Neuromorphic Bayesian Optimization in Lava

Shay Snyder
George Mason University
Fairfax, Virginia, USA
ssnyde9@gmu.edu

Sumedh R. Risbud
Intel Labs
Santa Clara, California, USA
sumedh.risbud@intel.com

Maryam Parsa
George Mason University
Fairfax, Virginia, USA
mparsa@gmu.edu

ABSTRACT

The ever-increasing demands of computationally expensive and high-dimensional problems require novel optimization methods to find near-optimal solutions in a reasonable amount of time. Bayesian Optimization (BO) stands as one of the best methodologies for learning the underlying relationships within multi-variate problems. This allows users to optimize time consuming and computationally expensive black-box functions in feasible time frames. Existing BO implementations use traditional von-Neumann architectures, in which data and memory are separate. In this work, we introduce Lava Bayesian Optimization (LavaBO) as a contribution to the open-source Lava Software Framework. LavaBO is the first step towards developing a BO system compatible with heterogeneous, fine-grained parallel, in-memory neuromorphic computing architectures (e.g., Intel's Loihi platform). We evaluate the algorithmic performance of the LavaBO system on multiple problems such as training state-of-the-art spiking neural networks through back-propagation and evolutionary learning. Compared to traditional algorithms (such as grid and random search), we highlight the ability of LavaBO to explore the parameter search space with fewer expensive function evaluations, while discovering the optimal solutions.

CCS CONCEPTS

• **Mathematics of computing** → **Bayesian computation**; • **Computing methodologies** → **Search methodologies**; • **Computer systems organization** → *Distributed architectures.*

KEYWORDS

Bayesian optimization, neuromorphic computing, asynchronous computing

ACM Reference Format:
Shay Snyder, Sumedh R. Risbud, and Maryam Parsa. 2023. Neuromorphic Bayesian Optimization in Lava. In *International Conference on Neuromorphic Systems (ICONS '23), August 1–3, 2023, Santa Fe, NM, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3589737.3605998

1 INTRODUCTION

At the core of a vast array of problems lies a large function, evaluation of which is the most computationally expensive step: neural network design [19], transportation systems [26], graph neural networks [5], and evolutionary algorithms [17] are some examples. The common attribute in these problems is the computational complexity of the evaluation. Computer scientists and mathematicians dedicate tremendous amounts of time and energy developing and optimizing a plethora of algorithms for finding optimal or near-optimal solutions to such problems. The algorithms are either rooted in the study of the theory of computation [27] or the development of statistical models of relationships between multiple variables [28]. The use of Bayesian Optimization (BO), based on Bayes' theorem, belongs to the latter class. Published in 1763, Bayes' theorem [4] revolutionized the field of conditional probability and probabilistic inference. It allows Bayesian systems to use their prior knowledge to construct probabilistic models of the world. There are many applications ranging from communication systems [14], medical diagnosis [11], and quantitative finance [21] to hyperparameter optimization [16–18].

In this work, we introduce Lava Bayesian Optimization (LavaBO), which adds support for BO in the open-source Lava Software Framework [8] and makes it available to the neuromorphic community.[1] LavaBO is the first step towards developing a BO system compatible with fine-grained parallel neuromorphic computing architectures (e.g., Intel's Loihi platform [6]). Specifically, using the CPU-based implementation of LavaBO, we evaluate its algorithmic performance on multiple problems: the classical non-convex case of the Ackley function [2], a classification problem using an evolutionary algorithm (EONS [24]) on the TENNLab framework [20], and a deep spiking neural network performing classification problem on the NMNIST dataset [15] using the SLAYER algorithm [10] from the Lava deep learning library [13].

We compare our results with grid and random search algorithms to highlight LavaBO's ability to uncover optimal solutions with fewer numbers of expensive evaluations of the corresponding black-box functions. We conclude this article with a detailed discussion of the broader impact of this technology and our plan to implement it on Intel's Loihi 2 neuromorphic chip [12].

