2019 IEEE International Conference on Big Data (Big Data)

Bayesian-based Hyperparameter Optimization for Spiking Neuromorphic Systems

Maryam Parsa
Department of ECE, Purdue University
West Lafayette, Indiana, USA
mparsa@purdue.edu

J. Parker Mitchell
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
mitchelljp1@ornl.gov

Catherine D. Schuman
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
schumancd@ornl.gov

Robert M. Patton
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
pattonrm@ornl.gov

Thomas E. Potok
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
potokte@ornl.gov

Kaushik Roy
Department of ECE, Purdue University
West Lafayette, Indiana, USA
kaushik@purdue.edu

***Abstract*—Designing a neuromorphic computing system involves selection of several hyperparameters that not only affect the accuracy of the framework, but also the energy efficiency and speed of inference and training. These hyperparameters might be inherent to the training of the spiking neural network (SNN), the input/output encoding of the real-world data to spikes, or the underlying neuromorphic hardware. In this work, we present a Bayesian-based hyperparameter optimization approach for spiking neuromorphic systems, and we show how this optimization framework can lead to significant improvement in designing accurate neuromorphic computing systems. In particular, we show that this hyperparameter optimization approach can discover the same optimal hyperparameter set for input encoding as a grid search, but with far fewer evaluations and far less time. We also show the impact of hardware-specific hyperparameters on the performance of the system, and we demonstrate that by optimizing these hyperparameters, we can achieve significantly better application performance.**

***Index Terms*—Hyperparameter Optimization, Spiking Neuromorphic Computing, Accurate and Energy Efficient Machine Learning**

I. Introduction

The inherent challenges of cloud computing infrastructures such as power management and sustainability, that emerged with the ever-increasing amount of data [1], inspired seeking alternative computing platforms such as spiking neuromorphic systems. These biologically-inspired computing platforms not only offer tremendous energy efficiency for computing in resource-constrained environments such as mobile and edge devices, but also extend the ability to solve challenging machine learning problems due to its massive connectivity of synthetic neurons and synapses [2]. The in-memory computing capability of neuromorphic systems proposes a promising alternative or complement to von Neumann architectures that suffer from the low bandwidth between CPU and memory, also known as the von Neumann bottleneck [3]. In addition, the brain-like structure of spiking neuromorphic systems is suitable for on-line, real-time learning for certain tasks such as smart healthcare diagnosis on edge devices, special purpose applications on drones, and robotics.

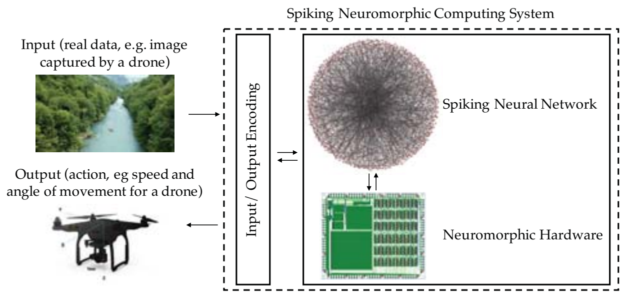
Spiking Neuromorphic Computing System
Input (real data, e.g. image captured by a drone)
Output (action, eg speed and angle of movement for a drone)
Input/ Output Encoding
Spiking Neural Network
Neuromorphic Hardware

Fig. 1. High-level Overview of a Spiking Neuromorphic Computing System

As shown in Figure 1, a spiking neuromorphic computing systems consists of two key building blocks, namely a spiking neural network (SNN) and an underlying neuromorphic hardware. There are different techniques available to train an SNN [4]–[6]. As for the spike-based neuromorphic systems there have been various CMOS-based [7]–[9] and beyond-CMOS [10] neuromorphic accelerators introduced in literature.

Each of these building blocks have inherent hyperparameters that directly affect the final performance of the system, such as accuracy, energy efficiency, inference time, and network size. While one set of hyperparameters might satisfy one or all performance metrics, a minor change can drastically alter the resulting performance.

Bayesian optimization is among the most prominent techniques to optimize hyperparameters of networks that are

This manuscript has been authored [in part] by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

expensive to evaluate. This stochastic method is suitable for problems with unknown objective functions such as the accuracy of spiking neural network or the energy efficiency of a neuromorphic hardware [11]–[13].

