A PROPOSAL FOR OPTIMIZING DEVICE DESIGN WITH BAYESIAN OPTIMIZATION

A PREPRINT

Nathan O. Lambert, Kristofer S.J. Pister
Department of Electrical Engineering and Computer Sciences
The University of California, Berkeley
Berkeley, CA 94720, United States
{nol,ksjp}@berkeley.edu

December 28, 2020

ABSTRACT

This proposal is a working document detailing how Bayesian Optimization, and similar 0^{th} order (derivative free) optimizers can be used to be used in conjunction with expert knowledge when designing devices. Micro-electromechanical systems (MEMs) are physical devices that often fabricated in a traditional clean-room environment, so each design iteration is costly financially and in wall time. General hyperparameter optimization has been successful across numerous digital domains, but translation to more physical problems is primarily limited by overlap of expertise. This document includes a running example, motivation for the work, and future directions for the joining of cutting edge hardware design with parameter optimization.

1 Introduction

Scaling design problems to high dimensions is a challenging task for humans. In many fields, experts make design changes over few iterations based on internal heuristics. Recent advances in optimization techniques have enabled many advancements in data-driven assistance of design, including materials research, circuit design, animations, and more [1, 2, 3, 4]. In this work, we propose an additional arm to this data-driven design: aiding in the fabrication of micro-electromechanical systems (MEMs). MEMs are frequently optimized by tuning a subset of many design parameters in silicon processing.

There are many candidate methods for optimizing a parameter space – they vary widely, including in sample efficiency, dimensionality scaling, complexity of use, and types of models. In this work we focus on Bayesian Optimization (BO) because of it's **sample efficiency** and **structured uncertainty estimation** (model). BO fits a smooth model to a target function that approximates a costly experiment, which is optimized iteratively in a sample efficient manner.

Translating results in BO and other optimization techniques to physical domains holds strong promise, but is often limited by the availability of experts in both fields collaborating. This proposal covers the background of parameter optimization and examples in MEMs design: gap-closing actuators, a thermo-mechanical valve, and a jumping silicon robot. Associated code for this project can be found at `https://github.com/natolambert/mems-bo`.

2 Related Works

2.1 Parameter Optimization

The modern history of optimization algorithms can be traced to Evolutionary Algorithms (EA) [5, 6]. One type of EA are genetic algorithms, originating from a problem solved with a "bit-flipping" technique of gene sequences [7]. Modern development of parameter optimization mostly falls in hyperparameter tuning for machine learning models [8], where even random search can be effective for purely digital problems [9] Other methods such as quasi-random Sobol

searches [10], Covariance matrix adaptation evolution strategy (CMA) [11], or the cross-entropy method (CEM) [12] are popular alternatives.

2.2 Bayesian Optimization

Bayesian optimization is a relatively young optimization technique that is popular with expensive-to-evaluate functions [13]. Bayesian optimization does not require any assumption on model structure [14] and the variant we deploy is detailed in Sec. 3. BO is a sequential design framework that can utilize multiple models and activation functions to suit the desired problem. The Bayesian Optimization framework has been successfully deployed in multi-task optimization [15], tuning of algorithms [16], integrating constraints towards robotic tasks [17], learning locomotion tasks, [18], modeling dynamics [19], and more.

2.3 Optimization for Design

Recent advances in machine learning have changed the landscape of what tasks can be aided by data-driven optimization. Automated Machine Learning (AutoML) has proliferated over recent years in the digital domain to tune parameters in a data-efficient manner. One sub-area is neural architecture search (NAS) [20, 21, 2], specializing in the optimization of neural network models. Historically, optimizing hardware has been more difficult because the cost of evaluating the function-to-optimize is much higher. Some successful examples include the optimization of novel materials [1], analog circuits [3, 22, 23], and high-resolution animations [4].