2 ARCHITECTURE OF LAVABO WITHIN THE LAVA FRAMEWORK

Lava Software Framework is an open-source software framework, aimed at lowering the barrier for programming neuromorphic hardware and prototyping neuro-inspired algorithms [8]. It achieves this goal by providing the necessary tools and abstractions for creating applications that can run on different types of neuromorphic platforms, such as Intel's Loihi chips. Lava also supports conventional CPUs and GPUs for prototyping and testing purposes. Lava aims to be a modular, composable, and extensible framework that allows researchers to integrate their ideas into a growing algorithms library

[1] Code available as a part of `https://github.com/lava-nc/lava-optimization`

This work is licensed under a Creative Commons Attribution International 4.0 License.
ICONS '23, August 1–3, 2023, Santa Fe, NM, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0175-7/23/08.
https://doi.org/10.1145/3589737.3605998

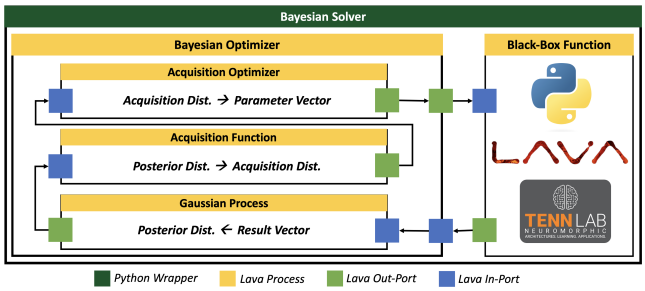

Figure 1: A flowchart presenting the programmatic architecture of the Lava Bayesian Optimization system.

and build complex neuromorphic applications. One such library built on top of the Lava primitives is the Lava Optimization library. LavaBO is implemented with the same interface structure as the rest of the solvers within the Lava-Optimization library. This leads to a situation where the main interface between users and LavaBO is a wrapper class that initializes, connects, and executes all underlying Lava Processes[2]. Figure 1 shows a detailed look at the structure of LavaBO. The code and a tutorial on how to leverage this tool are accessible at `https://github.com/lava-nc/lava-optimization`.

Bayesian Solver

The BayesianSolver abstracts lower-level functionalities of the optimization process away and serves as the interface contract between users and developers. The system is highly configurable with multiple acquisition functions, acquisition optimizers, initial point generators, the number of initial points, and the random state. Once the desired parameters are determined and the BayesianSolver is initialized, the final remaining steps are creating a Lava Process and ProcessModel wrapper around the black-box function. The only requirements for these wrappers are that they must receive an array of parameters and return an array of parameters along with the resulting score.

Gaussian Regressor

The Gaussian-Regressor learns through prior knowledge and constructs a statistical model of unknown points and their resultant values. This system is implemented as a floating-point Gaussian regressor based on equation 1

$$f(x) \sim \mathbb{GP}(m(x), k(x, x')) \tag{1}$$

where $m(x)$ is a mean function over a real process $f(x)$ and $k(x, x')$ is a covariance function over the same $f(x)$ [22].

Acquisition Function

We will explore and exploit the search space through the acquisition function. The AcquisitionFunction defines and calculates a function on the posterior distribution from the GaussianRegressor that represents uncertainly across the space. LavaBO supports three different acquisition functions: lower confidence bound [23], negative expected improvement [3], and negative probability of improvement [3].

[2]See `http://lava-nc.org` for details about Lava concepts like a *Process*.

Acquisition Optimizer

The AcquisitionOptimizer determines how points are selected based on the acquisition distribution from the AcquisitionFunction. For example, one could calculate the acquisition function across all points in the space or intelligently or randomly select a subset to calculate. We currently support two methods: random sampling [1] and inverse Hessian matrix estimation [29].

Closing the loop

The process begins at the initial point sampler (IPS), where points are broadly sampled to construct the initial surrogate model of the space within the GaussianRegressor. The Acquisition-Distribution then calculates an acquisition distribution from the GaussianRegressor's posterior distribution that models uncertainty throughout the search space. The AcquisitionOptimizer uses the acquisition distribution to determine which point to select for evaluation. The selected point is transferred from the Bayesian-Optimizer to the user's black-box function where the given hyperparameter configuration will be evaluated. After evaluation, the black-box function transfers the result back to the BayesianOptimizer.

The BayesianOptimizer has new information about the underlying search space from the black-box process. Therefore, the next step is to transfer this information to the GaussianRegressor where it will incorporate this result into its statistical model of the space. The newfound posterior distribution from the GaussianRegressor is then sent to the AcquisitionFunction. Here, the posterior distribution is used to update the acquisition distribution. Lastly, the acquisition distribution is sent to the AcquisitionOptimizer where the next point is determined.

