Multi-Objective Optimization for Size and Resilience of Spiking Neural Networks

Mihaela Dimovska,[1] Travis Johnston,[2] Catherine D. Schuman,[2]
J. Parker Mitchell,[2] Thomas E. Potok [2]
[1]*Department of Electrical and Computer Engineering,*
University of Minnesota, Minneapolis, USA
dimov003@umn.edu
[2]*Oak Ridge National Laboratory, Oak Ridge, USA*
{johnstonjt, schumancd, mitchelljp1, potokte}@ornl.gov

***Abstract*—Inspired by the connectivity mechanisms in the brain, neuromorphic computing architectures model Spiking Neural Networks (SNNs) in silicon. As such, neuromorphic architectures are designed and developed with the goal of having small, low power chips that can perform control and machine learning tasks. However, the power consumption of the developed hardware can greatly depend on the size of the network that is being evaluated on the chip. Furthermore, the accuracy of a trained SNN that is evaluated on chip can change due to voltage and current variations in the hardware that perturb the learned weights of the network. While efforts are made on the hardware side to minimize those perturbations, a software based strategy to make the deployed networks more resilient can help further alleviate that issue. In this work, we study Spiking Neural Networks in two neuromorphic architecture implementations with the goal of decreasing their size, while at the same time increasing their resiliency to hardware faults. We leverage an evolutionary algorithm to train the SNNs and propose a multi-objective fitness function to optimize the size and resiliency of the SNN. We demonstrate that this strategy leads to well-performing, small-sized networks that are more resilient to hardware faults.**

***Index Terms*—Neuromorphic Computing, Spiking Neural Networks, Multi-objective, Fault Tolerance, Evolutionary Optimization**

I. Introduction

With the advent of internet of things, smaller sensors and smarter environments are on the rise [1]. This increases the demand for low power hardware that can preform machine learning tasks [2], [3]. Neuromorphic computing architectures are promising hardware architectures that fulfill these requests [4]–[6]. These architectures are non-von Neumann chips that are inherently low power and parallel as they model a biological neural system via Spiking Neural Networks (SNNs) implementations in silicon [4], [7]–[9]. However, neuromorphic hardware implementations of SNNs tend to suffer from environmental perturbations due to current or voltage variations or bit flips, among others, with the synapse weight being the most critical part susceptible to perturbations [10], [11]. The perturbations that lead to a change in the synaptic parameters of the implemented spiking neural network often result in a greatly degraded performance of the network [11]. At the same time, smaller sensors require smaller networks due to power restrictions [12]. Thus, taking neuromorphic computing to the next step and developing small size SNNs that are also resilient to perturbations of their parameters is a necessity.

In this work we leverage an evolutionary optimization framework called EONS [13], [14] to train spiking neural networks. This evolutionary approach has been shown to lead to well-performing SNNs, as demonstrated on several machine learning and control tasks [6], [15], [16]. At the core of the evolutionary (genetic) algorithm is the fitness function which evaluates every network in a population. We propose a multi-objective fitness function which incorporates a penalty for the number of neurons in a network as a way to generate smaller sized networks. We further include the performance of several variations of the network that is currently evaluated to produce networks that are resilient to some particular kind of perturbations that are possible to be encountered in the hardware. We show that this is a flexible approach for generating well performing, small sized SNNs that are more resilient to hardware faults.

II. Background

Reducing the size of artificial neural networks is an ongoing quest in science and technology, ever more so with the development of low-power devices for machine learning [17]. In the area of deep learning, there are many different reduction and pruning methods that aim to decrease the size of the powerful but over-parameterized deep neural networks that are deployed on traditional CPU or GPU architectures [18]–[21]. On the other hand, for *spiking* neural networks, more efforts have been devoted to their training [8], [22] and not so much to their efficiency. The problems of size reduction and hardware

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy.

perturbation resilience (noise sensitivity) for SNNs have been considered in limited scenarios. For example, for sparse SNNs, there are algorithms that map an SNN to a specific hardware, transforming the SNN in order to match some of the hardware constraints, such as power consumption or memory access latency [23], [24]. However, these methods either require sparse SNNs already or they sparsify the SNN, but there are no guarantees that the initial accuracy of the network will be preserved after the transformation. In this work we propose a multi-objective evolutionary training method for SNNs that optimizes the network for hardware constraints such as lower power consumption and fault tolerance, while providing some confidence in the performance of the network via the training and testing accuracy.