3 Background

Problem Formulation Bayesian optimization is an iterative approach to find the global solution to the maximization (or minimization) of the form

$$x^* = \arg\max_{x \in \mathcal{X}} f(x) \tag{1}$$

Here $\mathcal{X}$ represents the parameter space being optimized, and $f(x) \in \mathcal{R}$ is the optimized function. Through an iterative process, we collect a dataset $\mathcal{D} : \{(x_i, y_i)\}_{i=1}^{N}$ where N is the number of samples, and y_i is a true objective measurement.

Gaussian Processes Gaussian Processes are non-parametric Gaussian fits to data. A Gaussian Process, $\mathcal{GP}(\mu_\theta, k_\theta)$, is defined by a mean vector, $\mu_\theta : \mathcal{X} \mapsto \mathcal{R}$, and a covariance function (as opposed to a matrix), $k_\theta(x_1, x_2) : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$. The subscript θ indicates values that are learned from data.

Probabilistic Inference Probabilistic inference is an important tool for Bayesian Optimization because it is the underlying mathematics for choosing a new sample – *a new sample is chosen based on a prior distribution over an objective value*. This section follows an example from [14] explaining the notion of probabilistic inference with linear models. Bayesian Optimization with Gaussian Processes follows the same approach, but with nonparametric models capturing noise in the data. Using Bayes rule, a Gaussian Process can be used for probabilistic inference to a) *choose* a new set of parameters or b) *measure* the likelihood of past data. We consider the latent variable of the model, x with a prior distribution $p(x)$ capturing the *a priori* belief about probable values of x. With data $\mathcal{D}$ and a *likelihood model* $p(\mathcal{D}|x)$, we infer the *a posterior* distribution over the parameters $p(x|\mathcal{D})$:

Algorithm 1: Bayesian Optimization with a $\mathcal{GP}$

for $n \leftarrow 1$ **to** $\cdots$: **do**
- sample initial state s_0 from $\widetilde{\mathcal{D}}$;
- $x_{n+1} \leftarrow \arg\max_x \alpha(x; \mathcal{D}_n)$ `// Optimize acquisition function on model`
- $y_{n+1} \leftarrow f(x_{n+1}))$ `// evaluate new parameters on real experiment`
- $\mathcal{D}_{n+1} \leftarrow \{\mathcal{D}_n, (x_{n+1}, y_{n+1})\}$ `// Append data`
- $\mathcal{GP} \leftarrow \mathcal{D}_n$ `// Update model`

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p(x)}{p(\mathcal{D})} \tag{2}$$

When fitting a function, the maximum likelihood estimate (MLE) of this posterior distribution represent the best fit to our sampled goal function to optimize.

Bayesian Optimization BO is a generic, iterative framework for optimizing a costly objective function. For some number of iterations, it selects a new sample x_n by optimizing a weighted approximation of the objective function called the acquisition function, α, that handles the exploitation versus exploration trade-off inherent to any approximate optimization (candidates for the acquisition function include expected improvement (EI), Thompson sampling (TS), probability of improvement, entropy search (ES), and more. Next, the selected parameters are measured on the true system, y_n. With the new data, some model is fit to the data and the process repeats.

Bayesian Optimization with Gaussian Processes Optimizing experimental data accurately requires capturing the uncertainty in the data. Now, assume the observations from the true model $f(p)$ is obscured with variance σ^2. Without loss of generality, we place a zero-mean Gaussian prior on the model parameters $p(x|V_0)$.

In this section we include more verbose notation – the Gaussian Processes, $\mathcal{GP}$, are parametrized as follows. The covariance matrix, $K^{\theta}_{i,j}$, and mean vector are computed across the previously logged datapoints as:

$$K^{\theta}_{i,j} = k(x_i, x_j) \tag{3}$$

$$m_i = \mu_0(x_i) \tag{4}$$

Please note that many different covariance functions (kernels) can be used. Kernels are used to project the data into a more useful function space and reduce the dimensionality of data[1].