In this work, we propose a Bayesian-based hyperparameter optimization framework that defines the optimum set of hyperparameters with only a few evaluations of SNNs. This iterative, heuristic optimization algorithm predicts how performance matrices change by altering hyperparameters. Instead of optimizing a black-box, expensive objective function (i.e., the performance metrics), the optimum set of hyperparameters is predicted by approximating a posterior Gaussian distribution and optimizing a surrogate model. We consider several types of hyperparameters ranging from input encoding to SNN's training and neuromorphic hardware implementation. We perform sensitivity analysis on various hyperparameter sets and demonstrate how critical some sets of hyperparameters are, directly impacting the performance of the system. We further analyze different input encoding schemes and show how combining multiple schemes might boost the performance of the system.

We made the following contributions:

1) We propose a simple, effective, and generalizable hyperparameter optimization algorithm that can be applied to any type of spiking neuromorphic algorithm or architecture.
2) We perform a sensitivity analysis on spiking neuromorphic system hyperparameters, discussing strategic role of some sets of hyperparameters on the system's final performance.
3) We show that hyperparameters of a resilient training framework for spiking neuromorphic systems such as EONS [4] have the least impact on the final performance of the system compared to the input encoding or hardware-specific hyperparameters.
4) To best of our knowledge, for the first time in the literature, we introduce a hyperparameter optimization technique for spiking neuromorphic computing system and analyse the effect of different types of hyperparameters on the overall performance of the system

II. Background and Related Work

With the ambition to mimic the biological brain, the core of any spiking neuromorphic system is millions of neurons and synapses communicating with asynchronous spikes. This high-dimensional, and nonlinear computational framework, although offering exceptional assets, pose fundamental hurdles when deploying in real applications [2], [14]. A high-performance neuromorphic system requires substantial competence in encoding real data to spikes and vice-versa, modeling neurons/synapses, designing neuromorphic algorithms and architectures, and advancements in neuromorphic spike-based hardware.

Figure 1 demonstrates an overview of a spiking neuromorphic system. The input/output (I/O) encoding block is an indispensable bridge that is necessary to input real data as spikes into neuromorphic systems. For example, in the case of controlling a drone, the real data are the images captured by the camera on the drone. In the I/O coding block, these input images will be encoded to spikes and used to train a spiking neural network on a neuromorphic hardware. The output of the spiking network is decoded to the real data, which are the drone control actions for this example (speed, angle of movement, etc). The data conversion (input data to spikes and output spikes to decisions) has a substantial effect on the performance of the computing system, not only on the accuracy, but also on energy efficiency. Binning-based, rate-based, and temporal-based coding schemes are among the most common approaches to code real data to spikes and vise-versa [15].

There are different techniques to train a densely connected network of neurons and synapses. These include deep spiking neural networks (SNN) with back-propagation using spike-timing dependent plasticity (STDP) [16], [17], genetic algorithm-based training such as EONS [4], [18], SNNs created by converting traditional artificial neural networks (ANN) [5] with binary communications [6], liquid state machines and reservoir computing [19]. Within each of these training approaches are several inherent hyperparameters that could directly impact the performance of the spiking neuromorphic system. These include but not limited to ANN's hyperparameters (such as number of layers) and the conversion parameters such as sharpening values for this SNNs trained by converting ANN to SNN [6], back-propagation and STDP hyperparameters [17], and population size, crossover and mutation rates for genetic algorithm-based trainings [4].

There are also several hardware-specific hyperparameters such as the ranges and resolutions of neuron thresholds and synaptic weights [7], and number of crossbars and crossbar sizes for memristor-based neuromorphic devices [10].

Grid search, random search and stochastic grid search [12], [20], [21] are among the most common approaches to find the optimum set of hyperparameters. Other iterative approaches such as simulated annealing (SA), genetic algorithms (GA), and Bayesian optimization are also introduced to find the optimum set of HPs for an artificial neural network [12], [13], [22]. These techniques are mostly focused on the mathematics of the optimization problem regardless of the type of hyperparameters and their impact on the overall performance of the system. To the best of our knowledge, for the first time in literature, we tackle hyperparameter optimization for spiking neuromorphic systems; in particular, we use Bayesian optimization to not only determine the optimum set of HPs, but also to analyze how the optimization procedure moves toward the optimum point, and the impacts of different types of HPs on the final performance.