Multi-objective evolutionary training for neural networks has been used in several works though the objectives have been mostly centered around performance metrics or classification sensitivity or specificity [25], [26]. Furthermore, training spiking neural networks to be fault tolerant, i.e., addressing certain hardware fault tolerance limitations, to the best our knowledge, is a research topic that is only beginning to emerge. For example, only a very recent work addresses the noise-handling limitations of hardware implementations of spiking neurons from a software (algorithmic) perspective [27]. The work in [27] adds noise to the threshold of the spiking neuron in the process of evolutionary training of the SNN, thus effectively making the trained SNN more resilient to noise fluctuations in hardware.

In this work, we consider multi-objective fitness function in the evolutionary process to train a network that is resilient to synaptic perturbations, which are one of the most critical structures in neuromorphic hardware that are susceptible to perturbations [10], [11]. The other objective in the multi-objective fitness function is to minimize the number of neurons in the SNN. We show that by penalizing for the number of neurons we can significantly reduce the number of synapses in the network as well. We demonstrate these results on two neuromorphic implementations and via several applications. We explain the genetic algorithm and the neurmorphic implementations considered in the following subsections.

A. Genetic Algorithm for Training SNNs

In this work, we use a genetic algorithm-based training approach called Evolutionary Optimization for Neuromorphic Systems (EONS) [13]. This approach evolves both the parameters of the network (e.g., weights of synapses and thresholds of neurons), as well as the structure of the network (e.g., number of neurons and synapses and how they are connected). In all of the experiments, the following parameters were used during the evolutionary process: population size of 100, crossover rate of 0.5, mutation rate of 0.9, and merge rate of 0.1.

B. DANNA2

DANNA2 (Dynamic Adaptive Neural Network Arrays) [6] is a synchronous digital neuromorphic architecture with integrate-and-fire neurons and optional synaptic plasticity. Networks are represented as a directed graph in a two-dimensional space using neurons as graph nodes and synapses as edges. When a neuron's charge exceeds the configured 10-bit threshold, the neuron fires and enters a configurable refractory period in which it temporarily may not fire again. Fires from a neuron travel along synapses with an individually configurable 4-bit temporal delay and signed 9-bit weight value.

DANNA2 may be simulated on a traditional CPU or be implemented on an FPGA or ASIC. In this work, DANNA2 sparse is used which allows for a network to be converted into an optimized hardware implementation. The resulting hardware directly builds the network graph and removes any unnecessary functionality which allows for improved efficiency.

C. NIDA

Neuroscience-inspired dynamic architectures or NIDA [28] is a simple spiking neural network implementation using integrate-and-fire neurons and assumes analog synaptic weight values (specifically, floating point values between -1 and 1). Neurons are laid out in three-dimensional space and the delays between neurons depend on the distance between neurons in the space. NIDA is implemented in simulation only.

III. Main Results

In this section we present the main contribution of this article: a multi-objective fitness function that allows for the evolution of smaller-size SNNs that are also resilient to particular hardware perturbations of the synaptic weights. Another contribution of this article is the analysis of two network size-reducing strategies: during-training size optimization and post-training pruning. Namely, we provide empirical results that suggest it is better to constrain the size of a network during the training process, as opposed to pruning the network after training. We also show that penalizing for the number of neurons during the process of training actually leads to a much smaller number of synapses as well. The next subsection explores these two different size-reduction strategies.