With $\mathcal{D}_n = \left\{(x_i, y_i)\right\}_{i=1}^{n}$, the $\mathcal{GP}$ is updated at each index of optimization with a two-step function fitting.

$$\mu_n(x) = \mu_0(x) + k(x, x_{1:n})^{\top}(K + \sigma^2 I)^{-1}(y - m) \tag{5}$$

$$\sigma^2_n(x) = k(x, x) - k(x, x_{1:n})^{\top}(K + \sigma^2 I)^{-1}k(x, x_{1:n}) \tag{6}$$

The one time matrix inversion at each step in Eq. (5) and Eq. (6) indicated a matrix inversion of computational cost $\mathcal{O}(n^3)$, which is costly with large number of points (and makes running these systems in real-time control challenging) [24].

With the fitted model, then we need to recall how to use this for prediction. The model is a fitted Gaussian process of the form shown in Eq. (7), which serves as a prior for predicting the optimization function shown in Eq. (8):

$$f|X \sim \mathcal{N}(m, K) \tag{7}$$

$$y|f, \sigma^2 \sim \mathcal{N}(f, \sigma^2 I) \tag{8}$$

The acquisition function used in this work is expected improvement, which maximizes the gain over the current maximum value, f' (the expectation can be expanded as a integral over the Gaussian probability), as

$$\alpha(X) = \mathbb{E}\left[u(x)|x, \mathcal{D}\right], \tag{9}$$

$$u(x) = \max\left((0, f(x) - f'\right). \tag{10}$$

This acquisition function is repeatedly sampled over to get a new candidate value for evaluation on the expensive system (such optimization of the model can be done by less sample efficient methods such as CEM or CMA-ES).

4 Optimal Parameter Search

We now detail more explicitly how BO can be used for parameter search in a design problem. First, we define the state-space and the decision variables, with an example shown in Tab. 1. Optimal parameter search in this case entails using SOBOL search (near uniform grid-search) or previously human-optimized values to initialize the search, followed by expected improvement over the Gaussian process model. This section details some additional experimental tools that can aid in design. For example, BO can accommodate multi-modal data such as floats, integers, or Booleans, but more powerful tools such as context measurements and constraints can be added.

Parameter	Max	Min	Data Type
a	6	87837	float
b	7	78	integer
c	545	778	float
d	545	18744	float
e	0	1	Boolean

Table 1: Sample state-space for optimization.

An important option for data-driven design optimization is the inclusion of context variables, $\boldsymbol{c}$. In this case, each device can have additional measurements (such as input resistance, mass, or more intrinsic parameters post-manufacturing) that

[1]For more on kernels see `https://en.wikipedia.org/wiki/Kernel_method`.

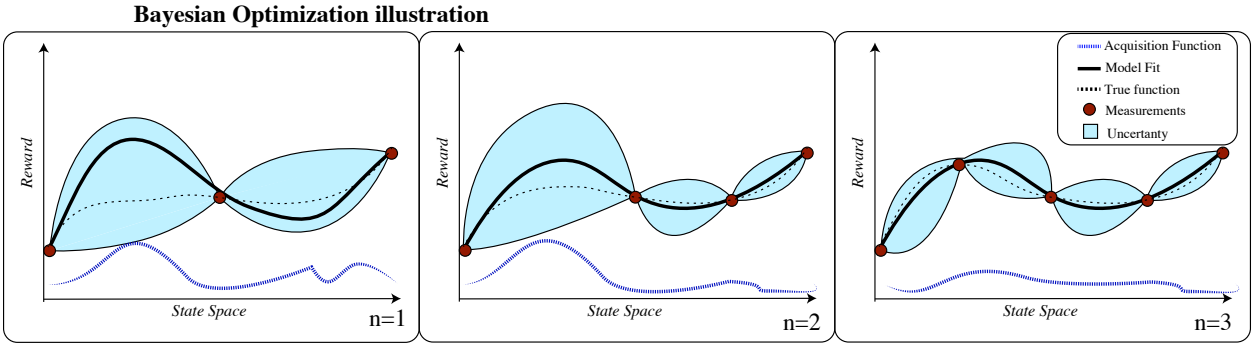


Figure 1: An illustrative example of Bayesian Optimization with key components: acquisition function, measurement, model fit & uncertainty, and the true function.

can be used to condition the $\mathcal{GP}$. Such a contextual optimization will fit a model $\mathcal{GP}(\mu_\theta, k_\theta; \boldsymbol{c})$, where the optimization only acts on the design parameters.