3 RESULTS

To highlight the capability of LavaBO, we chose three experiments for optimizing:

(1) a standard two dimensional function, the Ackley function [2].
(2) hyperparameters of an evolutionary algorithm for training a spiking neural network to classify the IRIS dataset [7].
(3) hyperparameters of a backpropagation-based learning algorithm, SLAYER from the Lava deep learning library [13].

We use random search or grid search algorithms as a baseline performance measurement for all experiments. Table 1 summarizes the results from all experiments. In the subsections following the table, we discuss the details of each experiment.

The Ackley function

Finding the global minimum of the Ackley function is a classical benchmark problem for non-convex global optimization. The equation for the Ackley function is given as 2.

$$f(x) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d} cos\,(cx_i)\right) + (a + e) \tag{2}$$

Experiment	Baseline algo.	Function eval.s		Gain
		Baseline	LavaBO	
Ackley function	Random search	500	50	10X
Evolutionary algo.	Grid search	432	20	21X
Deep SNN training†	Random search	28	8	3.5X

Table 1: Summary of results from all three experiments detailed in this work. †In the case of training deep SNNs, the number of network evaluations to reach the same accuracy was counted. This iso-accuracy level is ~30% after 10 training epochs, as evident from Figure 5.

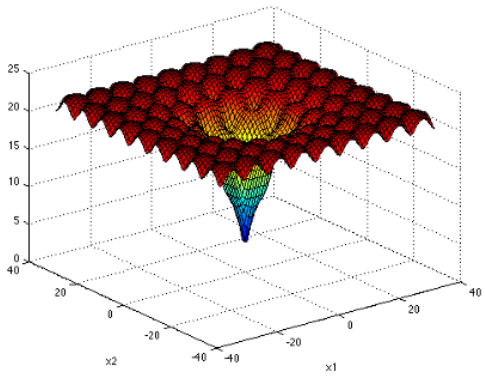

Figure 2: The Ackley function where $x_i \in [-32.768, 32.768]$ along with $a = 20$, $b = 0.2$, and $c = 2\pi$.

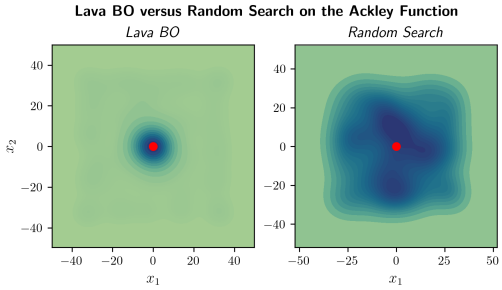

Figure 3: The probability distributions of observed points from Lava Bayesian Optimization and random search on the Ackley function. The red dots at $x_i = 0$ represent the optimum point within the search space.

where $a, b, c \in \mathbb{R}$ and $x_i \in [-32.768, 32.768]$. Figure 2 presents a visualization of the Ackley function within the x_i bounds where $a = 20$, $b = 0.2$, and $c = 2\pi$.

We conducted 10 LavaBO runs over the bounds of the Ackley function with each run consisting of randomly sampling 10 initial points from the search space before the surrogate model and acquisition function are used to intelligently observe 40 more points. A baseline random search was also conducted 10 times over the search space with each run consisting of randomly sampling and evaluating 50 points from the space. The probability distributions of the observed points for both algorithms are shown in Figure 3.

Parameter	*Options*
Crossover Rate	0.1, 0.3, 0.5, 0.7
Mutation Rate	0.1, 0.3, 0.5, 0.7
Num Mutations	1, 2, 3
Num Starting Edges	3, 5, 7
Num Starting Node	3, 5, 7

Table 2: The hyperparameter search space for the grid search and LavaBO on the evolutionary learning for IRIS classification problem. Search space size is 432.

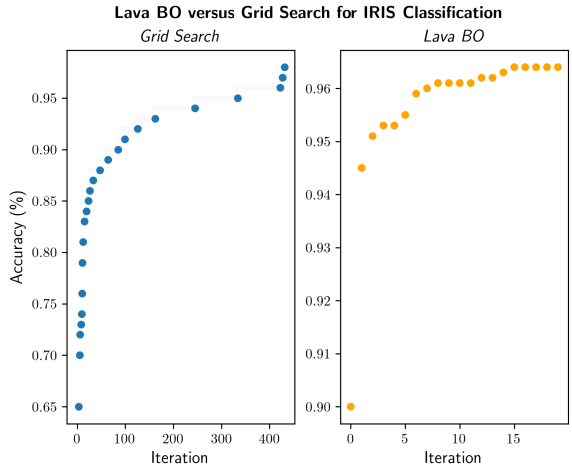

Figure 4: A comparison of LavaBO and grid search performance in optimization accuracy of IRIS classification using evolved spiking neural networks.