A. Size Optimization

There are two main approaches to reducing the size of a spiking neural network: a post-training approach and during-training approach. First, we examine the post-training approach of pruning the network by removing neurons that have low spiking frequency. The hypothesis is that those low-frequency neurons have a small contribution to the frequency of spiking of the output neurons and thus removing them will not have a significantly negative effect on the accuracy. We perform this post-training pruning analysis on three applications for the digital neuromorphic architecture DANNA2 [6]. The applications we consider are the following: a pole-balancing control task (abbreviated as PB), a classification task on satellite radio signal data, and an Atari-like game Asteroids. Table I contains statistics from 100 spiking neural networks that have been trained via an evolutionary algorithm which optimizes only for performance. We can see that the average spiking frequency of internal neurons is much lower than the spiking frequency of the input and output neurons. This is an indication that we could prune a large number of internal neurons from a network without a significant negative effect on performance of the network.

TABLE I
STATISTICS FROM 100 SPIKING NEURAL NETWORKS FOR DIGITAL IMPLEMENTATION. THE AVERAGE SPIKING FREQUENCY OF MOST OF THE INTERNAL NEURONS IS AN ORDER OF MAGNITUDE LESS THAN THE SPIKING FREQUENCY OF INPUT AND OUTPUT NEURONS.

	PB	Radio	Asteroids
Avg. number of internal neurons	9.36	77.8	75.09
Avg. number of synapses	32.61	139.19	438.18
Avg. spiking freq. of internal neurons	0.0015	0.036	0.00016
Avg. spiking freq. of input neurons	0.033	0.269	0.00064
Avg. spiking freq. of output neurons	0.022	0.266	0.0013
Avg. performance	292.2 (sec)	0.777 (accuracy)	214.9 (score)

Thus, we prune the internal neurons whose spiking frequency is an order of magnitude less than the spiking frequency of the output neurons. Specifically, we prune neurons whose frequency f is such that $f < 0.1 * f_{avg}(\text{outputs})$, where $f_{avg}(\text{outputs})$ denotes the average spiking frequency of the output neurons. When performing such pruning, we get networks that have most of the internal neurons of the original network pruned, as shown in Table II. Though the number of internal neurons and the number of synapses is very significantly decreased, the loss of performance is comparatively small, as we can see from Table II.

TABLE II
STATISTICS FROM 100 SPIKING NEURAL NETWORKS FOR DIGITAL IMPLEMENTATION THAT WERE PRUNED BASED ON LOW FREQUENCY SPIKING INTERNAL NEURONS.

	PB	Radio	Asteroids
Avg. number of internal neurons	2.43	4.97	35.24
Avg. number of synapses	21.53	62.52	393.4
Avg. Performance	288.7 (sec)	0.778 (accuracy)	208.3 (score)

These results lead to the question of whether it is worth implementing a penalty for the size of the network during the training process, or whether post-training pruning performs better, i.e. leads to a smaller size and higher performance of the network. To answer this, we generate 100 networks from the same applications, but in this case, during the evolutionary optimization training of these networks, we penalize the fitness function for the number of neurons. In particular, the fitness function $F(N)$ for a network N that we consider is the following:

$$F(N) = \text{performance}(N) * \\ * \left(1 - \frac{\text{num. of hidden neurons}(N)}{\text{total number of neurons}(N)} * \delta\right),$$

where $\delta = 0.001$ has been experimentally chosen. Evolving networks in this manner we get the best results for both size and performance of the networks, as shown in Table III. We also note that the number of synapses in the resulting networks is also very low.

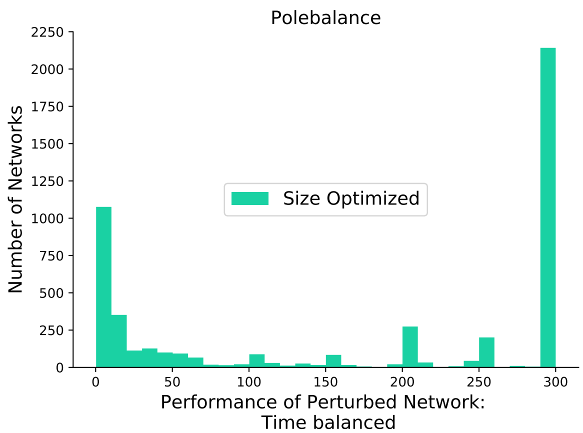
TABLE III
STATISTICS FROM 100 SPIKING NEURAL NETWORKS FOR DIGITAL IMPLEMENTATION THAT WERE EVOLVED WITH A MULTI-OBJECTIVE FITNESS THAT OPTIMIZES FOR PERFORMANCE AND FOR THE NUMBER OF NEURONS IN THE NETWORK.