III. Hyperparameter Optimization Framework

Designing a spiking neuromorphic system requires several testbeds such as specialized hardware, software, network configuration, and application. To this end, we use the TENNLab Exploratory Neuromorphic Computing Framework [23] that not only provides support to develop and test various neuromorphic applications and devices, but also is suitable for hyperparameter optimization problems that require concurrent inference on neuromorphic applications and devices for different sets of hyperparameters and evaluation metrics. In addition, the primary learning algorithm used in this framework is EONS (Evolutionary Optimization for Neuromorphic Systems) [4], which is a well-suited approach for our hyperparameter optimization problem. This is mainly due to the flexibility of using this approach with different input/output coding schemes without the need to change the entire algorithm [15].

Bayesian optimization methods are widely used to solve problems where the objective functions are expensive to evaluate. In hyperparameter optimization for spiking neuromorphic systems, objective functions are different performance metrics vary from network accuracy, and inference time to hardware energy consumption or area usage. The Bayesian-based hyperparameter optimization approach that we propose can be applied to any spiking neuromorphic system on any underlying neuromorphic hardware. For the purpose of this work, we use TENNLab framework with one neuromorphic application (pole-balance [24], [25]), on one CMOS-based neuromorphic hardware (DANNA2 [7]).

A. TENNLab Framework

1) Input/Output Coding Module: Neuromorphic devices and architectures run only on spikes; however, the real-world applications communicate their states through values, and only accept values as their input. The TENNLab framework facilitates value to spike encoding and vice-versa through an encoding/decoding module. This module accepts different state-of-the-art input/output coding schemes such as direct, binning [15], rate coding [16], and temporal coding [18]. In this work we considered a combination of several coding techniques that can handle a wide range of data within a short time period for real-time online applications (specially for edge devices) [15].

- ***Binning***. Encoding an input with single spike in a single time step, through creating multiple neurons and spiking on a particular neuron based on the value of the input.
- ***Spike-count***. Converting input values to a number of spikes at a fixed rate.
- ***Charge-injection***. Injecting a specific charge value into a neuron rather than a fixed magnitude spike.
- ***Combined coding schemes***. Combining all above techniques in a hierarchy by introducing three different inter-bin functions of *simple, flip-flop, and triangle*. *Simple* inter-bin function linearly maps the values between the preassigned charge values for each bin, whereas in *flip-flop*, this mapping is only for odd-numbered bins and then flips for even-numbered ones. This function enables continuous mapping across the bins. Finally, the *triangle* inter-bin function creates overlapped bins to encourage smooth mapping.

Details of the above input/output coding schemes are given in [15]. Each of these methods have inherent hyperparameters that directly impact the performance of the system. Here our main focus is finding the optimum set of input encoding hyperparameters to maximize the performance of spiking neuromorphic system for a specific application and hardware.

2) EONS: EONS trains a graph representation of a spiking neural network for a neuromorphic hardware system using Genetic Algorithms (GA). This graph representation of a neuromorphic network generates an initial population of random graphs based on the application inputs and hardware specifics. Next generations are produced by duplicating, merging, mutation or crossover on current generations based on their fitness values. The resulting generations are converted to spiking neural networks, and the new performance values are calculated. The process repeats until the desired fitness is reached or terminated due to the running time [4], [23].

3) Neuromorphic Hardware: EONS is a graph based representation of a spiking neural network. Neurons and synapses are the building blocks of this graph and are inherent to the device characteristics. Synaptic weights and delays ranges, neurons thresholds, as well as their leaky rates or plasticity parameters are defined by the neuromorphic hardware and are also targeted in our proposed optimization framework.

B. Bayesian-based Hyperparameter Optimization

Bayesian optimization is a derivative-free technique that estimates an unknown objective function by building a surrogate model based on Gaussian distributions for the known observations. The procedure starts with evaluating the objective function for two random sets of hyperparameters. A posterior Gaussian process is estimated using the two known observations. An acquisition function is then calculated as a surrogate model to optimize in each iteration. There are different techniques to calculate an acquisition function, which are all based on building a surrogate model that explores and exploits the search space. By exploration we mean evaluating the objective function for different sets of HPs in various locations of search space to avoid trapping in local minimums, and exploitation means to leverage the decent available sets of HPs. In this work we used *Expected Improvement* acquisition function [12]. The optimum point of the acquisition function (surrogate model) is the best next set of HP to evaluate in the next iteration. For all previous observations as well as the new set of HP, we re-estimate the Gaussian distribution, and repeat the process of calculating the surrogate model, and optimizing it to find the best next set of HP. The process repeats until the best next point is already observed or until termination.

