ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-FRXE1

# BAYESIAN ALGORITHMS FOR PRACTICAL ACCELERATOR CONTROL AND ADAPTIVE MACHINE LEARNING FOR TIME-VARYING SYSTEMS

A. Scheinker[*], Los Alamos National Laboratory, Los Alamos, NM 87544, USA
R. Roussel[†], SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

*Abstract*

Particle accelerators are complicated machines with thousands of coupled time varying components. The electromagnetic fields of accelerator devices such as magnets and RF cavities drift and are uncertain due to external disturbances, vibrations, temperature changes, and hysteresis. Accelerated charged particle beams are complex objects with 6D phase space dynamics governed by collective effects such as space charge forces, coherent synchrotron radiation, and whose initial phase space distributions change in unexpected and difficult to measure ways. This two-part tutorial presents recent developments in Bayesian methods and adaptive machine learning (ML) techniques for accelerators. Part 1: We introduce Bayesian control algorithms, and we describe how these algorithms can be customized to solve practical accelerator specific problems, including online characterization and optimization. Part 2: We give an overview of adaptive ML (AML) combining adaptive model-independent feedback within physics-informed ML architectures to make ML tools robust to time-variation (distribution shift) and to enable their use further beyond the span of the training data without relying on re-training.

## INTRODUCTION

Particle accelerators are large complex systems whose beams evolve according to dynamics governed by nonlinear collective effects such as space charge forces and coherent synchrotron radiation. Because of their complexity, the control of charged particle beams in accelerators and diagnostics of these beams can greatly benefit from the application of machine learning (ML) [1, 2] methods and advanced control theory techniques [3].

The development of ML-based tools for particle accelerator applications is an active area of research. At CERN, supervised learning techniques are being applied for the reconstruction of magnet errors in the incredibly large (thousands of magnets) LHC lattice [4]. At the LCLS, Bayesian methods have been developed for online accelerator tuning [5], Bayesian methods with safety constraints are being developed at the SwissFEL and the High-Intensity Proton Accelerator at PSI [6], at SLAC Bayesian methods are being developed for the challenging problem of hysteresis [7] and surrogate models are being developed for the beam at the injector [8], and at LANL researchers have been developing methods to combine neural networks with model-independent adaptive feedback for automatic control of the $(z, E)$ longitudinal phase space (LPS) of intense short electron beams [9]. Convolutional neural networks (CNN) have been used to generate incredibly high resolution virtual diagnostics of the LPS of the electron beam in the EuXFEL [10]. A laser plasma wakefield accelerator has also been optimized by utilizing Gaussian processes at the Central Laser Facility [11].

In this tutorial we give a brief introduction to some machine learning methods including neural networks, Bayesian algorithms, and adaptive feedback.

## BAYESIAN METHODS

Bayesian optimization (BO) [12] is a model based optimization method that is well suited for online accelerator control problems [13–15]. BO consists of three components, shown in Fig. 1, a Bayesian statistical model of the objective function known as a Gaussian process (GP) [16], an acquisition function which characterizes the value of making potential observations and a numerical optimizer that optimizes the acquisition function to pick the highest valued point. This method excels at optimizing functions that are expensive to evaluate (such as quadrupole scan emittance measurements) because GP models provide uncertainty information when making predictions, allowing BO to balance exploration and exploitation when searching for global optima. Furthermore, GP models used in BO explicitly model noisy systems such as accelerators, making optimization less sensitive to jitter relative to other black box optimization algorithms.