	PB	Radio	Asteroids
Avg. number of internal neurons	5.08	0.56	1.3
Avg. number of synapses	23.41	122.9	147.17
Avg. performance	297.5 (sec)	0.788 (accuracy)	235.22 (score)

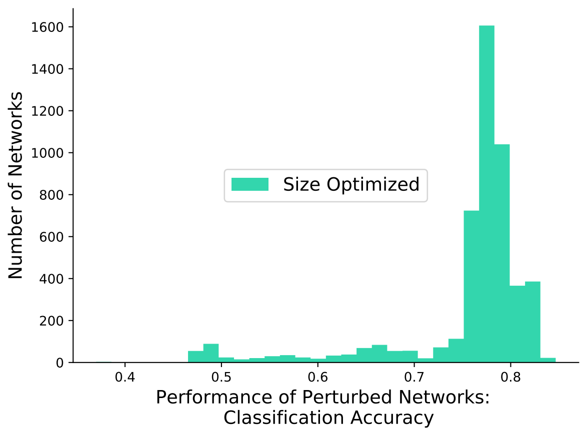
Furthermore, as we can see from Table III, using a multi-objective fitness function to generate smaller sized networks even improves the average performance of the networks. Hence, we keep the size-penalty in the fitness function in our further experiments and we choose to optimize networks for size during training instead of the post-training pruning. We note however that we observed longer training time with the multi-objective fitness function, thus if computing resources for training are limited, the post-training pruning approach is a good alternative for decreasing the size of a network.

B. Size and Resilience Optimization

Though optimizing for size and performance during training gave better results than pruning the network post-training, because the number of internal neurons is very low, the resulting size-optimized networks have many input to output synapses. We performed a simple perturbation experiment to test the resiliency to synaptic weight perturbations of these size-optimized networks. Namely, we randomly chose 10 networks out of those 100 size-optimized networks from each of the pole balance and the radio application networks. We perturbed those networks by randomly sampling $1, 2, 3, 4$ and 5 of its synapses for 100 times consecutively and flipping one bit in the weight of the sampled synapse. Thus, we get 5000 perturbed networks in total. We chose the 8^{th} bit from the sampled synapses' weights to be flipped. Having networks that are resilient to such a bit flip change or similar changes is especially important for sensors that are in an environment with high radioactivity, where, for example, bit flip resilience is crucial to the sensors reliability [29]. But the size and performance optimized networks are observed to be sensitive to such perturbations. Namely, in the experiment many of the resulted perturbed networks have their performance significantly affected, as shown in Figure 1. Namely, 35.3% of the perturbed networks for the pole balance task have balancing



(a)



(b)

Fig. 1. Perturbing 10 of the size-optimized networks by flipping their 8$^{\text{th}}$ bit. (a) Pole balance application: many of the synapses are very sensitive to this perturbation as more than a third of the resulting generated (perturbed) networks have their balancing time degrading to zero. (b) Radio application: though the SNNs for this application are less sensitive to perturbations, there is still a significant percentage of synapses whose perturbation leads to loss of accuracy.

time of less than 50 sec, while the optimal balancing time for this app is 300 sec. For the radio application, the evolutionary algorithm generates networks that are much more resilient to bit flips when compared to the SNNs for the pole balance application, but here too we have that almost 7% of the perturbed networks have accuracy less than 60%, while the un-perturbed size-optimized networks all had accuracy greater than 60%. Ultimately, we'd like the percentage of perturbed networks that lose their good performance to be as small as possible.

