Black-Box Optimization with Local Generative Surrogates

Sergey Shirobokov[1*], **Vladislav Belavin**[2*], **Michael Kagan**[3],
Andrey Ustyuzhanin[2], **Atılım Güneş Baydin**[4]
[1]*Department of Physics, Imperial College London, United Kingdom*
[2]*National Research University Higher School of Economics, Moscow, Russia*
[3]*SLAC National Accelerator Laboratory, Menlo Park, CA, United States*
[4]*Department of Engineering Science, University of Oxford, United Kingdom*

Abstract

We propose a novel method for gradient-based optimization of black-box simulators using differentiable local surrogate models. In fields such as physics and engineering, many processes are modeled with non-differentiable simulators with intractable likelihoods. Optimization of these forward models is particularly challenging, especially when the simulator is stochastic. To address such cases, we introduce the use of deep generative models to iteratively approximate the simulator in local neighborhoods of the parameter space. We demonstrate that these local surrogates can be used to approximate the gradient of the simulator, and thus enable gradient-based optimization of simulator parameters. In cases where the dependence of the simulator on the parameter space is constrained to a low dimensional submanifold, we observe that our method attains minima faster than baseline methods, including Bayesian optimization, numerical optimization, and approaches using score function gradient estimators.

1. Introduction

Computer simulation is a powerful method that allows for the modeling of complex real-world systems and the estimation of a system's parameters given conditions and constraints. Simulators drive research in many fields of engineering and science [5] and are also used for the generation of synthetic labeled data for various tasks in machine learning [18, 15, 16]. A common challenge is to find optimal parameters of a system in terms of a given objective function, e.g., to optimize a real-world system's design or efficiency using the simulator as a proxy, or to calibrate a simulator to generate data that match a real-data distribution. A typical simulator optimization problem can be defined as finding $\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \sum_{\boldsymbol{x}} \mathcal{R}(F(\boldsymbol{x}, \boldsymbol{\psi}))$, where $\mathcal{R}$ is an objective we would like to minimize and F is a simulator that we take as a black box with parameters $\boldsymbol{\psi} \in \mathbb{R}^n$ and inputs $\boldsymbol{x} \in \mathbb{R}^d$.

In this work, we focus on cases where the simulator and its inputs are stochastic, so that $\boldsymbol{y} = F(\boldsymbol{x}, \boldsymbol{\psi})$ is a random variable $\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\psi})$, the inputs are $\boldsymbol{x} \sim q(\boldsymbol{x})$, and the objective is expressed as the expectation $\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi})}[\mathcal{R}(\boldsymbol{y})]$. The choice of modeling the simulator inputs $\boldsymbol{x}$ as random reflects the situation common in scientific simulation settings, and our methods are equally applicable for the case without stochastic inputs such that $\boldsymbol{y} \sim p(\boldsymbol{y}; \boldsymbol{\psi})$.

*. Equal contribution. Correspondence to: [1]s.shirobokov17@imperial.ac.uk, [2]vbelavin@hse.ru.

In many settings the cost of running the simulator is high, and thus we aim to minimize the number of simulator calls needed for optimization. Such stochastic simulator optimization occurs in an array of scientific and engineering domains, especially in cases of simulation-based optimization relying on Monte Carlo techniques.

Several methods exist for such optimization, depending on the availability of gradients for the objective function [12]. In order to utilize the strengths of gradient-based optimization while avoiding the high variance often observed with score function gradient estimators, our approach employs deep generative models as differentiable surrogate models to approximate non-differentiable simulators, as described in Figure 1. In high-dimensional parameter spaces, training such surrogates over the complete parameter space becomes computationally expensive. Our technique, which we name *local generative surrogate optimization* (L-GSO), addresses this by using successive local neighborhoods of the parameter space to train surrogates at each step of parameter optimization.

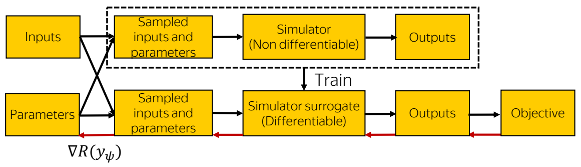

Inputs
Parameters
Sampled inputs and parameters
Simulator (Non differentiable)
Outputs
Train
Sampled inputs and parameters
Simulator surrogate (Differentiable)
Outputs
Objective
$\nabla R(y_\psi)$

Figure 1: Simulation and surrogate training. *Black:* forward propagation. *Red:* error back-propagation.