Bayesian optimization also has the capacity to deal with constraints, such as forces that would shear or break parts of the system. BO rejects samples suggested by the acquisition function when they fall out of the defined design space. Note that this process is imperfect because the rejection comes post sampling, so in some problems the sampler can have difficulty choosing new candidate designs if the constraints are difficult to satisfy. Only certain linear constraints work with the provided framework (e.g. less than or equal to), please see `https://ax.dev` for more documentation.

5 Learned Models as a Verification Method

After the initial phase of design optimization, the learned model can be used to empirically verify the fabrication variance over time. Given an analytical model of parameters

$$g = f(\boldsymbol{p}), \tag{11}$$

our method is to learn a model to fit this as

$$\hat{g} = f_\theta(\boldsymbol{p}). \tag{12}$$

Now, for each batch of experiments – which can have parameters different from optimal – we can compute the likelihood of a experimental point conditioned on $\hat{g}$. We consider the likelihood of a sampled point, but also consider other formulations. The log-likelihood of a Multi-variate Gaussian distribution is

$$\ln L(\hat{g}, g) = -\frac{1}{2}\left[\ln(|\Sigma|) + (\hat{g} - g)^\top \Sigma^{-1}(\hat{g} - g) + k\ln(2\pi)\right]. \tag{13}$$

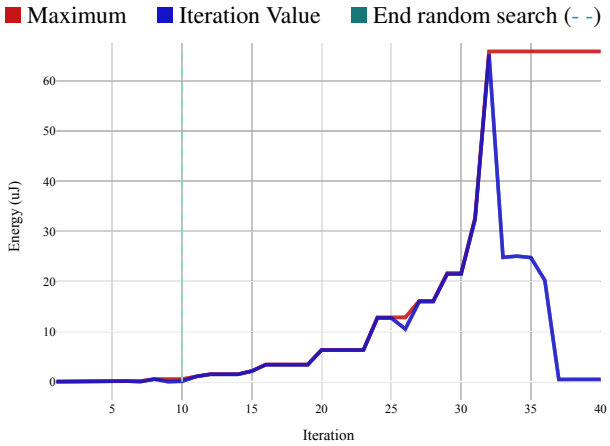
For a Gaussian Process non-parametric model, $\mathcal{GP}(\mu_\theta, K^\theta)$, this becomes

$$\log p(y|x_{1:n}, \theta) = -\frac{1}{2}(y - \mu_\theta)^\top (K^\theta + \sigma^2 I)^{-1}(y - \mu_\theta) - \frac{1}{2}\log|K^\theta + \sigma^2 I| - \frac{n}{2}\log(2\pi). \tag{14}$$

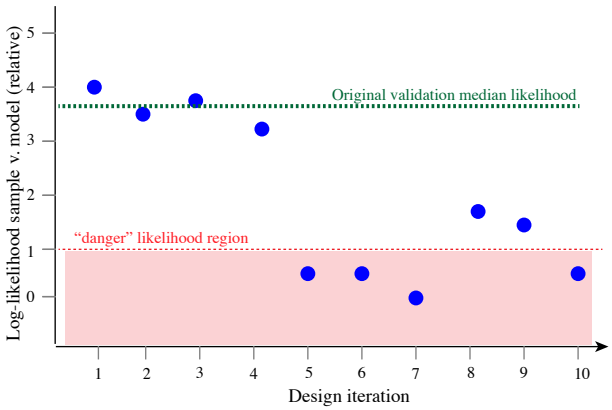
When measuring a new datapoint, we can numerically return a likelihood that the point is sampled from the measured model. This serves as a probe into both model and manufacturing uncertainty.