C. Experimental Setup

In this work we use TENNLab's framework [23] for input encoding, EONS, and neuromorphic hardware hyperparameter (HP) optimization to find the optimum set of HPs that maximizes the overall performance of the system. We considered the fitness value (accuracy) as the performance metric of the system; however, this can be easily modified to any other metric such as energy requirement or inference time.

TABLE I
CASE STUDY ONE: EVALUATED PARAMETERS

Hyperparameter	Range
b_k (Number of bins)	1,2,4,8
p_k (Number of pulses per bins, i.e. spikes)	1,2,4,8
$[c_k, C_k]$ (Range of charge values)	[0,0.5],[0,1],[0.25,0.5], [0.25,1],[0.5,0.5],[1,1]
Inter-bin encoding function	simple, flip-flop, triangle

TABLE II
CASE STUDY ONE: FIXED PARAMETERS

Parameter	Value
Interval	1
population size	1000
Mutation rate	0.9
Crossover rate	0.5
Synaptic weight	[-255,255]
Neuron threshold	[0, 1023]
Synaptic delay	127

For all different case studies, we utilize a canonical pole balancing control task [24], [25] on the DANNA2 neuromorphic hardware [7]. DANNA2 is a digital programmable device with integrate-and-fire neurons, and can be deployed either on an FPGA or a custom chip.

For the Bayesian optimization framework, we chose Matern covariance function with the Kernel function shown in Equation 1.

$$C_\nu(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \Big(\sqrt{2\nu}\frac{d}{\rho}\Big)^\nu K_\nu\Big(\sqrt{2\nu}\frac{d}{\rho}\Big) \tag{1}$$

where d is a distance function, Γ is the gamma function, K_ν is the modified Bessel function of the second kind, ρ and ν are positive parameters. For this paper, we found out the fastest most accurate HP optimization results are obtained when $\rho = 1$, and $\nu = 1.5$. We selected Matern kernel function due to smooth Gaussian distribution estimate it provides.

IV. RESULTS

We investigate and validate our Bayesian-based hyperparameter optimization approach across four case studies. For the first case study, we start with finding the optimum set of HP for the input encoding hyperparameters of an SNN. For this problem, a grid search technique has previously been applied for all HP combination settings and thus, the optimum HP set is known [15]. Table I shows the possible values for different HPs. The ranges are all based on reasonable and acceptable values for each of the HPs and the combinations that do not make sense are removed from the search space. The HPs that are fixed and not included in the optimization search are summarized in Table II.

Figure 2 shows the grid search results for hyperparameter combinations shown in Table I compared to the results obtained from our proposed hyperparameter optimization technique. The grid search results are achieved after 100 runs on each of the valid 240 different HP combinations [15]. Our proposed Bayesian-based HP optimization technique reaches

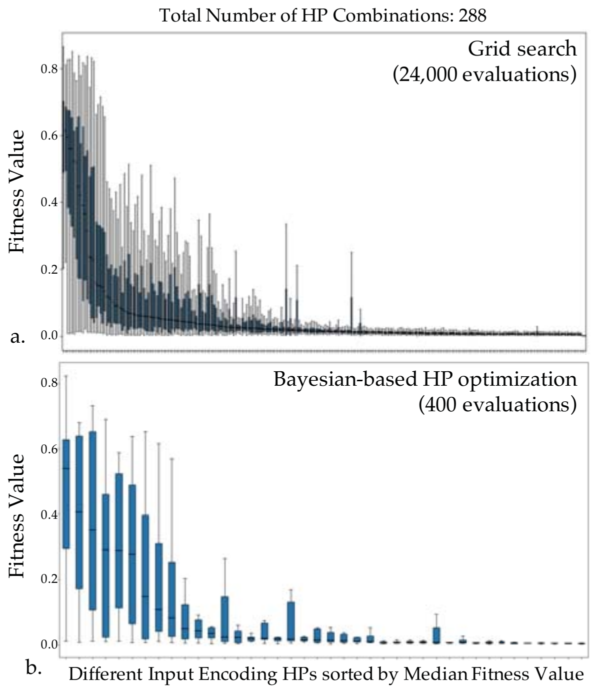

Fig. 2. Case study one: Comparing grid search with HP optimization for problem with HP combinations shown in Table I. a. Grid search: 100 runs for each of the valid 240 different HP sets according to [15]. b. Bayesian-based HP optimization: 10 runs for selected 40 HP combinations. Both techniques report the same optimum HP set ($b_k = 2$, $p_k = 8$, $charge = [0, 0.5]$, $function = flip - flop$) with median fitness value of 52%.

to exactly the same optimum HP combination, only after 40 iterations, and for only 10 runs for each of the HP combinations evaluated. This reduces the number of evaluations required in order to find the optimum point from 24,000 to 400.