Bayesian inference is the process of systematically updating prior statistical beliefs in the presence of experimental measurements. Imagine a parametric model $y = f(\mathbf{x};\theta)$, where we have collected training data pairs $D = \{X, \mathbf{y}\}$ and $\theta$ parameterized the model. Bayes rule applied to determining model parameters is given by

$$p(\theta|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \theta)p(\theta)}{p(X, \mathbf{y})} \qquad (1)$$

where $p(\theta)$ represents the *prior*, $p(\mathbf{y}|X, \theta)$ is the *likelihood* or *evidence*, $p(\theta|X, \mathbf{y})$ is the *posterior* and $p(X, \mathbf{y})$ is the *marginal likelihood* and $p(.|.)$ denotes a conditional probability. Bayes rule is useful in fitting model parameters due to the weighting of a prior distribution which regularizes predictive values of model versus the strength of experimental evidence. In the case of a uniform prior (no prior information is known) and a Gaussian likelihood (Gaussian statistical noise) Bayes rule reduces to non-linear least squares regression. Gaussian processes use Bayes' rule to predict objective function values based on approximate prior knowledge of

[*] ascheink@lanl.gov
[†] rroussel@slac.stanford.edu

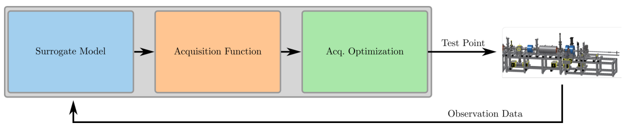

Figure 1: Major components of a general Bayesian Optimization algorithm.

the general behavior of the function (sensitivity to input parameters, structure etc.). This improves model accuracy (and as a result optimization performance) when experimental measurements of the objective function are limited.

A GP represents the function value at a given input point via a random variable drawn from a joint Gaussian distribution $g(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ where $\mu(\mathbf{x})$ is the mean and $k(\mathbf{x}, \mathbf{x}')$ is known as the kernel [16], where $x, x'$ represents points in the input space. To make a model prediction at the location $x_*$ we start with a prior Gaussian distribution with $\mu(\mathbf{x}) = 0$ (without loss of generality) and the covariance matrix given by $k(\mathbf{x}_*, x_*)$. We then condition the joint multivariate Gaussian distribution on the data set of $N$ observed points $\mathcal{D}_N = \{X, \mathbf{y}\}$. This gives us a probability distribution of the function value at the test point $g_* = g(\mathbf{x}_*)$ with expected noise $\sigma_n^2$ by

$$p(g_*|\mathcal{D}_N) \sim \mathcal{N}(\mu_*, \sigma_*^2) \tag{2}$$

$$\mu_* = \mathbf{k}^T[K + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y} \tag{3}$$

$$\sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^T[K + \sigma_n^2\mathbf{I}]^{-1}\mathbf{k} \tag{4}$$

$$\mathbf{k} = [k(\mathbf{x}_*, \mathbf{x}_1), k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N)] \tag{5}$$

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \tag{6}$$

We encode our prior knowledge about the general function behavior through our choice of kernel function. Examples of popular kernels include the radial basis function (RBF) and the Matern kernels. These kernels contain hyperparameters, such as the length scale parameter which controls the smoothness of function predictions, are fit to data via maximum likelihood estimation (MLE). These kernels can be generalized by using separate length scales for each design parameter, a process known as automatic relevance determination. Fitting independent length scales to each parameter through this process results in dimensionality reduction as model predictions will be largely invariant along dimensions that have long length scales.

Encoding prior information into the model through kernel construction and hyperparameters can have a significant positive impact on optimization performance, since it improves model accuracy in the absence of measurements [5]. For example, if we expect the function to have periodic or polynomial structure associated with it, we can combine kernels by adding or multiplying them to impose more accurate correlations in the input domain. Furthermore, prior information can also be encoded into kernels by specifying priors over their hyperparameters and using maximum *a posterori* (MAP) estimation or variational inference [17] to determine their posterior values given observed data.