Model Updating When evaluating the effectiveness of a prior model to the measured data, we can use this tool to evaluate regions of candidate parameters. For example, consider a parameter set y^* derived by optimizing an offline analytical model $f(p)$. With a set of experimental data $\mathcal{D}$, we can fit a $\mathcal{GP}$ to the data and evaluate the likelihood of points from our analytical model relative to the measured data $\ln L(y^*, \mathcal{GP})$. When $\ln L(y^*, \mathcal{GP})$ is low, the analytical model should be reconsidered in the region of the state-space that y^* occupies. When $\ln L(y^*, \mathcal{GP})$ is high, the analytical model can continued to be optimized around the local maximum y^*, or *model match between experiment and theory*.

Fabrication Verification By tracking the likelihood score over time, manufacturing can be validated as a non-stationary process. Consider the sketch in Fig. 2b, if the likelihood of multiple samples drops notably from the previous iterations, these designs can be flagged. Grouping the likelihood of predictions per-batch also can give numerical scores to process variance.



(a) Optimization over iterations for jumping energy.



(b) Verification sketch

Figure 2: Tools for optimizing design, preliminary illustrations.

Parameter	Value	Max	Min
L_{ol} (µm)		15.3	76.5
w_{s} (µm)		1	4.5
w_{f} (µm)	4		
T (µm)	40		
x_{o} (µm)	5.8		
x_{f} (µm)	1		
L_{s} (µm)	240		
N	70		

Table 2: Sample state-space for GCA optimization. No min and max means the variable cannot be optimized.

Parameter	Value	Max	Min
Valve side (µm)		300	1500
pacing between fingers (µm)		50	200
Finger length (µm)		100	700
finger width (µm)		10	50
number of fingers		1	30
membrane thickness (nm)		200	1000
metal thickness (nm)		10	200
metal density (gm/cc)		1	30

Table 3: Sample state-space for Thermo-mechanical valve optimization. No min and max means the variable cannot be optimized.

6 Design Examples

In this section we detail two design examples, one with more variables, that we are considering optimizing. The key scripts in this codebase are

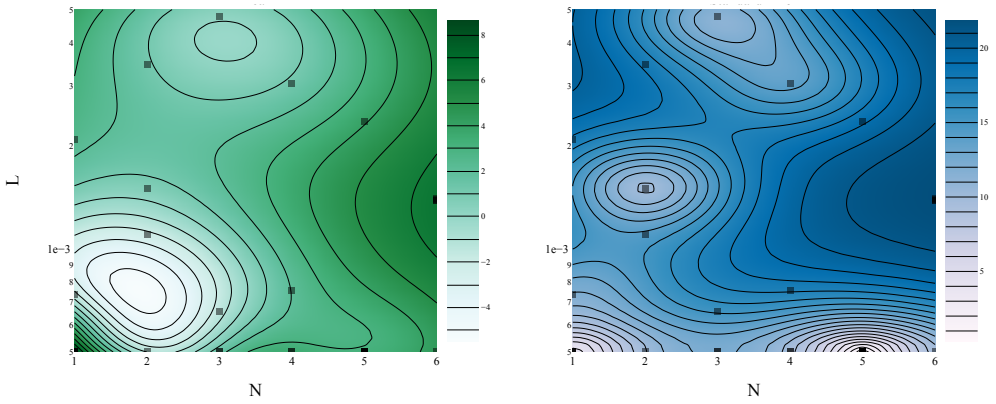
- *sim.py*: in this file a simulated problem is (maybe jointly) optimized by Bayesian Optimization.
- *validate.py*: in this file raw data is loaded into a data-frame and the likelihood of the points versus a simulated and experimental model are listed (*under construction*).