Given these observations we propose the following multi-objective fitness function $F(N)$ for spiking neural network N, that penalizes both for the number of neurons and for the sensitivity to perturbations:

$$
\begin{aligned}
F(N) = w_1 * \text{performance}(N) * \\
* (1 - \frac{\text{num. of hidden neurons}}{\text{total number of neurons}} * \delta) + \\
+ w_2 * (\frac{\sum_{i=1}^{n} \text{performance of variation}(N_i)}{n})
\end{aligned}
$$

where δ, n, w_1, w_2 are parameters that are to be chosen experimentally with the following constraints: $w_1, w_2 \in [0, 1]$ and $w_1 + w_2 = 1$; $\delta \in (0, 1)$; $n \geq 1$. By N_i we denote a variation of the network whose fitness is currently evaluated. The variation will depend on the type of perturbation that we are expecting to see in hardware. For example, if we are considering a digital implementation of a SNN, then a typical hardware fault is a bit flip. Then, one type of variation of the network that we can consider is a network that has one (or more) of its synapse weights changed to a weight that has one (or more) of its bits flipped.

C. Multi-objective Evolutionary Optimization Applications

We apply the multi-objective function that optimizes for size and resilience to applications from two types of neuromorphic implementations: one where the weights are integer values and one which admits floating point weights. Details about the digital implementation with integer weights (DANNA2) can be found in [6] and details about the floating-point weights implementation (NIDA) can be found in [28]. The main distinction of these two implementations for the purposes of this work, is that in the digital implementation the synaptic weights are integers in the range $[-1024, 1024]$ and in the NIDA implementation synapses admit floating point weights between -1 and 1. In all of our experiments, the parameters for the multi-objective fitness function stated in Equation III-B were experimentally chosen as $\delta = 0.001$ $n = 5$, and $w_1 = w_2 = 0.5$. For the Pole balance task, all the networks were trained until they achieved training balancing time of 300.02 seconds; for the radio task, all the networks were trained until they achieved accuracy of 77% or greater; for the asteroids app, all the networks were trained for 50 epochs. While for the resiliency-and-size optimized networks we achieve better fault tolerance, the training time was naturally longer than if we were optimizing networks for size and performance only. The experiments are further detailed below.

DANNA2 Pole Balance

In a digital implementation, the type of hardware fault that can be experienced is a bit flip. As flipping the 8^{th} bit of digitally implemented SNNs led to significant loss of performance in the performance-and-size only optimized networks, we considered evolving networks that are resilient to the flip of the 8^{th} bit. To this end, we considered variations in which every synapse has the 8^{th} bit flipped with probability 0.1. In summary, the SNN synaptic weight operations considered were the following:

- Perturbation: a sampled synapse has 8^{th} bit flipped.

- Variations: 5 variations were considered, and in each network variation, each synapse has its 8^{th} bit flipped with probability 0.1.

As we can see from Figure 2, evolving the networks with the resilience to perturbations taken into account in the fitness function leads to much more resilient networks (fewer sensitive synapses). In Figure 2(b), a Gaussian is fitted for both the resilience metric of the resiliency optimized and the resilience unoptimized networks. The resilience metric for each network N that we calculated is:

$$\frac{\text{optimal performance} - \text{network performance}}{\text{optimal performance}} \tag{1}$$

where for the pole balance task, optimal performance is 300.02 seconds. As seen in Figure 2(b), the resilience-optimized networks have a twice higher resiliency mean than the size-and-performance only optimized SNNs.

NIDA Polebalance

In the NIDA implementation, the type of hardware fault that can be experienced is a "diminishing weight". Namely, due to current and voltage drops, the weight value can diminish closer to zero. Thus, the synaptic weight operations considered were the following:

- Perturbation: a sampled synapse is diminished by $\epsilon \in (0.005, 0.05)$.
- For the multi-objective fitness function, 5 variations are considered for resilience. In each variation, every synapse is sampled with probability 0.5 and is diminished by 0.05.

The results, showing that the resilience optimized networks are indeed resilient to these perturbations, while the size-optimized networks are not, are shown in Figure 3.