L-GSO relies primarily on two assumptions: (a) that the objective function is continuous and differentiable, and (b) that the parameters $\boldsymbol{\psi}$ are continuous variables. The first assumption may be relaxed for cases when the objective gradient is not accessible by incorporating the objective into the surrogate.

2. Method

Problem Statement We target an optimization formulation applicable in domains where a simulator characterized by parameters $\boldsymbol{\psi}$ takes stochastic inputs $\boldsymbol{x} \sim q(\boldsymbol{x})$ and produces outputs (observations) $\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi})$. For example in the case of designing the shape of an experimental device, $\boldsymbol{x}$ may represent random inputs to an experimental apparatus, $\boldsymbol{\psi}$ define the shape of the apparatus, and $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi})$ encodes the impact of the apparatus on the input to produce observations $\boldsymbol{y}$. A task-specific objective function $\mathcal{R}(\boldsymbol{y})$ encodes the quality of observations and may be optimized over the parameters $\boldsymbol{\psi}$ of the observed distribution. In cases when a simulator F can only draw samples from the distributions $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi})$ the optimization problem can be approximated as

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathbb{E}[\mathcal{R}(\boldsymbol{y})] = \arg\min_{\boldsymbol{\psi}} \int \mathcal{R}(\boldsymbol{y}) p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi}) q(\boldsymbol{x}) d\boldsymbol{x} d\boldsymbol{y} \approx \arg\min_{\boldsymbol{\psi}} \frac{1}{N}\sum_{i=1}^{N} \mathcal{R}(F(\boldsymbol{x}_i;\boldsymbol{\psi})) \tag{1}$$

where $\boldsymbol{y}_i = F(\boldsymbol{x}_i;\boldsymbol{\psi}) \sim p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\psi}),\quad x_i \sim q(\boldsymbol{x})$ and a Monte Carlo approximation to the expected value of the objective function is computed using samples drawn from the simulator. Here F represents a stochastic process, which may itself depend on latent random variables.

2.1 Deep generative models as differentiable surrogates

Given a non-differentiable simulator F, direct gradient-based optimization of Eq. 1 is not possible. We propose to approximate F with a learned differentiable model, denoted a surrogate, $\bar{\boldsymbol{y}} = S_\theta(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\psi})$ that approximates $F(\boldsymbol{x}; \boldsymbol{\psi})$, where $\boldsymbol{z} \sim p(\boldsymbol{z})$ are latent variables accounting for the stochastic variation of the distribution $p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\psi})$, θ are surrogate model parameters, and $\bar{\boldsymbol{y}}$ are surrogate outputs. When the samples $\bar{\boldsymbol{y}}$ are differentiable with respect to $\boldsymbol{\psi}$, direct optimization of Eq. 1 can be done with the surrogate gradient estimator:

$$\nabla_{\boldsymbol{\psi}} \mathbb{E}[\mathcal{R}(\boldsymbol{y})] \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\psi}} \mathcal{R}(S_\theta(\boldsymbol{z}_i, \boldsymbol{x}_i; \boldsymbol{\psi})) \,. \tag{2}$$

To obtain a differentiable surrogate capable of modeling a stochastic process, S_θ is defined as a deep generative model whose parameters θ are learned. We present results using conditional variants of two recently proposed models, Cramer GAN [4] and the FFJORD flow model [6], to show the independence of L-GSO from the choice of generative model.

2.2 Local generative surrogates

The L-GSO optimization algorithm is summarized in Algorithm 1.