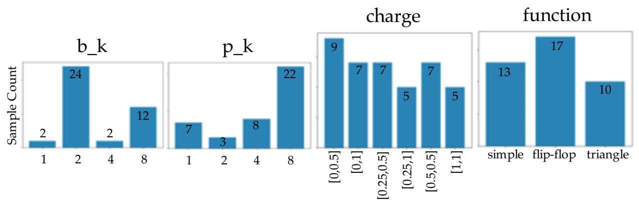

Fig. 3. Case study one: Histogram of the HP combinations for 40 evaluations.

The iterative Bayesian-based HP optimization finds the optimum set of HPs by exploring and exploiting the search space. This means that it not only maintains the HP combinations that creates the highest fitness values, but also explores the search space to avoid trapping in local minima. The frequency of selecting the value for each hyperparameter for case study one is shown in Figure 3. The hyperparameter value with maximum number of calls is consistent with the optimum HP set defined by the optimization technique ($b_k = 2, p_k = 8, charge = [0, 0.5], function = flip - flop$).

For case study two, we change one of the fixed hyperparameter values related to the hardware implementation (the

TABLE III
COMPARING FITNESS VALUE FOR DIFFERENT CASE STUDIES

	Input Encoding HPs					EONS HPs			Hardware HPs			Fitness Value
	b_k	p_k	charge	inter-bin function	interval	population size	mutation rate	crossover rate	synaptic weight	neuron threshold	synaptic delay	
Case study one	2	8	$[0, 0.5]$	flip-flop	1	1000	0.9	0.5	$[-255, 255]$	$[0, 1023]$	127	52%
Case study two	2	8	$[0, 0.5]$	flip-flop	1	1000	0.9	0.5	$[-511, 511]$	$[0, 1023]$	127	60%
Case study three	2	8	$[0, 0.5]$	flip-flop	1	1000	0.9	0.5	$[-511, 511]$	$[0, 1023]$	255	67.40%
Case study four	2	12	$[0, 0.5]$	flip-flop	5	1500	0.9	0.4	$[-127, 127]$	$[0, 1023]$	255	70.99%

synaptic weight range) and repeat the experiment from case study one. Table III shows that if we maintain the exact simulation environment and HP set for case studies one and two, and only change the synaptic weight range given in Table II, the fitness value will increase from 52% to 60%. For case study three, in addition, we also increase the synaptic delay range. In this case, the resulting accuracy will increase to 67.40%. This proves the need to optimize not only the input encoding HPs of the spiking neuromorphic system, but also other HPs such as EONS or hardware-related ones.

In case study four we considered three types of HP combinations; input encoding related, EONS, or hardware-related hyperparameters. The summary of the evaluated parameters and their corresponding ranges are given in Table IV. In this experiment the total number of hyperparameter combinations is $54,432,000$. This shows how drastically the number of hyperparameter sets increase in real problems where there are multiple HPs involved in different modules, frameworks, algorithm, and architectures. In Figure 4 we plotted the median fitness value for 50 HP combinations sets (50 iterations of Bayesian optimization) each repeated for 100 times. It is evident from the figure that the method ensures obtaining the optimum HP set through exploring the search space by evaluating HP combinations with low fitness values that were chosen outside the predicted range of optimum HPs, while performing exploitation by maintaining the predicted optimum HPs in overall higher fitness value domains.