NIDA Radio

The performance-and-size-optimized networks for the Radio signal classification application were more resilient to diminishing weight changes, as most of the networks perturbed in such a way still led to classification greater than 75%. However, the networks are less resilient to decrements by $\epsilon \in (0.008, 0.1)$. Thus, the synaptic weight operations considered were the following:

- A sampled synapse is decreased by $\epsilon \in (0.008, 0.1)$.
- For the multiobjective fitness function, 5 variations for resilience were considered. In each variation, every synapse is decreased by $\epsilon \in (0.008, 0.1)$, where ϵ is sampled uniformly from the interval $(0.008, 0.1)$.

The results are show in Figure 4. Again, the negative effect on performance in the resilience optimized perturbed networks is not present.

NIDA Asteroids

For the asteroid application, we perform a perturbation that diminishes the synaptic weights by $\epsilon \in (0.001, 0.1)$. In this application we have three metrics: the time the player stayed alive, its score, and the total shooting points the player has acquired. We chose the score as a comparison metric

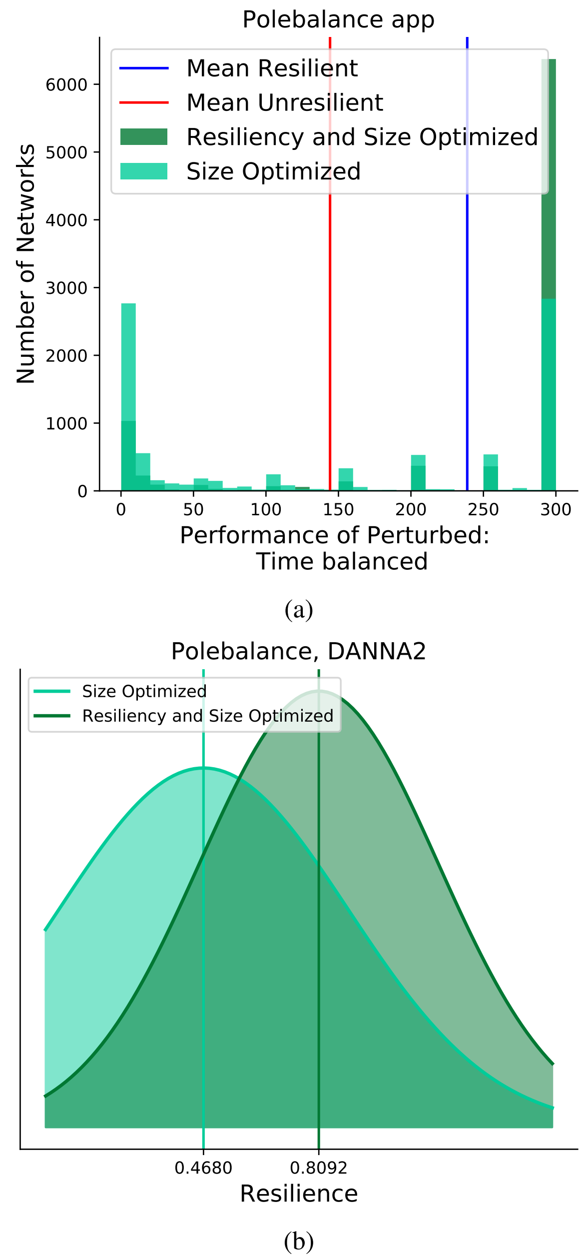


Fig. 2. Perturbing 20 size-optimized and 20 size-and-resilience optimized networks by flipping their 8th bit: we sample 1 through 5 synapses for 100 times for each of those networks and flip the 8th bit of the sampled synapses. (a) Histogram of the experiment. (b) Fitted Gaussians from resiliency scores.

between the networks that were optimized for size only and the networks optimized for size and resilience. We have trained 10 networks for 50 epochs in both scenarios. The experiments involved the following synaptic operations:

- Perturbation: a sampled synapse is diminished by $\epsilon \in (0.001, 0.1)$.
- Variations: in the multiobjective fitness, 5 variations are considered. For every synapse in each variation, we diminish its weight by $\epsilon = 0.1$ with probability 0.5.