There are stark performance differences between random search and LavaBO. As expected, the probability density function from the random search closely mirrors a uniform distribution. LavaBO's observations are tightly clustered around the minimum of the Ackley function, with a small percentage of the observations used in the fringes of the search space.

Evolutionary Learning for IRIS Classification

In this experiment we compare performance of LavaBO and an exhaustive grid search on optimizing hyperparameters of an evolutionary-based learning algorithm (EONS) [25] to solve a classification task on the IRIS dataset [7]. Details of the hyperparameters and search space can be found in Table 2. We completed this task within the TENNLab framework [7]. The grid search was performed over all 432 parameter combinations; however, the LavaBO was only ran for 20 iterations. We repeated the LavaBO process 10 times and averaged the final results to gain a broader understanding of the stability of the system. This is shown in Figure 4. LavaBO was able to learn the hyperparameter space and achieve 96.3% accuracy after 16 Bayesian iterations.

Parameter	*Lower Bound*	*Upper Bound*	*Delta*
Threshold	0.0	5.0	0.125
Current Decay	0.0	0.7	0.01
Voltage Decay	0.0	0.7	0.01
Tau Grad	0.0	0.7	0.01
Learning Rate	10^{-20}	0.1	0.1

Table 3: The hyperparameter search space for the optimization problem with the Lava Deep Learning library, SLAYER, and NMNIST. All parameters have a series of discrete options between the min and max values. The individuals values are separated by the delta. This search space contains 137M parameter combinations.

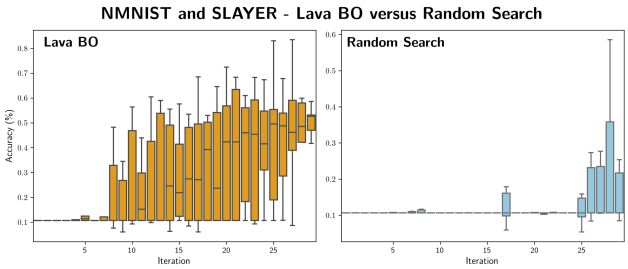

Figure 5: Side by side box plots of observation evaluations at every optimization iteration with Lava BO and random search. Box plots along the x-axis were sorted by their means. We see that the random search algorithm rarely ever evaluated hyperparameter configurations with greater than 25% **accuracy while Lava BO is consistently exploring points over** 40% **accuracy despite only training for** 10 **epochs.**

SLAYER on Lava-DL for NMNIST Classification

In the third experiment, we demonstrate LavaBO's ability to find optimum hyperparameters of a deep spiking neural network with limited training epochs. We used the Lava deep learning library [15] on the NMNIST dataset which aims to classify event-based handwritten digits. While the native training configuration requires 200 epochs, we limited the number of training epochs during the optimization process to 10 as many applications require models to converge very quickly. Therefore the goal of this experiment is to use LavaBO to find a hyperparameter configuration that achieves the highest validation accuracy during this limited training period.

The search space consists of parameters for voltage threshold of the spiking neurons ('threshold'), decay constants for current and voltage ('current decay' and 'voltage decay', respectively), the time constant of the spike function derivative ('tau grad'), and the rate of change of the model parameters (the 'learning rate'). More information on the specifics of each parameter and their ranges are shown in Table 3.

We ran random search and LavaBO on Lava-DL-SLAYER for NMNIST digit classification for 30 iterations and repeated each hyperparameter combination for three times. We only trained each hyperparameter combination for 10 epochs. Lava BO, on average, converges to a specific subset of the parameter search space that

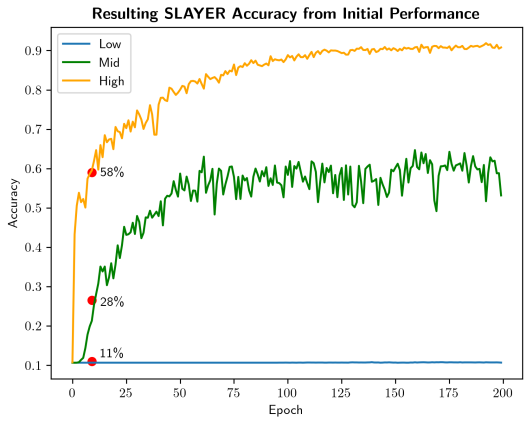

Figure 6: Accuracy plots for classifying NMNIST dataset for different hyperparameter combinations using SLAYER for 200 epochs. The three hyperparameter configurations are chosen based on low, medium, and high performing after training for 10 epochs.