The next step in BO is the calculation of the acquisition function $\alpha(\mathbf{x})$ which uses the GP model to quantify the value of making future measurements in input space. When attempting to find global optima of a function it is advantageous to value observations that balance exploitation (choosing points that are likely to be ideal) and exploration (choosing points that have high uncertainty). Examples of popular acquisition functions that achieve this are Expected Improvement (EI) [18] and Upper Confidence Bound (UCB) [19]. We can also customize acquisition functions to take into account a set of constraining functions $g_i(\mathbf{x}) \leq h_i$ by modeling constraining functions as separate GPs and then weighting the acquisition function according to the probability of satisfying those constraints

$$\hat{\alpha}(\mathbf{x}) = \alpha(\mathbf{x}) \prod_i p(g_i(\mathbf{x}) \leq h_i). \tag{7}$$

Furthermore, for accelerator control applications it is often necessary to reduce the size of jumps in parameter space in order to maintain stability in external feedback systems. To achieve this in BO we can bias the algorithm towards making small steps in input space

$$\bar{\alpha}(\mathbf{x}, \mathbf{x}_0) = \alpha(\mathbf{x}) \exp\left[-(\mathbf{x} - \mathbf{x}_0)^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}_0)\right] \tag{8}$$

where $\mathbf{x}_0$ is the last point measured and $\Sigma^{-1}$ controls the biasing strength [20].

The final step of BO is to optimize the acquisition function. As BO represents a nested optimization problem, the acquisition function should be cheap to optimize, often using gradient based optimization methods. In service of this, both the surrogate model and the acquisition function should be *differentiable*, ie. the gradient with respect to input parameters should be analytically calculable or cheap to calculate via differentiable calculations commonly used in machine learning languages. In cases where acquisition functions are defined by complicated integrals that do not have analytical solutions, monte carlo methods using the reparameterization trick [21] can be used.

A simple example of BO based maximization of a 1D function is shown in Fig. 2. At each step a GP model is created based on the observed data. Then the acquisition function is fed the GP model and is numerically optimized to choose the next observation point. Initially the model has high uncertainty and thus BO attempts to reduce uncertainty by sampling points at the domain boundaries. Eventually a location is found that is significantly more optimal than previous measurements. Once found the BO algorithm attempts to exploit the model near the perceived optimum of the function to find the extrema.

There are several easy to use implementations of BO for a variety of applications. The python library *BoTorch* [22] is recommended as it is based on the well established machine learning library *PyTorch* and it has a robust community of

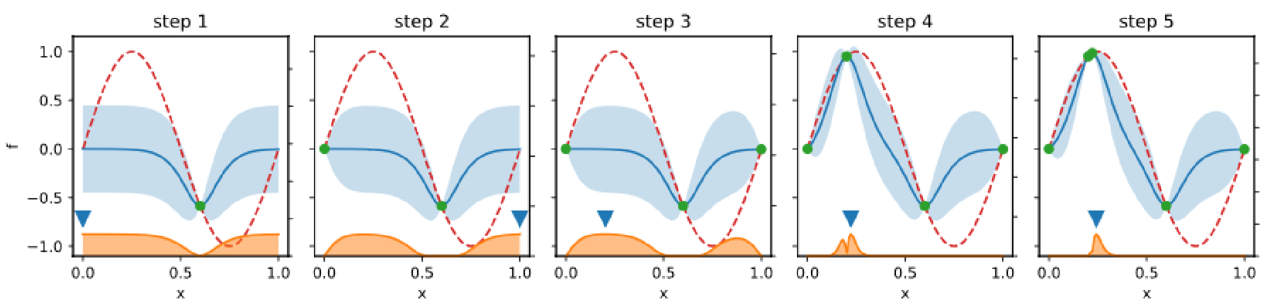

Figure 2: Example BO of a simple function using the Expected Improvement acquisition function. The ground truth function is (dotted red line) is optimized over 5 iterations. The blue line and shading denotes the mean prediction and 95% confidence bounds of the GP model based on measurements (blue dots). The acquisition function (orange) is maximized (blue triangle) to select the next observation.

developers. Alternatively, for most basic BO applications the Xopt [23] python library has been developed by the SLAC machine learning group for both experimental and computational optimization. It is currently used at several accelerator facilities including the Linac Coherent Light Source and the Argonne Wakefield Accelerator and has been used at high performance computing facilities such as NERSC.