The results are show in Figure 5. Again, the resiliency opti-

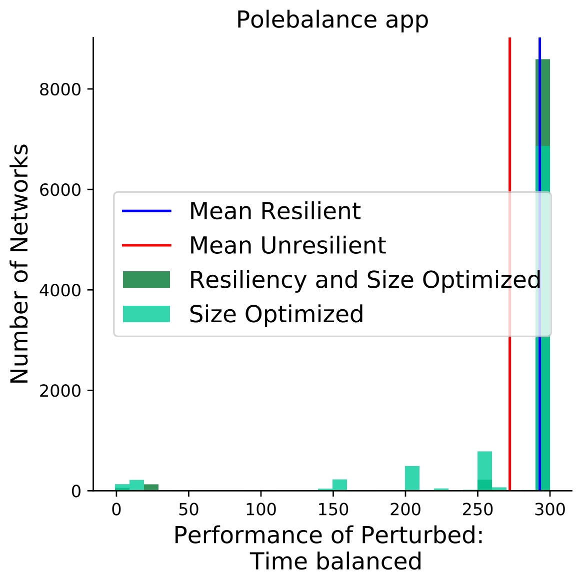


Fig. 3. Perturbing 20 of the size-optimized and 20 size-and-resilience optimized networks by diminishing them for $\epsilon \in (0.005, 0.05)$: we sample 1 through 5 synapses for 100 times for each of those networks and diminish the sampled synapses. The size-and-resilience optimized networks have higher mean (almost all of them balance at around 300 sec., which is not the case for the size-only optimized networks).

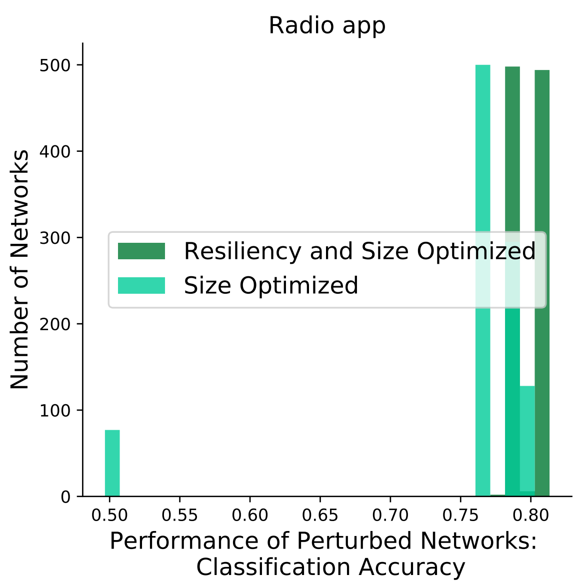
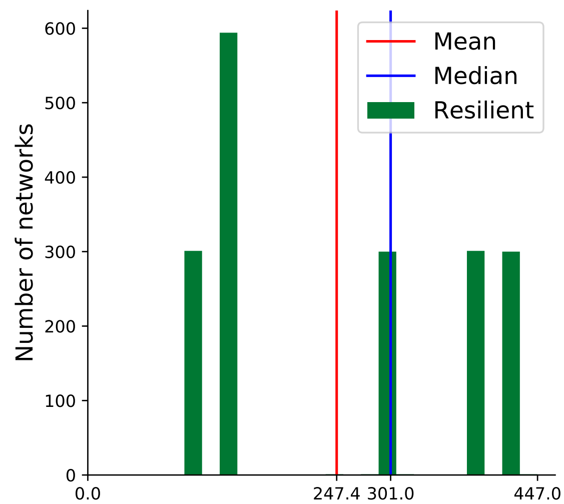
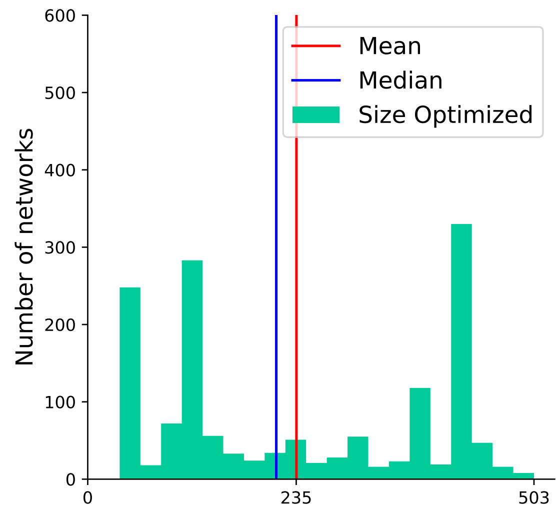