has over twice the accuracy of random search. This suggests that a subset of the space leads to more rapid convergence on the NMNIST digit classification task. Figure 5 highlights this fact where Lava BO is constantly exploring new parameter combinations that have a much higher probability of performing well versus random search where the only high performing combinations are outliers.

Evaluating the implications of our reduced number of training epochs, we chose 3 average configurations representing low, medium, and high performers with medians of 11%, 28%, and 58% after 10 epochs, respectively. These categories and statistics were selected based on the distribution of results from Figure 5. For these parameter combinations we continued the training for 200 epochs. Figure 6 highlights the accuracy values for training for 200 epochs. For each combination, this figure also shows the original accuracy values after 10 epochs. The low performing model never improved its accuracy. The medium configuration represents a midpoint between the low and high categories, where the the final accuracy is between the upper and lower categories. Lastly, the high performance model achieved around 90% accuracy after 200 epochs. We conclude that the highest performing models during the initial 10 epochs training have a higher likelihood of performing well at the end of full training sessions (200 epochs).

4 DISCUSSION

In this work, we introduced the open-access Lava Bayesian Optimization (LavaBO) solver within the Lava Software Framework for neuromorphic computing. We outlined the architecture of the solver and compared its algorithmic performance with traditional algorithms like random search, with the help of three problems: optimization of the Ackley function, hyperparameter tuning for evolutionary algorithms, and hyperparameter tuning for training deep spiking neural networks. The results consistently show that LavaBO finds optimal hyperparameter combinations in significantly less iterations than random search.

As the natural next step, we are going to extend the LavaBO solver to be compatible with constraints imposed by the Loihi 2 neuromorphic chip. We hope that the extension will enable us to accelerate the execution of the solver using Loihi 2, while significantly reducing the power consumption. Such extension requires us to focus on two aspects: (a) Loihi 2 supports only fixed-point integer arithmetic with limited precision. As a result, we are investigating the effects of rounding on the accuracy of our solver, (b) additionally, for neural implementation of various components of LavaBO (e.g., the Gaussian Process regression), we are exploring the area of hyperdimensional (HD) computing using vector symbolic architectures [9]. We hope this enables us to map all components of LavaBO efficiently on a neuromorphic substrate like Loihi 2.

ACKNOWLEDGMENT

The work in this paper is supported by a gift from Intel Corporation.

REFERENCES

[1] Anita S Acharya, Anupam Prakash, Pikee Saxena, and Aruna Nigam. 2013. Sampling: Why and how of it. *Indian Journal of Medical Specialties* 4, 2 (2013), 330–333.

[2] David H Ackley. 1987. The model. In *A Connectionist Machine for Genetic Hillclimbing*. Springer, 29–70.

[3] Apoorv Agnihotri and Nipun Batra. 2020. Exploring Bayesian Optimization. *Distill* (2020). https://doi.org/10.23915/distill.00026 https://distill.pub/2020/bayesian-optimization.

[4] Thomas Bayes. 1763. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London* 53 (1763), 370–418.

[5] Guojing Cong, Seung-Hwan Lim, Shruti Kulkarni, Prasanna Date, Thomas Potok, Shay Snyder, Maryam Parsa, and Catherine Schuman. 2022. Semi-Supervised Graph Structure Learning on Neuromorphic Computers. In *Proceedings of the International Conference on Neuromorphic Systems 2022*. 1–4.

[6] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[7] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[8] Lava Software Framework. A software framework for neuromorphic computing. http://lava-nc.org

[9] P Michael Furlong, Terrence C Stewart, and Chris Eliasmith. [n. d.]. Fractional binding in vector symbolic representations for efficient mutual information exploration.