## NEURAL NETWORKS

The ability of neural networks to approximate functions arbitrarily accurately is a famous result [24] based on the Stone-Weierstrass Theorem [25]. The result guarantees that for any measurable function $\mathbf{f}(\mathbf{x}) : \mathbf{R}^n \longrightarrow \mathbf{R}^{n_2}$, any compact set $K \subset \mathbf{R}^n$, and any $\epsilon > 0$ it is possible to create a neural network-based approximation $\hat{\mathbf{f}}$ such that

$$\sup_{\mathbf{x}\in K} \|\mathbf{f}(\mathbf{x}) - \hat{\mathbf{f}}(\mathbf{x})\| < \epsilon. \tag{9}$$

Intuitively this result should not be too surprising due to the well known fact that any function in the Hilbert space $L^2[0,T]$ can be represented arbitrarily accurately as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} w_i \psi_i(\mathbf{x}), \tag{10}$$

for various choices of basis functions $\{\psi_i\}_{i\in\mathbf{N}}$, such as the well known Fourier series or Legendre polynomials.

Considering $n_2 = 1$ for notational simplicity, the first step of a NN is linear regression: an inner product of an input vector $\mathbf{x}$ with a set of input weights $\mathbf{w}_0$ and the addition of a bias, followed by applying a nonlinear activation function

$$\mathbf{x} \longrightarrow b_0 + \mathbf{w}_0^T \mathbf{x} \longrightarrow f_0\left(b_0 + \mathbf{w}_0^T \mathbf{x}\right) = y_0. \tag{11}$$

Instead of forming just one single output $y_0$, if we multiply $\mathbf{x}$ by a matrix of weights, $W_0$, and add a vector of biases, $\mathbf{b}_0$ where the number of weight vectors, $m$, is known as the width of the NN's layer. A nonlinear activation function is applied to each element resulting in a vector $\mathbf{y}_0$ of $m$ outputs

$$y_{0j} = f_0\left(\mathbf{w}_{0j}^T \mathbf{x} + b_{0j}\right), \tag{12}$$

where $\mathbf{w}_{0j}^T = \left(w_{0j1}, \ldots, w_{0jn}\right)$. A linear combination of the outputs and a bias is then formed giving the output

$$\hat{y}(\mathbf{x}) = \sum_{j=1}^{m} w_{1j} f_0\left(\sum_{k=1}^{m} w_{0jk} x_k + b_{0j}\right) + b_1, \tag{13}$$

which is just a particular case of (10).

The weights and biases are adjusted to minimize an error

$$C = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2 + \epsilon_w \|\mathbf{w}\|_2 + \epsilon_b \|\mathbf{b}\|_2, \tag{14}$$

which penalizes not only the NN's accuracy, but also the $l^2$ norms of the weights and biases for regularization. Training is a gradient descent of $C$ with respect to weights and biases:

$$w_{ijk} \longrightarrow w_{ijk} - \delta \left\langle \frac{\partial C}{\partial w_{ijk}} \right\rangle, \tag{15}$$

where the learning rate $\delta \ll 1$ is fixed or may be adaptively updated and the gradient is an average over a batch of random combinations of input-output pairs of data.

Instead of building wide single layer NNs, a better approach is to increase the network's depth by adding layers where the vector of a previous layer's outputs is multiplied by a matrix of weights and added to a vector of biases and again a set of nonlinear activation functions is applied.

### *Adaptively Tuned Latent Space of Physics-Informed Encoder-Decoder Convolutional Neural Networks for Time-Varying Systems*

The most powerful ML tools for working directly with high dimensional data, such as images, are convolutional neural networks (CNN). When a $N \times N$ input image matrix

$$I^0_{i_0,j_0}, \quad i_0, j_0 \in \{1, 2, \ldots, N\},$$

passes through a stride 2 convolutional layer with a $3\times3$ filter $F_{0,ij}$ the output image size is reduced by a factor of 4 resulting in a $N/2 \times N/2$ image $I^1$ with pixels defined as

$$I^1_{i_1,j_1} = f\left(b^0 + \sum_{i=-1}^{1}\sum_{j=-1}^{1} F_{0,ij} \times I^0_{i_0+i,j_0+j}\right), \tag{16}$$

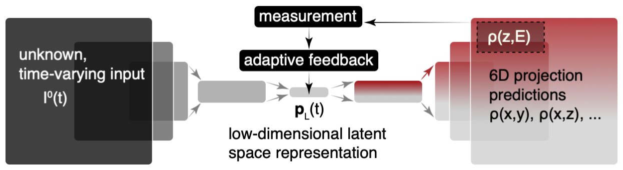
measurement
adaptive feedback
unknown, time-varying input
I⁰(t)
p_L(t)
low-dimensional latent space representation
ρ(z,E)
6D projection predictions
ρ(x,y), ρ(x,z), ...

Figure 3: Schematic of an adaptive CNN-based encoder-decoder for a system with unknown time-varying inputs.

with extra zeros padded onto the image as needed. A collection of $N_f > 1$ filters is used, resulting in an output image

$$I^1_{i_1,j_1} = b^1 + \sum_{n=1}^{N_f} w_n \times f_n \left( b^0_n + \sum_{i=-1}^{1}\sum_{j=-1}^{1} F_{0,ij,n} \times I^0_{i_0+i,j_0+j} \right). \quad (17)$$

For a wide encoder-decoder CNN with several layers the number of adjustable parameters quickly grows to millions and re-training requires large collections of new data sets.

One of the major limitations of machine learning is an inability to deal with time-varying systems, or systems with distribution shift [26–30]. Attempts to compensate for changing environments rely on techniques such as domain transfer, transfer learning, and active learning, which are all forms of re-training with new data. For particle accelerators re-training usually means lengthy beam interruption to collect new invasive data sets, and for most interesting problems is not possible.

Here we present an adaptive alternative for cases in which re-training is not possible because we no longer have access to the time-varying input distribution $I^0(t)$. Our approach is to use several layers of convolutions to significantly reduce the size of an image, ending up with a tensor of shape $N_i \times N_i \times N_f$ where $N_i \times N_i$ is the final image size and $N_f$ is the number of filters in the last convolution layer of the encoder. We can then flatten the image and apply dense fully connected layers to reach a low-dimensional latent space representation, a vector $\mathbf{v}_L$ of length $N_L \ll N_{i,p}$ which can be adaptively tuned by a $N_L$ dimensional control input vector $\mathbf{v}_{L,c}$, which is then passed through additional fully connected dense layers before being reshaped into a small image ($\sim 8\times 8$) before flowing through a series of 2D transpose convolution layers until a collection of $N_o$ output images of size $N_{im} \times N_{im}$ are generated in a final layer with $N_c$ channels with size $\hat{I}(i,j,d) = N_{im} \times N_{im} \times N_c$. Once the network is trained this collection of output images is a general nonlinear function (the generative branch of the network) of the parameters $\mathbf{p}_L$ of the form

$$\hat{I}_{i,j}(d) = \mathbf{F}\left(\mathbf{p}_L, \mathbf{w}, \mathbf{b}, \{A\}\right), \quad (18)$$

where $\mathbf{w}$ and $\mathbf{b}$ are the weights and biases and $\{A\}$ are the set of activation functions of the generative layers. Our prediction $\hat{I}$ is our estimate of some unknown physical quantity $I$ which we assume we cannot easily directly measure fully. In order to enable the adaptive feedback part of this procedure we must assume that we have some form of non-invasive online measurement of $M(I(t))$ that can be compared to a