Fig. 4. Perturbing 5 of the size-optimized and size-and-resilience optimized networks by decreasing them for $\epsilon \in (0.008, 0.1)$: we sample 1 through 5 synapses for 100 times for each of those networks and diminish the sampled synapses. The size-and-resilience optimized networks all retain their accuracy when perturbed (they perform at around $75 - 80\%$ classification accuracy), while a portion of the perturbed size and performance-only optimized networks has accuracy around 50%.



(a)



(b)

Fig. 5. Perturbing 10 of the size-optimized and size-and-resilience optimized networks by diminishing them by $\epsilon \in (0.001, 0.1)$: we sample 1 through 5 synapses for 100 times for each of those networks and diminish the sampled synapses. The size-and-resilience optimized networks show better performance when perturbed: they have higher mean and median for the score of perturbed networks. (a) Performance of perturbed networks when networks were evolved with performance, size and perturbation resilience optimization. (b) Performance of perturbed networks when networks were trained with performance and size optimization.

mized networks show higher mean and median in performance when perturbed than the size and performance optimized SNNs.

IV. Conclusion and Discussion

In this work we propose a multi-objective fitness optimization function for training Spiking Neural Networks via an evolutionary algorithm. This function optimizes for performance, size and perturbation resilience of the network. The factor that accounts for perturbation resilience involves several variations of the network whose fitness is evaluated and takes into account their performance. The network variations that we consider depend on the type of perturbation we can encounter due to hardware faults. Another contribution of this article is an empirical comparison of two size-reduction methods, where we show that in-training size constraints lead to better results in terms of network size and accuracy as opposed to pruning networks post-training. In summary, this work shows that algorithmic, software-side solutions to producing spiking neural networks which are resilient to hardware faults and satisfy hardware constraints are possible. As future work, it would be interesting to consider finding good initialization techniques for the seed networks in the evolutionary algorithm as well as performing a hyperparameter analysis of the parameters in the multi-objective fitness function. This would allow for a faster convergence to well-performing and resilient networks, trained by utilizing the multi-objective fitness function.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Robinson Pino, program manager, under contract number DE-AC05-00OR22725, and in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory.

References

[1] X. Wang, H. Qiu, and F. Xie, "A survey on the industrial readiness for internet of things," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE, 2017, pp. 591–596.

[2] F. Samie, L. Bauer, and J. Henkel, "Iot technologies for embedded computing: A survey," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2016, p. 8.

[3] S. Chatterjee, S. Chatterjee, S. Choudhury, S. Basak, S. Dey, S. Sain, K. S. Ghosal, N. Dalmia, and S. Sircar, "Internet of things and body area network-an integrated future," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE, 2017, pp. 396–400.

[4] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[5] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[6] 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.

[7] 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.

[8] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117–1125.

[9] 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.

[10] A. P. Johnson, J. Liu, A. G. Millard, S. Karim, A. M. Tyrrell, J. Harkin, J. Timmis, L. J. McDaid, and D. M. Halliday, "Homeostatic fault tolerance in spiking neural networks: A dynamic hardware perspective," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 687–699, 2017.

[11] T. C. Südhof and R. C. Malenka, "Understanding synapses: past, present, and future," *Neuron*, vol. 60, no. 3, pp. 469–476, 2008.

[12] S. Hammoudi, Z. Aliouat, and S. Harous, "Challenges and research directions for internet of things," *Telecommunication Systems*, vol. 67, no. 2, pp. 367–385, 2018.

[13] 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.

[14] 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.

[15] J. J. Reynolds, J. S. