel scheduler–Part 1, Comput. Phys. Commun. **207**, 83 (2016).

[14] M. Borland, ELEGANT: A flexible sdds-compliant code for accelerator simulation, Technical Report No. LS-287, Argonne National Laboratory, 2000–2018.

[15] H. Takeda and J. E. Stovall, Modified PARMILA code for new accelerating structures, in *Proceedings of the Particle Accelerator Conference, Dallas, TX, 1995* (IEEE, New York, 1995), Vol. 4, pp. 2364–2366.

[16] X. Pang, *Advances in Proton Linac Online Modeling, 6th International Particle Accelerator Conference, Richmond, VA, USA* (JACoW, Geneva, Switzerland, 2015), paper WEXC2, https://doi.org/10.18429/JACoW-IPAC2015-WEXC2.

[17] X. Pang, L. J. Rybarcyk, S. A. Baily, High-performance beam simulator for the LANSCE linac, in *Proceedings of the 3rd International Particle Accelerator Conference, New Orleans, LA, 2012* (IEEE, Piscataway, NJ, 2012), C1205201, https://www.osti.gov/servlets/purl/1040799/.

[18] J.-L. Vay, Noninvariance of Space- and Time-Scale Ranges under a Lorentz Transformation and the Implications for the Study of Relativistic Interactions, Phys. Rev. Lett. **98**, 130405 (2007).

[19] J.-L. Vay, I. Haber, and B. B. Godfrey, A domain decomposition method for pseudospectral electromagnetic simulations of plasmas, J. Comput. Phys. **243**, 260 (2013).

[20] R. Lehe, M. Kirchen, I. A. Andriyash, B. B. Godfrey, and J.-L. Vay, A spectral, quasicylindrical and dispersion-free particle-in-cell algorithm, Comput. Phys. Commun. **203**, 66 (2016).

[21] A. Edelen *et al.*, Neural network virtual diagnostic for the FAST low energy beam line, in *Proceedings of IPAC 2018, Vancouver, Canada* (JACoW, 2018), WEPAF040.

[22] A. Edelen *et al.*, Neural network based approaches to the modeling and control of particle accelerators, in *Proceedings of IPAC 2018, Vancouver, Canada* (JACoW, 2018), THYGBE2, https://accelconf.web.cern.ch/AccelConf/ipac2018/talks/thygbe2_talk.pdf.

[23] A. Edelen, J. Edelen, S. Biedron, S. Milton, and P. van der Slot, Using a neural network control policy for rapid switching between beam parameters in a FEL, in *Proceedings of the International Free Electron Laser Conference, Santa Fe, NM, USA* (JACoW, 2017), pp. 406–409, http://accelconf.web.cern.ch/AccelConf/fel2017/papers/wep031.pdf.

[24] A. Edelen, S. Biedron, J. Edelen, and S. Milton, First steps toward incorporating image based diagnostics into particle accelerator control systems using convolutional neural

networks, in *Proceedings of NAPAC 2016, Chicago, IL, USA* (JACoW, 2017), TUPOA51.

[25] A. Adelmann, On nonintrusive uncertainty quantification and surrogate model construction in particle accelerator modeling, SIAM/ASA J. Uncertainty Quantif. **7**, 383 (2019).

[26] A. Edelen *et al.*, Opportunities in machine learning for particle accelerators, *White paper from the 1st ICFA Machine Learning Workshop for Particle Accelerators*, https://arxiv.org/abs/1811.03172.

[27] A. L. Edelen, S. G. Biedron, B. E. Chase, D. Edstrom, S. V. Milton, and P. Stabile, Neural networks for modeling and control of particle accelerators, IEEE Trans. Nucl. Sci. **63**, 878 (2016).

[28] M. Conde *et al.*, Research program and recent results at the Argonne Wakefield Accelerator Facility (AWA), in *Proceedings of IPAC 2017, Copenhagen, Denmark* (JACoW, 2017), WEPAB132.

[29] A. Bungau *et al.*, Proposal for an Electron Antineutrino Disappearance Search Using High-Rate ^{8}Li Production and Decay, Phys. Rev. Lett. **109**, 141802 (2012).

[30] J. D. Meiss, Symplectic maps, variational principles, and transport, Rev. Mod. Phys. **64**, 795 (1992).

[31] A. Adelmann, P. Calvo, M. Frey, A. Gsell, U. Locans, C. Metzger-Kraus, N. Neveu, C. Rogers, S. Russell, S. Sheehy, J. Snuverink, and D. Winklehner, OPAL a versatile tool for charged particle accelerator simulations, arXiv:1905.06654.