6.1 Gap closing electrostatic motor array (GCA)

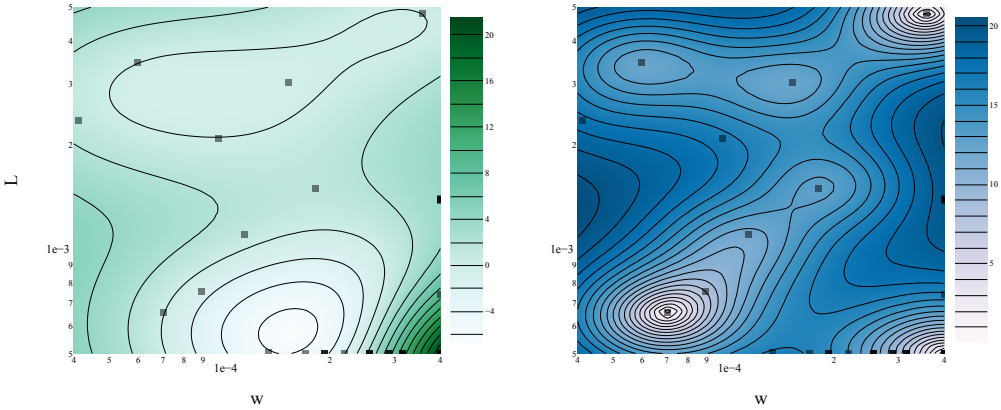
Gap closing actuators (GCA) have been used substantially in our group in order to optimize walking and crawling robots [25]. The CGAs represent the simpler optimization problem with fewer tunable varaibles. The values to be optimized are shown in Tab. 2. The code found at `https://github.com/natolambert/mems-bo`, optimizes the pull-in and pull-out time of the motor arrays with numeric differential equation solvers.

6.2 Thermo-mechanical valve (TMV)

Thermo-mechanical valves (TMVs) represent a more complex design problem [26]. With more variables, and the possibility of visual analysis (see Sec. 7), the thermo-mechanical valves are a more advanced problem. The code currently does not have the representative differential equations and data added, so this is an action item with details in the next section. The primary objective of these values is to require low energy, and this can be modified into a multi-objective problem by co-optimizing to have low fabrication variance.



(a) *Left*: Mean prediction, *Right*: Variance of prediction. Number of fingers versus finger length.



(b) *Left*: Mean prediction, *Right*: Variance of prediction. Finger width vs finger length.

Figure 3: Parameter space for optimized jumping robot design.

6.3 Jumping Robot

An example of the converged design space and the optimization variable over iterations is shown in Fig. 3.

6.4 Adding Problems

There are only a few steps to add your problem into this toolbox! They are listed here:

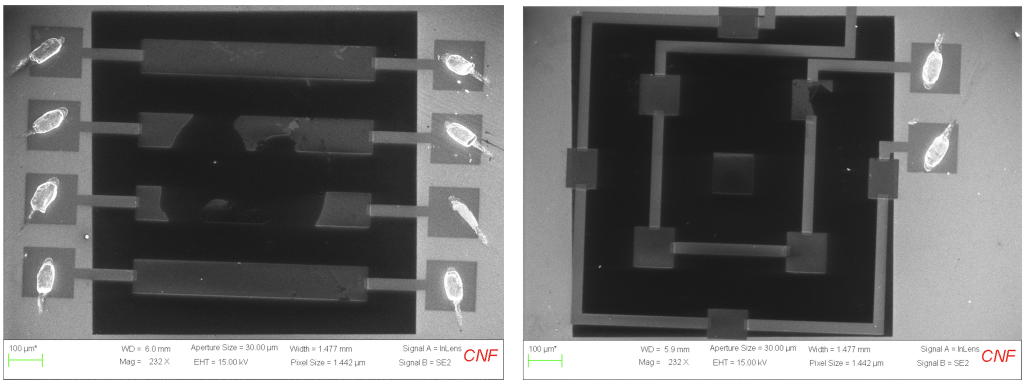
- *Optimization Function*: in `models/`, add a function that takes in a `dictionary` of parameters and returns a `dictionary` of optimization values. Add the import statement to `__init__.py`.
- *Configuration*: in `conf/models/`, add a problem space definition that is read by the code.