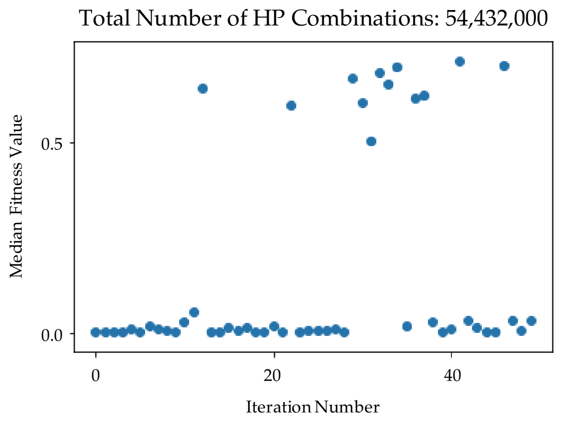

Fig. 4. Case study four: Median fitness value after 100 runs with 50 different HP evaluations for the parameters given in Table IV. The optimum set of HP combination in this case study is shown in Table III with median fitness value of 70.99%

TABLE IV
CASE STUDY FOUR: EVALUATED PARAMETERS

	Parameter	Range
Input Encoding HPs	b_k (Number of bins)	2, ... , 8
	p_k (Number of spikes)	1, ... , 12
	$[c_k, C_k]$ (Range of charge values)	[0,0.5],[0.1,],[0.25,0.5],[0.25,1],[0.5,0.5],[1,1]
	Inter-bin encoding function	simple, flip-flop, triangle
	Interval	1, ... , 5
EONS HPs	Population size	600, 800, 1000, 1200, 1500, 2000
	Mutation rate	0.6, 0.7, 0.8, 0.9
	Crossover rate	0.3, 0.4, 0.5, 0.6, 0.7
Hardware HPs	Synaptic Weights	[-127, 127], [-255, 255], [-511, 511], [-1023, 1023]
	Neuron Thresholds	255, 511, 1023
	Synaptic Delay	15, 31, 63, 127, 255

Figure 5 demonstrates a partial dependence plot for non-categorical hyperparameters in case study four. Inter-bin function types and population size are considered as categorical HPs and therefore not shown in this plot. The black dots are the set of parameters we have evaluated and the red dot is the best parameter we found. In these plots colors are the surrogate model built by the Gaussian Process. Light regions are the best (highest fitness values) while the darker ones are the worst. To make each of these partial dependence plots, we make a grid for a set of two parameters. We then calculate the surrogate model with fix values for those two parameters while generating random values for all other parameters. We repeat the process and average the fitness values and plot the color map. These plots are only used for extra analysis on the optimization search process and are not deterministic due to the inherent variability of the built surrogate models for Gaussian distribution with random parameter selections. In this figure for some HPs the counter plots show the convergence toward the optimum values. For example for p_k versus $synapticdelay$ partial dependence plot it is clear that the higher the value for both parameters is the better performance. However for all partial dependence plot for parameter b_k, as long as the optimum set stays in any of the light regions, it does not matter what combination we choose for the next iteration.

A. Key Observations

1) Observation One: Different hyperparameters have different impact on the final performance of the neuromorphic