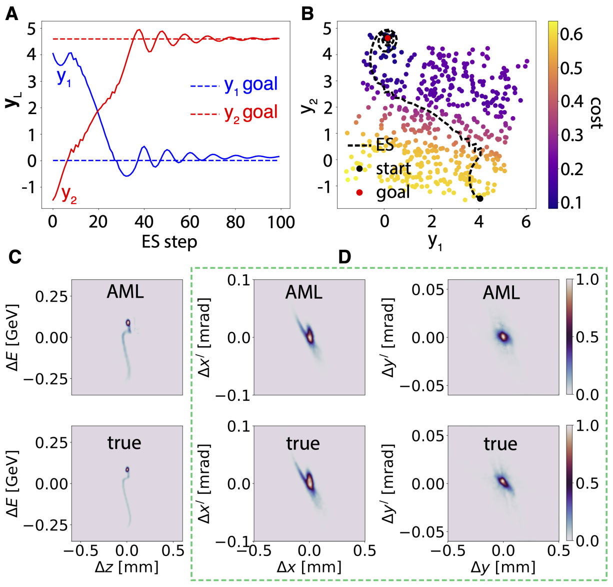

Figure 4: (A, B): ES convergence in the 2D latent space to match the LPS distribution. C: AML-based adaptive match of LPS distribution. D: Matched transverse phase space distributions as a result of matching the LPS (other 12 projections also matched closely, are not shown).

simulated measurement of our prediction $\hat{M}(\hat{I})$, which we know how to approximate as shown in Fig. 1.

For example we may be interested in a 6D density distribution but we only have direct measurements of the $\rho(z,E)$ 2D projection. The generative half of our CNN may then be used to generate estimates of all 15 unique projections of a beam's 6D phase space $\{\hat{\rho}_1(x,y),\ldots,\hat{\rho}_{15}(z,E)\}$.

By forcing the CNN to simultaneously generate all 15 projections of the 6D phase space we introduced observational biases directly through data embodying the underlying physics, allowing the CNN to learn functions that reflect the physical structure of the data. Additional physics constraints are enforced by a PINNs approach in which the CNN's training cost function is of the form:

$$C_{NN} = \epsilon_M \sum_i \left\| M_i - \hat{M}_i \right\| + \epsilon_D \sum_i \left\| \frac{\partial \hat{M}_i}{\partial t} - D\hat{M}_i \right\|, \quad (19)$$

where the first term is standard supervised training and the second enforces that the CNN satisfy a PDE operator $D$ [31].

Our dynamic feedback minimizes the cost

$$C(\mathbf{p},t) = \int_z \int_E \left(\rho_{15}(z,E,t) - \hat{\rho}_{15}(z,E,t)\right)^2 dE\,dz, \quad (20)$$

where the measurement $\rho_{15}(z,E,t)$ is time-varying due to uncertain time-varying input beam distributions and accelerator parameters. The latent space is tuned according to

$$\frac{dp_{L,i}(t)}{dt} = \sqrt{\alpha\omega_i}\cos\left(\omega_i t + kC(t)\right), \quad (21)$$

where the feedback dynamics (21) are chosen based on a recently developed form of adaptive feedback control known as extremum seeking (ES) which was designed for the stabilization and optimization of analytically unknown nonlinear time-varying systems [32], which results in a minimization of $C$ by tracking the time-varying $M(t)$ with the approximation $\hat{M}(t)$ according to average dynamics of the form:

$$\frac{d\bar{\mathbf{p}}_L}{dt} = -\frac{k\alpha}{2}\nabla_{\bar{\mathbf{p}}_L} C(\bar{\mathbf{p}}_L, t). \tag{22}$$

Because the CNN is physics-informed, it is possible to predict un-measured 2D phase space projections by tracking just the single measured $(z, E)$ distribution as the network has learned the correlations between the various dimensions of the 6D phase space. Such an approach was recently demonstrated for tracking the time varying input beam of the HiRES UED [33], and the 2D projections of the time-varying 6D phase space of charged particle beams in particle accelerators as shown in Fig. 4 adapted from [34].

Creating custom deep learning models, including deep neural networks, CNN's, and variational autoencoders is easily done via powerful open-source software packages such as TensorFlow [35].

## CONCLUSIONS

Bayesian optimization-based methods as well as neural networks are powerful tools for online accelerator controls and diagnostics. While Bayesian methods provide uncertainty quantification and can guide intelligent data collection CNNs are able to handle and generate incredibly high dimensional objects such as images directly. Bayesian neural networks attempt to combine the strengths of NNs and Bayesian methods to provide uncertainty quantification of NN-based models [36]. Furthermore, combining model-independent adaptive feedback techniques with physics-informed ML tools has proven to help make them more robust to uncertainty, time-variation, and distribution shift.

## ACKNOWLEDGEMENTS

Work funded by US Department of Energy (DOE) and the Los Alamos National Laboratory LDRD Program Directed Research (DR) project 20220074DR.

## REFERENCES

[1] U. Gentile and L. Serio, "A machine-learning based methodology for performance analysis in particles accelerator facilities," in *Proc. EECS 2017*, Bern, Switzerland, Nov. 2017, pp. 90–95. doi:10.1109/EECS.2017.26

[2] A. L. Edelen, S. Biedron, B. Chase, D. Edstrom, S. Milton, and P. Stabile, "Neural networks for modeling and control of particle accelerators," *IEEE Trans. Nucl. Sci.*, vol. 63, no. 2, pp. 878–897, 2016. doi:10.1109/TNS.2016.2543203

[3] A. Scheinker and M. Krstić, "Minimum-seeking for CLFs: Universal semiglobally stabilizing feedback under unknown control directions," *IEEE Trans. Autom. Control*, vol. 58, no. 5, pp. 1107–1122, 2012. doi:10.1109/TAC.2012.2225514

[4] E. Fol, R. Tomás, and G. Franchetti, "Supervised learning-based reconstruction of magnet errors in circular accelerators," *Eur. Phys. J. Plus*, vol. 136, no. 4, p. 365, 2021. doi:10.1140/epjp/s13360-021-01348-5

[5] J. Duris, D. Dylan, M. McIntire, and D. Ratner, "Bayesian optimization at LCLS using Gaussian processes," Daejeon, Korea, 61, presented at HB2018 in Daejeon, Korea, unpublished, 2018.

[6] J. Kirschner, M. Mutný, A. Krause, J. Coello de Portugal, N. Hiller, and J. Snuverink, "Tuning particle accelerators with safety constraints using bayesian optimization," *Phys. Rev. Accel. Beams*, vol. 25, no. 6, p. 062 802, 2022. doi:10.1103/PhysRevAccelBeams.25.062802

[7] R. Roussel *et al.*, "Differentiable preisach modeling for characterization and optimization of particle accelerator systems with hysteresis," *Phys. Rev. Lett.*, vol. 128, no. 20, p. 204 801, 2022. doi:10.1103/PhysRevLett.128.204801

[8] L. Gupta, A. Edelen, N. Neveu, A. Mishra, C. Mayes, and Y.-K. Kim, "Improving surrogate model accuracy for the LCLS-II injector frontend using convolutional neural networks and transfer learning," *Mach. Learn.: Sci. Technol.*, vol. 2, no. 4, p. 045 025, 2021. doi:10.1088/2632-2153/ac27ff

[9] 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.*, vol. 121, no. 4, p. 044 801, 2018. doi:10.1103/PhysRevLett.121.044801

[10] J. Zhu, Y. Chen, F. Brinker, W. Decking, S. Tomin, and H. Schlarb, "High-fidelity prediction of megapixel longitudinal phase-space images of electron beams using encoder-decoder neural networks," *Phys. Rev. Appl.*, vol. 16, no. 2, p. 024 005, 2021. doi:10.1103/PhysRevApplied.16.024005

[11] R. Shalloo *et al.*, "Automation and control of laser wakefield accelerators using Bayesian optimization," *Nat. Commun.*, vol. 11, no. 1, pp. 1–8, 2020. doi:10.1038/s41467-020-20245-6