[10] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Zé Vinícius, cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene rex, Kejia (KJ) Shi, Justus Schwabedal, carlosdanielcsantos, Hvass-Labs, Mikhail Pak, Fred Callaway, Loïc Estève, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. 2018. *scikit-optimize/scikit-optimize: v0.5.2*. https://doi.org/10.5281/zenodo.1207017

[11] Igor Kononenko. 1993. Inductive and Bayesian learning in medical diagnosis. *Applied Artificial Intelligence an International Journal* 7, 4 (1993), 317–337.

[12] Intel Labs. 2021. Intel Loihi 2 Technology Brief. https://www.intel.com/content/www/us/en/research/neuromorphic-computing-loihi-2-technology-brief.html

[13] Intel Labs. 2023. Lava Deep Learning. https://github.com/lava-nc/lava-dl

[14] David Middleton, Institute of Electrical, and Electronics Engineers. 1960. *An introduction to statistical communication theory*. Vol. 960. McGraw-Hill New York.

[15] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. 2015. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience* 9 (2015). https://doi.org/10.3389/fnins.2015.00437

[16] Maryam Parsa, Aayush Ankit, Amirkoushyar Ziabari, and Kaushik Roy. 2019. Pabo: Pseudo agent-based multi-objective bayesian hyperparameter optimization for efficient neural accelerator design. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[17] Maryam Parsa, Shruti R Kulkarni, Mark Coletti, Jeffrey Bassett, J Parker Mitchell, and Catherine D Schuman. 2021. Multi-Objective Hyperparameter Optimization for Spiking Neural Network Neuroevolution. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1225–1232.

[18] Maryam Parsa, John P Mitchell, Catherine D Schuman, Robert M Patton, Thomas E Potok, and Kaushik Roy. 2020. Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design. *Frontiers in neuroscience* 14 (2020), 667.

[19] Maryam Parsa, Catherine Schuman, Nitin Rathi, Amir Ziabari, Derek Rose, J Parker Mitchell, J Travis Johnston, Bill Kay, Steven Young, and Kaushik Roy. 2021. Accurate and Accelerated Neuromorphic Network Design Leveraging A Bayesian Hyperparameter Pareto Optimization Approach. In *International Conference on Neuromorphic Systems 2021*. 1–8.

[20] James S. Plank, Catherine D. Schuman, Grant Bruer, Mark E. Dean, and Garrett S. Rose. 2018. The TENNLab Exploratory Neuromorphic Computing Framework. *IEEE Letters of the Computer Society* 1, 2 (2018), 17–20. https://doi.org/10.1109/LOCS.2018.2885976

[21] Svetlozar T Rachev, John SJ Hsu, Biliana S Bagasheva, and Frank J Fabozzi. 2008. *Bayesian methods in finance*. John Wiley & Sons.

[22] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[23] Sheldon M. Ross. 2017. Chapter 8 - Estimation. In *Introductory Statistics (Fourth Edition)* (fourth edition ed.), Sheldon M. Ross (Ed.). Academic Press, Oxford, 329–380. https://doi.org/10.1016/B978-0-12-804317-2.00008-4

[24] Catherine D. Schuman, J. Parker Mitchell, Robert M. Patton, Thomas E. Potok, and James S. Plank. 2020. Evolutionary Optimization for Neuromorphic Systems. In *Proceedings of the 2020 Annual Neuro-Inspired Computational Elements Workshop* (Heidelberg, Germany) *(NICE '20)*. Association for Computing Machinery, New York, NY, USA, Article 2, 9 pages. https://doi.org/10.1145/3381755.3381758

[25] Catherine D Schuman, James S Plank, Adam Disney, and John Reynolds. 2016. An evolutionary optimization framework for neural networks and neuromorphic architectures. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 145–154.

[26] Di Sha, Kaan Ozbay, and Yue Ding. 2020. Applying Bayesian optimization for calibration of transportation simulation models. *Transportation Research Record* 2674, 10 (2020), 215–228.

[27] Jan A Snyman, Daniel N Wilke, et al. 2005. *Practical mathematical optimization*. Springer.

[28] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. 2019. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics* 50, 8 (2019), 3668–3681.

[29] Yu Wang and Justin Solomon. 2019. Chapter 2 - Intrinsic and extrinsic operators for shape analysis. In *Processing, Analyzing and Learning of Images, Shapes, and Forms: Part 2*, Ron Kimmel and Xue-Cheng Tai (Eds.). Handbook of Numerical Analysis, Vol. 20. Elsevier, 41–115. https://doi.org/10.1016/bs.hna.2019.08.003

Received 30 April 2023