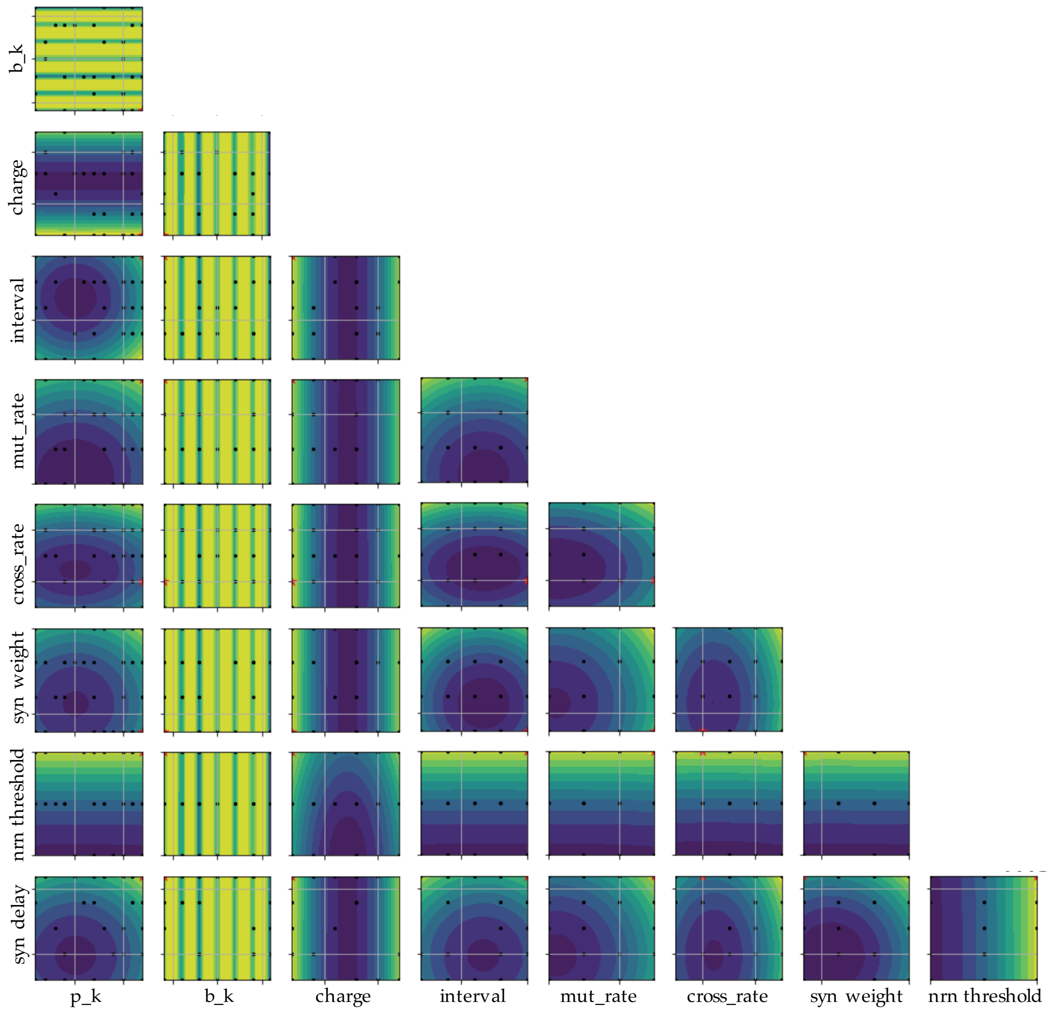

Fig. 5. Case study four: Partial dependence plot

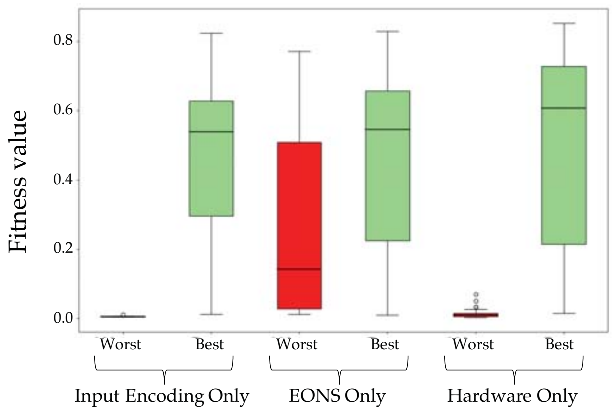

Fig. 6. Observation: Sensitivity analysis on different types of HPs. Comparing the fitness values for best and worst HP combinations

system. Figure 6 demonstrates how critical it is to select the optimum set of input encoding and hardware hyperparameters to obtain the maximum fitness value. In both cases the worst HP combination lead to almost 0% fitness value. However, EONS hyperparameters such as crossover and mutation rates have less impact on the final performance of the system. This shows resiliency of EONS framework. Details of the HP sets in each network is given in Table V.

2) Observation Two: Table III shows a minor change on critical hyperparameters of the system could drastically improve the overall performance.

V. Conclusion and Future Work

Neuromorphic computing systems provide a potential solution for energy efficient machine learning. However, there are many aspects of spiking neuromorphic computing systems that are not well understood, including how to input information into the spiking neuromorphic computing system, how to train associated spiking neural networks for these systems, and hardware details themselves. In this work, we demonstrate a Bayesian-based hyperparameter optimization approach for neuromorphic computing systems. We show that this approach

TABLE V
OBSERVATION ONE: EVALUATED PARAMETERS FOR BEST AND WORST NETWORKS FOR ISOLATED HP OPTIMIZATION ANALYSIS

	Best Network	Worst Network
Only Input Encoding HPs	$b_k = 2$	$b_k = 8$
	$p_k = 8$	$p_k = 1$
	$[c_k, C_k]$ = $[0, 0.5]$	$[c_k, C_k]$ = $[0.5, 0.5]$
	function = flip-flop	function = simple
	Interval = 1	Interval = 1
Fitness Value	52%	0.4%
EONS HPs	Population size = 1000	Population size = 1000
	Mutation rate = 0.9	Mutation rate = 0.6
	Crossover rate = 0.6	Crossover rate = 0.3
Fitness Value	54.4%	25.2%
Hardware HPs	Synaptic Weights = $[-511, 511]$	Synaptic Weights = $[-1023, 1023]$
	Neuron Thresholds = $[0, 1023]$	Neuron Thresholds = $[0, 255]$
	Synaptic Delay = 127	Synaptic Delay = 15
Fitness Value	61.86%	0.8%

can discover the appropriate hyperparameters for input encoding for neuromorphic systems in many fewer iterations than a grid search. We also show that selecting the appropriate hyperparameters can have a tremendous impact on application performance.

For future work, we intend to expand this framework to multiple applications, neuromorphic hardware implementations, and other neuromorphic training approaches. We also intend to explore hyperparameter optimization for multi-objective optimization approaches. In this work, we focused simply on improving the accuracy of the system on an application. For our multi-objective approach, we may also consider additional factors, such as time-to-solution, energy efficiency, and working within hardware design constraints. In general, we will continue to investigate approaches for automatically optimizing the hyperparameters for all aspects of neuromorphic computing systems so that they can be more easily and rapidly applied to real-world applications.

ACKNOWLEDGEMENTS

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number DE-AC05-00OR22725.

REFERENCES

[1] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: A review," *IEEE transactions on neural networks and learning systems*, 2019.

[2] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[3] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Communications of the ACM*, vol. 57, no. 6, pp. 13–15, 2014.

[4] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds, "An evolutionary optimization framework for neural networks and neuromorphic architectures," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 145–154.

[5] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, 2019.

[6] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Whetstone: A method for training deep artificial neural networks for binary communication," *arXiv preprint arXiv:1810.11521*, 2018.

[7] J. P. Mitchell, M. E. Dean, G. R. Bruer, J. S. Plank, and G. S. Rose, "Danna 2: Dynamic adaptive neural network arrays," in *Proceedings of the International Conference on Neuromorphic Systems*. ACM, 2018, p. 10.

[8] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[9] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[10] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-m. W. Hwu, J. P. Strachan, K. Roy *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2019, pp. 715–731.

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

[12] J. S. Bergstra and e. al., "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[13] M. Parsa, A. Ankit, A. Ziabari, and K. Roy, "Pabo: Pseudo agent-based multi-objective bayesian hyperparameter optimization for efficient neural accelerator design," *arXiv preprint arXiv:1906.08167*, 2019.

[14] M. Davies, "Benchmarks for progress in neuromorphic computing," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 386–388, 2019.

[15] C. D. Schuman, J. S. Plank, G. Bruer, and J. Anantharaj, "Non-traditional input encoding schemes for spiking neuromorphic systems," in *JCNN: The International Joint Conference on Neural Networks*, Budapest, 2019.

[16] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *The 2011 International Joint Conference on Neural Networks*. IEEE, 2011, pp. 1775–1781.

[17] C. Lee, S. S. Sarwar, and K. Roy, "Enabling spike-based backpropagation in state-of-the-art deep neural network architectures," *arXiv preprint arXiv:1903.06379*, 2019.

[18] J. V. Arthur and K. Boahen, "Learning in silicon: Timing is everything," in *Advances in neural information processing systems*, 2006, pp. 75–82.

[19] P. Wijesinghe, G. Srinivasan, P. Panda, and K. Roy, "Analysis of liquid ensembles for enhancing the performance and accuracy of liquid state machines," *Frontiers in neuroscience*, vol. 13, p. 504, 2019.

[20] H. Larochelle and e. al., "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 473–480.

[21] J. Bergstra and e. al., "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[22] J. Snoek and e. al., "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[23] J. S. Plank, C. D. Schuman, G. Bruer, M. E. Dean, and G. S. Rose, "The tennlab exploratory neuromorphic computing framework," *IEEE Letters of the Computer Society*, vol. 1, no. 2, pp. 17–20, 2018.

[24] A. P. Wieland, "Evolving neural network controllers for unstable systems," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. 2. IEEE, 1991, pp. 667–673.

[25] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Efficient non-linear control through neuroevolution," in *European Conference on Machine Learning*. Springer, 2006, pp. 654–662.