[32] X. Pang and L. Rybarcyk, Multiobjective particle swarm and genetic algorithm for the optimization of the LANSCE linac operation, Nucl. Instrum. Methods Phys. Res., Sect. A **741**, 124 (2014).

[33] J. Kennedy and R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95—International Conference on Neural Networks* (1995), Vol. 4, pp. 1942–1948, https://doi.org/10.1109/ICNN.1995.488968.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. **6**, 182 (2002).

[35] G. Ha, M. H. Cho, W. Gai, K.-J. Kim, W. Namkung, and J. G. Power, Perturbation-minimized triangular bunch for high-transformer ratio using a double dogleg emittance exchange beam line, Phys. Rev. Accel. Beams **19**, 121301 (2016).

[36] W. Gai, , J. G. Power, and C. Jing, Short-pulse dielectric two-beam acceleration, J. Plasma Phys. **78**, 339 (2012).

[37] R. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications* (SIAM, Philadelphia, PA, 2014).

[38] V. Vapnik, *Statistical Learning Theory. Adaptive and Learning Systems for Signal Processing, Communications, and Control* (Wiley, New York, 1998).

[39] I. Sobol, Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, Math. Comput. Simul. **55**, 271 (2001).

[40] J. R. Alonso *et al.*, Medical isotope production with the IsoDAR cyclotron, Nat. Rev. Phys. **1**, 533 (2019).

[41] C. K. Allen and T. P. Wangler, Beam halo definitions based upon moments of the particle distribution, Phys. Rev. ST Accel. Beams **5**, 124202 (2002).

[42] F. Wang, M. Song, A. Edelen, and X. Huang, Machine learning for design optimization of storage ring nonlinear dynamics, arXiv:1910.14220.

[43] A. Edelen, J. Edelen, S. Biedron, S. Milton, and P. van der Slot, Using a neural network control policies for rapid switching between beam parameters in a FEL, in *NeurIPS Deep Learning for Physical Sciences Workshop (DLPS 2017), Long Beach, CA* (2017), https://dl4physicalsciences.github.io/files/nips_dlps_2017_16.pdf.

[44] A. Edelen, N. Neveu, C. Mayes, C. Emma, and D. Ratner, Machine learning models for optimization and control of x-ray free electron lasers, in *NeurIPS Machine Learning for the Physical Sciences Workshop* (2019), https://ml4physicalsciences.github.io/files/NeurIPS_ML4PS_2019_90.pdf.

[45] C. Gulliford *et al.*, Demonstration of low emittance in the Cornell Energy Recovery Linac injector prototype, Phys. Rev. ST Accel. Beams **16**, 073401 (2013).

[46] C. Emma, A. Edelen, M. J. Hogan, B. O'Shea, G. White, and V. Yakimenko, Machine learning-based longitudinal phase space prediction of particle accelerators, Phys. Rev. Accel. Beams **21**, 112802 (2018).

[47] C. E. Rasmussen and C. K. I. Williams, Bayesian regression and Gaussian processes, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge Massachusetts, 2006), Chap. 2.

[48] M. McIntire, T. Cope, S. Ermon, and D. Ratner, Bayesian optimization of FEL performance at LCLS, in *Proceedings of IPAC 2016, Busan, Korea* (JACoW, 2016), WEPOW055.

[49] M. McIntire, D. Ratner, and S. Ermon, Sparse Gaussian processes for Bayesian optimization, in *UAI'16 Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence* (2016), https://www-cs.stanford.edu/ermon/papers/sparse-gp-uai.pdf, pp. 517–526.

[50] K. Johannes, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause, Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces, *UAI'16 Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence* (2019), https://arxiv.org/pdf/1902.03229.pdf.

[51] J. Duris, D. Kennedy, A. Hanuka, J. Shtalenkova, A. Edelen, A. Egger, T. Cope, and D. Ratner, Bayesian Optimization of a Free-Electron Laser, arXiv:1909.05963 [Phys. Rev. Lett. (to be published)].

[52] A. Hanuka, J. Duris, J. Shtalenkova, D. Kennedy, A. Edelen, D. Ratner, and X. Huang, Online tuning and light source control using a physics-informed Gaussian process, *Machine Learning for the Physical Sciences Workshop, NeurIPS 2019, Vancouver CA*.

[53] A. Scheinker, A. Edelen, D. Bohler, C. Emma, and A. Lutman, Demonstration of Model-Independent Control of the Longitudinal Phase Space of Electron Beams in the Linac-Coherent Light Source with Femtosecond Resolution, Phys. Rev. Lett. **121**, 044801 (2018).