[12] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, "Bayesian Optimization for Adaptive Experimental Design: A Review," *IEEE Access*, vol. 8, pp. 13 937–13 948, 2020, Conference Name: IEEE Access. doi:10.1109/ACCESS.2020.2966228

[13] J. Duris *et al.*, "Bayesian optimization of a free-electron laser," *Phys. Rev. Lett.*, vol. 124, p. 124 801, 12 2020. doi:10.1103/PhysRevLett.124.124801

[14] R. Roussel, A. Hanuka, and A. Edelen, "Multiobjective bayesian optimization for online accelerator tuning," *Phys. Rev. Accel. Beams*, vol. 24, no. 6, p. 062 801, 2021. doi:10.1103/PhysRevAccelBeams.24.062801

[15] S. A. Miskovich *et al.*, "Online bayesian optimization for a recoil mass separator," *Phys. Rev. Accel. Beams*, vol. 25, no. 4, p. 044 601, 2022. doi:10.1103/PhysRevAccelBeams.25.044601

[16] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Am. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, 2017. doi:10.1080/01621459.2017.1285773

[18] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. NIPS 2012*, vol. 25, 2012, pp. 2951–2959.

[19] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. ICML 2010*, 2010, pp. 1015–1022.

[20] R. Roussel and A. Edelen, "Proximal biasing for bayesian optimization and characterization of physical systems,"

[21] J. T. Wilson, R. Moriconi, F. Hutter, and M. P. Deisenroth, "The reparameterization trick for acquisition functions," 2017. doi:10.48550/ARXIV.1712.00424

[22] M. Balandat *et al.*, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Proc. NeurIPS 2020*, 2020.

[23] C. Mayes, R. Roussel, J. Garrahan, and H. Slepicka, *ChristopherMayes/Xopt: Xopt v1.1.1*, 2022. doi:10.5281/zenodo.6946250

[24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989. doi:10.1016/0893-6080(89)90020-8

[25] G. B. Folland, *Real analysis: modern techniques and their applications*. John Wiley & Sons, 1999, vol. 40.

[26] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. Mit Press, 2008.

[27] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodrıguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, no. 1, pp. 521–530, 2012. doi:10.1016/j.patcog.2011.06.019

[28] M. Sugiyama and M. Kawanabe, *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.

[29] Y. Ovadia *et al.*, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Proc. NeurIPS 2019*, 2019.

[30] A. Subbaswamy and S. Saria, "From development to deployment: Dataset shift, causality, and shift-stable models in health ai," *Biostat.*, vol. 21, no. 2, pp. 345–352, 2020. doi:10.1093/biostatistics/kxz041

[31] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422–440, 2021. doi:10.1038/s42254-021-00314-5

[32] A. Scheinker and D. Scheinker, "Bounded extremum seeking with discontinuous dithers," *Automatica*, vol. 69, pp. 250–257, 2016. doi:10.1016/j.automatica.2016.02.023

[33] A. Scheinker, F. Cropp, S. Paiagua, and D. Filippetto, "An adaptive approach to machine learning for compact particle accelerators," *Sci. Rep.*, vol. 11, no. 1, pp. 1–11, 2021. doi:10.1038/s41598-021-98785-0

[34] A. Scheinker, "Adaptive machine learning for time-varying systems: Low dimensional latent space tuning," *J. Instrum.*, vol. 16, no. 10, P10008, 2021. doi:10.1088/1748-0221/16/10/P10008

[35] M. Abadi *et al.*, "TensorFlow: A system for Large-Scale machine learning," in *Proc. OSDI 16*, 2016, pp. 265–283. doi:10.48550/arXiv.1603.04467

[36] E. Goan and C. Fookes, "Bayesian neural networks: An introduction and survey," in *Case Studies in Applied Bayesian Data Science*, 2020, pp. 45–87.