Algorithm 1 Local Generative Surrogate Optimization (L-GSO) procedure

```
Require: number N of $\boldsymbol{\psi}$, number M of $\boldsymbol{x}$ for surrogate training, number K of $\boldsymbol{x}$ for $\boldsymbol{\psi}$ optimization step, trust region $U_\epsilon$, size of the neighborhood $\epsilon$, Euclidean distance $d$
 1: Choose initial parameter $\boldsymbol{\psi}$
 2: while $\boldsymbol{\psi}$ has not converged do
 3:     Sample $\boldsymbol{\psi}_i$ in the region $U_\epsilon^{\boldsymbol{\psi}}$, $i = 1, \dots, N$
 4:     For each $\boldsymbol{\psi}_i$, sample inputs $\{\boldsymbol{x}_j^i\}_{j=1}^M \sim q(\boldsymbol{x})$
 5:     Sample $M \times N$ training examples from simulator $\boldsymbol{y}_{ij} = F(\boldsymbol{x}_j^i; \boldsymbol{\psi}_i)$
 6:     Store $\boldsymbol{y}_{ij}, \boldsymbol{x}_j^i, \boldsymbol{\psi}_i$ in history $H$
        $i = 1, \dots, N; j = 1, \dots, M$
 7:     Extract all $\boldsymbol{y}_l, \boldsymbol{x}_l, \boldsymbol{\psi}_l$ from history $H$,
        iff $d(\boldsymbol{\psi}, \boldsymbol{\psi}_l) < \epsilon$
 8:     Train generative surrogate model
        $S_\theta(\boldsymbol{z}_l, \boldsymbol{x}_l; \boldsymbol{\psi}_l)$, where $\boldsymbol{z}_l \sim \mathcal{N}(0, 1)$
 9:     Fix weights of the surrogate model $\theta$
10:     Sample $\bar{\boldsymbol{y}}_k = S_\theta(\boldsymbol{z}_k, \boldsymbol{x}_k; \boldsymbol{\psi}), \boldsymbol{z}_k \sim \mathcal{N}(0, 1)$,
        $\boldsymbol{x}_k \sim q(\boldsymbol{x})$, $k = 1, \dots, K$
11:     $\nabla_{\boldsymbol{\psi}} \mathbb{E}[\mathcal{R}(\bar{\boldsymbol{y}})] \leftarrow \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \mathcal{R}}{\partial \bar{\boldsymbol{y}}_k} \frac{\partial S_\theta(\boldsymbol{z}_k, \boldsymbol{x}_k; \boldsymbol{\psi})}{\partial \boldsymbol{\psi}}$
12:     $\boldsymbol{\psi} \leftarrow \mathrm{SGD}(\psi, \nabla_{\boldsymbol{\psi}} \mathbb{E}[\mathcal{R}(\bar{\boldsymbol{y}})])$
13: end while
```

For high-dimensional $\boldsymbol{\psi}$, a large number of parameter values $\boldsymbol{\psi}$ must be sampled to accurately train a single surrogate model. Otherwise the surrogate would not provide sufficiently well estimated gradients over the full parameter space that may be explored by the optimization. Thus optimization using a single upfront training of the surrogate model over all $\boldsymbol{\psi}$ becomes unfeasible. As such, we utilize a trust-region like approach [20] to train a surrogate model locally in the proximity of the current parameter value $\boldsymbol{\psi}$. Using this local model, a gradient at the current point $\boldsymbol{\psi}$ can be obtained and a step of SGD performed. After each SGD update a new local surrogate is trained. As a result, we do not expect domain shift to impact L-GSO.

In local optimization there are several hyperparameters that require tuning either prior to or dynamically during optimization. One must choose the sampling algorithm for $\boldsymbol{\psi}$ values in the region U_ϵ^{ψ} in step 3 of Algorithm 1. In high-dimensional space, uniform sampling is inefficient, thus we have adopted the Latin Hypercubes algorithm [10]. One must also choose a proximity hyperparameter ϵ, that controls the size of the region of $\boldsymbol{\psi}_i$ in which a set of $\boldsymbol{\psi}$ values is chosen to train a local surrogate. The number of $\boldsymbol{\psi}$ values sampled in the neighborhood is another key hyperparameter. We expect the optimal value to be highly correlated with the dimensionality and complexity of the problem.

Previously sampled data points can also be stored in history and later reused in our local optimization procedure (Algorithm 1). This provides additional training points for the surrogate as the optimization progresses. This results in a better surrogate model and, consequently, better gradient estimation.

The benefit of our approach, in comparison with numerical gradient estimation, is that a deep generative surrogate can learn more complex approximations of the objective function than a linear approximation, which can be beneficial to obtain gradients for surfaces with high curvature. In addition, our method allows a reduction of the number of function calls by reusing previously sampled points. Utilizing generative neural networks as surrogates also provides potential benefits such as Hessian estimation, that may be used for second-order optimization algorithms and/or uncertainty estimation, and possible automatic determination of a low-dimensional parameter manifold.

3. Experiments

We evaluate L-GSO on five toy experiments in terms of the attained optima and the speed of convergence, and present results in a physics experiment optimization. As simulation is assumed to be the most time consuming operation during optimization, the speed of convergence is measured by the number of simulator calls. The toy experiments, defined below, were chosen to explore low- and high-dimensional optimization problems, and those with parameters on submanifolds. Non-stochastic versions of the experiments are established benchmark functions in the optimization literature [11].

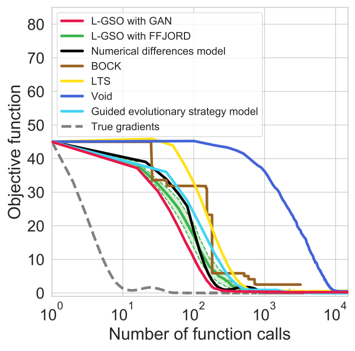
Rosenbrock Problem In the N-dimensional Rosenbrock problem we aim to find $\boldsymbol{\psi}$ that optimizes:

$$\begin{gathered}\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathbb{E}[\mathcal{R}(y)] = \arg\min_{\boldsymbol{\psi}} \mathbb{E}[y], \text{ s.t.} \\ y \sim \mathcal{N}\left(y; \sum_{i=1}^{n-1}\left[(\psi_i - \psi_{i+1})^2 + (1-\psi_i)^2\right] + x, 1\right), \quad x \sim \mathcal{N}(x; \mu, 1), \quad \mu \sim \mathrm{U}[-10, 10]\end{gathered} \tag{3}$$

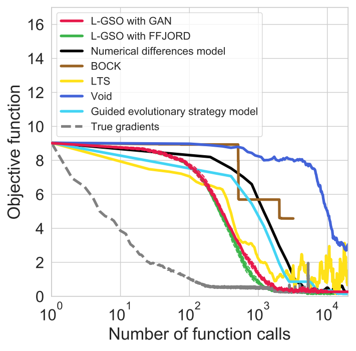
Submanifold Rosenbrock Problem To address problems where the parameters lie on a low-dimension submanifold, we define the submanifold Rosenbrock problem, with a mixing matrix A to project the parameters onto a submanifold. The orthogonal matrix A is generated via a QR-decomposition of a random matrix sampled from the normal distribution. In our experiments $A \in \mathbb{R}^{10\times 100}$ has full row rank. Prior knowledge of A or the submanifold dimension is not used in the surrogate training. The optimization problem is thus defined as above with $\boldsymbol{\psi}' = A \cdot \boldsymbol{\psi}$.

Neural Network Weights Optimization Problem In this problem, we optimize neural network weights for regressing the Boston house prices dataset [9]. In this experiment we explore the optimization capability of L-GSO over the number of parameter space points needed per surrogate training, and, indirectly, the intrinsic dimensionality of the problem.

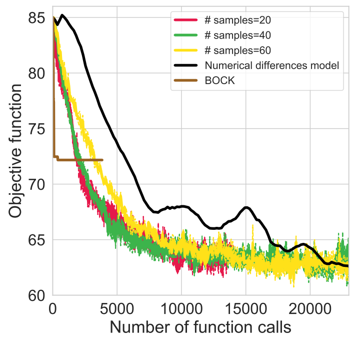
Baselines We compare L-GSO to: Bayesian optimization using Gaussian processes with cylindrical kernels [14], which we denote "*BOCK*", numerical differentiation with gradient descent (referred to as numerical optimization), and guided evolutionary strategies [13]. We also compare with score function-based optimization approaches. We use methods developed in [18] which we denote "*LTS*" and LAX estimator from [7] which we denote "*Void*".



(a)



(b)



(c)

Figure 2: The objective function value on the toy problems for baselines and our method. (a) Rosenbrock problem [17] in 10 dimensions, initial point is $\vec{2} \in \mathbb{R}^{10}$. (b) Submanifold Rosenbrock Problem in 100 dimensions, initial point is $\vec{2} \in \mathbb{R}^{100}$. True gradients are shown in gray dashed curves when available. Shaded region corresponds to 1σ confidence intervals. (c) Neural Network Weights Optimization problem $\boldsymbol{\psi} \in \mathbb{R}^{91}$

Results Our primary metrics for comparison are the final objective function value and the number of simulator function calls needed to find a minimum. The latter metric assumes that the simulator calls are driving the computation time of the optimization.

The objective value as a function of the number of simulator calls in three experiments is seen in Figure 2. L-GSO outperforms score function based algorithms in speed of convergence by approximately an order of magnitude. L-GSO also attains the same optima as other methods and the speed of convergence is comparable to numerical optimization. In Figure 2a BO struggles to find the optimum due to the high curvature of the objective function, whereas the convergence speed of L-GSO is on par with numerical optimization. In general, L-GSO has several advantages over BO: (a) it is able to perform optimization without specification of the search space [8, 19], (b) the algorithm is parallelizable, though we note that BO parallelization is an active area of research [21].

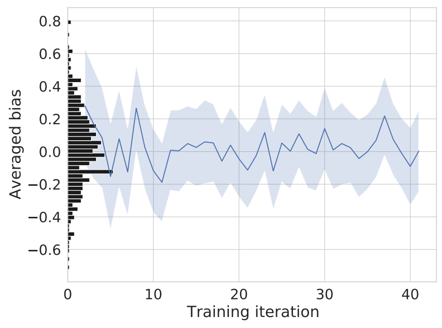


Figure 3: The bias (solid line) and one standard deviation (shaded region) of the GAN based L-GSO gradient averaged over all $\boldsymbol{\psi}$ dimensions in the 10D Rosenbrock problem versus training step. Gray histogram shows the empirical bias distribution over all training iterations.

The bias and variance of the GAN based L-GSO gradient estimate averaged over all parameters for the 10D Rosenbrock problem for each training step can be seen in Figure 3. The bias is close to, and within one standard deviation of, zero across the entire training.

The benefits of L-GSO can further be seen in problems with parameter submanifolds, i.e., the Submanifold Rosenbrock and Neural Network Weights Optimization problems where the relevant ψ parameters live on a latent low-dimensional manifold. No prior knowledge of the submanifold is used in the training and all dimensions of $\boldsymbol{\psi}$ are treated equally for all algorithms. The objective value versus the number of simulator calls can be seen in Figures 2b and 2c where we observe that L-GSO outperforms all baseline methods.

Figure 2c compares L-GSO with differing numbers of parameter space points used per surrogate training. In the submanifold problems, L-GSO converges fastest with far fewer parameter points than the full dimension of the parameter space. This indicates that the surrogate is learning about the latent manifold of the data, rather than needing to fully sample the volume around a current parameter point.

3.1 Physics experiment example

We apply L-GSO to optimize the parameters of an apparatus in a physics experiment setting that uses the physics simulation software GEANT4 [1] and FairRoot [2]. In this example, muon particles pass through a multi-stage steel magnet and their coordinates are recorded when muons leave the magnet volume if they pass through the sensitive area of a detection apparatus. As muons are unwanted in the experiment, the objective is to minimize number of recorded muons by varying the geometry ($\boldsymbol{\psi} \in \mathbb{R}^{42}$) of the magnet.

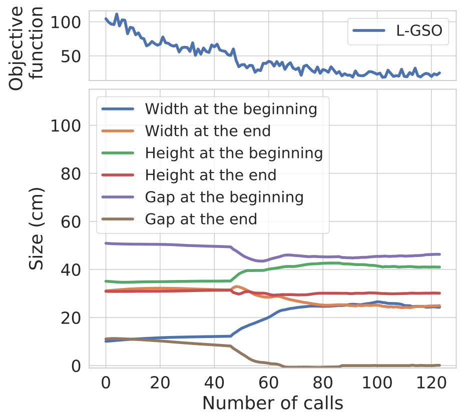


Figure 4: Magnet objective function (top) and six $\boldsymbol{\psi}$ parameters (bottom) during optimization.

Results of the optimization using L-GSO with a Cramer GAN [4] surrogate are presented in Figure 4. A previous optimization of this magnet system was performed using BO with Gaussian processes with RBF kernels [3]. The L-GSO optima has an objective function value approximately 25% smaller than the BO solution, while using approximately the same budget of $O(5,000)$ simulator calls. The L-GSO solution is shorter and has lower mass than the BO solution, which can both improve efficacy of the experiment and reduce cost.

4. Conclusions

We present a novel approach for the optimization of stochastic non-differentiable simulators. Our proposed algorithm is based on deep generative surrogates successively trained in local neighborhoods of parameter space during parameter optimization. We compare against baselines including methods based on score function gradient estimators [18, 7], numerical differentiation, and Bayesian optimization with Gaussian processes [14]. Our method, L-GSO, is generally comparable to baselines in terms of speed of convergence, but is observed to excel in performance where simulator parameters lie on a latent low-dimensional submanifold of the whole parameter space. L-GSO is parallelizable, and has a range of advantages including low variance of gradient estimates, scalability to high dimensions, and applicability for optimization on high curvature objective function surfaces. We performed experiments on a range of toy problems and a real high-energy physics simulator. Our results improved on previous optimizations obtained with Bayesian optimization, thus showing the successful optimization of a complex stochastic system with a user-defined objective function.

References

[1] S. Agostinelli et al. GEANT4: A Simulation toolkit. *Nucl. Instrum. Meth.*, A506: 250–303, 2003. doi: 10.1016/S0168-9002(03)01368-8.

[2] M. Al-Turany, D. Bertini, R. Karabowicz, D. Kresan, P. Malzacher, T. Stockmanns, and F. Uhlig. The fairroot framework. *Journal of Physics: Conference Series*, 396(2): 022001, 2012. doi: 10.1088/1742-6596/396/2/022001.

[3] A. Baranov, E. Burnaev, D. Derkach, A. Filatov, N. Klyuchnikov, O. Lantwin, F. Ratnikov, A. Ustyuzhanin, and A. Zaitsev. Optimising the active muon shield for the SHiP experiment at CERN. *Journal of Physics: Conference Series*, 934:012050, dec 2017. doi: 10.1088/1742-6596/934/1/012050. URL `https://doi.org/10.1088%2F1742-6596%2F934%2F1%2F012050`.

[4] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

[5] K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *arXiv preprint arXiv:1911.01429*, 2019.

[6] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. *CoRR*, abs/1810.01367, 2018. URL `http://arxiv.org/abs/1810.01367`.

[7] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SyzKd1bCW`.

[8] H. Ha, S. Rana, S. Gupta, T. Nguyen, H. Tran-The, and S. Venkatesh. Bayesian optimization with unknown search space. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11795–11804. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9350-bayesian-optimization-with-unknown-search-space.pdf`.

[9] D. Harrison and D. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102, 03 1978. doi: 10.1016/0095-0696(78)90006-2.

[10] R. Iman, J. Davenport, and D. Zeigler. *Latin hypercube sampling (program user's guide). [LHC, in FORTRAN]*. 01 1980. ISBN SAND-79-1473 United StatesThu Feb 07 00:00:56 EST 2008Dep. NTIS, PC A05/MF A01.SNL; ERA-05-017293; EDB-80-043435English.

[11] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. *IJMNO*, 4:150–194, 2013.

[12] G. Lei, J. Zhu, Y. Guo, C. Liu, and B. Ma. A review of design optimization methods for electrical machines. *Energies*, 10:1962, 11 2017. doi: 10.3390/en10121962.

[13] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein. Guided evolutionary strategies: augmenting random search with surrogate gradients. In *ICML*, 2018.

[14] C. Oh, E. Gavves, and M. Welling. Bock : Bayesian optimization with cylindrical kernels. In *ICML*, 2018.

[15] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016.

[16] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[17] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, 01 1960. ISSN 0010-4620. doi: 10.1093/comjnl/3.3.175. URL `https://doi.org/10.1093/comjnl/3.3.175`.

[18] N. Ruiz, S. Schulter, and M. Chandraker. Learning to simulate. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJgkx2Aqt7`.

[19] B. Shahriari, A. Bouchard-Cote, and N. Freitas. Unbounded bayesian optimization via regularization. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1168–1176, Cadiz, Spain, 09–11 May 2016. PMLR. URL `http://proceedings.mlr.press/v51/shahriari16.html`.

[20] D. C. Sorensen. Newton's method with a model trust region modification. *SIAM J. Numer. Anal.*, 19(2):409–426, 1982.

[21] J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Parallel bayesian global optimization of expensive functions. 2016.