[54] A. Scheinker, D. K. Bohler, A. L. Edelen, R. W. Garnett, S. V. Milton, and D. Rees, Applying artificial intelligence to accelerators, *Proc. 9th International Particle Accelerator Conference (IPAC'18), Vancouver, BC, Canada* (JACoW Publishing, Geneva, Switzerland, 2018), pp. 2925–2928, https://doi.org/10.18429/JACoW-IPAC2018-THYGBE1.

[55] S. C. Leemann, S. Liu, A. Hexemer, M. A. Marcus, C. N. Melton, H. Nishimura, and C. Sun, Demonstration of Machine Learning-Based Model-Independent Stabilization

of Source Properties in Synchrotron Light Sources, Phys. Rev. Lett. 123, 194801 (2019).

[56] R. M. French, Catastrophic forgetting in connectionist networks, Trends Cognit. Sci. 3, 128 (1999).

[57] M. McCloskey and N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, Psychology of Learning and Motivation (Academic Press, New York, 1989), Vol. 24, pp. 109–165.

[58] J. Kirkpatrick, R. Pascanu, N. Rabinowitz et al., Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. U.S.A. 114, 3521 (2017).

[59] A. Edelen et al., Internal report, Fermilab, 2014.

[60] A. Edelen et al., Initial experimental results of a machine learning-based temperature control system for an rf gun, in Proceedings of IPAC 2015, Richmond, VA, USA (2015), MOPWI028, http://accelconf.web.cern.ch/AccelConf/IPAC2015/papers/mopwi028.pdf.

[61] Y. Gal and Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in ICML (2016), http://proceedings.mlr.press/v48/gal16.pdf.

[62] V. Kuleshov, N. Fenner, and S. Ermon, Accurate uncertainties for deep learning using calibrated regression, in ICML (2018), http://proceedings.mlr.press/v80/kuleshov18a/kuleshov18a.pdf.

[63] R. M. Neal, Bayesian Learning for Neural Networks (Springer Science and Business Media, New York, New York, 2012).

[64] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, Deep Kernel Learning (2016), http://arxiv.org/abs/1611.00336.

[65] A. C. Damianou and N. D. Lawrence, Deep Gaussian Processes, International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA, Vol. 31, http://proceedings.mlr.press/v31/damianou13a.pdf.

[66] Y. Ineichen, Toward massively parallel multiobjective optimization with application to particle accelerators, Ph. D. thesis, ETH Zurich, Zurich, Switzerland, 2013.

[67] LibEnsemble, https://libensemble.readthedocs.io/en/master/.

[68] J. L. Warren, G. P. Boicourt, M. T. Menzel, G. W. Rodenz, and M. C. Vasquez, Revision of and documentation for the standard version of the POISSON group codes, IEEE Trans. Nucl. Sci. 32, 2870 (1985).

[69] K. Halbach and R. F. Holsinger, SUPERFISH—A computer program for evaluation of rf cavities with cylindrical symmetry, Part. Accel. 7, 213 (1976).

[70] O. Kononenko et al., Advances in massively parallel electromagnetic simulation suite ACE3P, in Proceedings of International Computational Accelerator Physics Conference, ICAP 2015, Geneva, Switzerland (JACoW, 2015), FRAJI3.

[71] J. Yang, A. Adelmann, W. Barletta, L. Calabretta, A. Calanna, D. Campo, and J. M. Conrad, Beam dynamics simulation for the high intensity DAEdALUS cyclotrons, Nucl. Instrum. Methods Phys. Res., Sect. A 704, 84 (2013).

[72] F. Chollet et al., KERAS: The Python deep learning library (2015), https://keras.io.

[73] M. Abadi et al., TENSORFLOW: Large-scale machine learning on heterogeneous systems, 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (2016), pp. 265–283, https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[74] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, CoRR (2014), https://arxiv.org/abs/1412.6980.

[75] B. Debusschere, K. Sargsyan, C. Safta, and K. Chowdhary, Uncertainty quantification toolkit (UQTk), in Handbook of Uncertainty Quantification, edited by R. Ghanem, D. Higdon, and H. Owhadi (Springer International Publishing, Cham, 2017), pp. 1807–1827, https://doi.org/10.1007/978-3-319-12385-1_56.

[76] B. Debusschere, H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P. Le Maître, Numerical challenges in the use of polynomial chaos representations for stochastic processes, SIAM J. Sci. Comput. 26, 698 (2004).

[77] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliability Engineering System Safety 93, 964 (2008).

[78] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13, 2171 (2012).

[79] F. Pedregosa et al., SCIKIT-LEARN: Machine learning in Python, J. Mach. Learn. Res. 12, 2825 (2011).