7 Advanced Considerations

The combination of design synthesis with machine learning is a challenging problem because it requires expert knowledge in multiple domains. As the areas of study are explored, additional areas for research represent exciting areas to further the space of data-driven design.

7.1 Image Conditioned Optimization

Fabrication variance frequently can be seen by human users in MEMs testing via microscopes. An example in Fig. 4 shows visual blemishes on TMVs, which could be combined with a basic visual feature extractor to add a context variable to the optimization.



(a) Image 1. (b) Image 2.

Figure 4: Two images of TMVs with fabrication blemishes.

7.2 Multi-objective Optimization

Multi-objective optimization is a common task [27]. Frequent solutions take a pre-defined weight of the metrics (combining the values), but more work can be done to show how the variables are jointly optimized. An interesting direction is optimizing the worst case variable, as this could also mitigate design variance.

8 Conclusion

So, it is said. Go forth and try it.

References

[1] Peter I Frazier and Jialei Wang. Bayesian optimization for materials design. In *Information Science for Materials Discovery and Design*, pages 45–75. Springer, 2016.

[2] Isabelle Guyon, Kristin Bennett, Gavin Cawley, Hugo Jair Escalante, Sergio Escalera, Tin Kam Ho, Núria Macià, Bisakha Ray, Mehreen Saeed, Alexander Statnikov, et al. Design of the 2015 chalearn automl challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[3] Kourosh Hakhamaneshi, Nick Werblun, Pieter Abbeel, and Vladimir Stojanovic. Bagnet: Berkeley analog generator with layout optimizer boosted with deep neural networks. *arXiv preprint arXiv:1907.10515*, 2019.

[4] Eric Brochu, Tyson Brochu, and Nando de Freitas. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 103–112, 2010.

[5] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

[6] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

[7] Alden H Wright. Genetic algorithms for real parameter optimization. In *Foundations of genetic algorithms*, volume 1, pages 205–218. Elsevier, 1991.

[8] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24:2546–2554, 2011.

[9] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[10] IM Sobol'. On the systematic search in a hypercube. *SIAM Journal on Numerical Analysis*, 16(5):790–793, 1979.

[11] Ilya Loshchilov and Frank Hutter. Cma-es for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

[12] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

[13] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. Boa: The bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference GECCO-99*, volume 1, pages 525–532. Citeseer, 1999.

[14] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[15] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[16] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[17] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016.

[18] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

[19] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomiin. Goal-driven dynamics learning via bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5168–5173. IEEE, 2017.

[20] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *arXiv preprint arXiv:1908.00709*, 2019.

[21] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[22] Eric Chang, Jaeduk Han, Woorham Bae, Zhongkai Wang, Nathan Narevsky, Borivoje NikoliC, and Elad Alon. Bag2: A process-portable framework for generator-based ams circuit design. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–8. IEEE, 2018.

[23] Eric Chang, Nathan Narevsky, Jaeduk Han, and Elad Alon. An automated serdes frontend generator verified with a 16-nm instance achieving 15 gb/s at 1.96 pj/bit. *IEEE Solid-State Circuits Letters*, 1(12):245–248, 2018.

[24] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8:514–520, 1995.

[25] Daniel S Contreras and Kristofer SJ Pister. Dynamics of electrostatic inchworm motors for silicon microrobots. In *2017 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS)*, pages 1–6. IEEE, 2017.

[26] V. Gund, A. Ruyack, S. Ardanuc, and A. Lal. Graphene stress transducer-based thermo-mechanically fractured micro valve. *Journal of Microelectromechanical Systems*, 27(3):555–569, 2018.

[27] Marco Laumanns and Jiri Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 298–307. Springer, 2002.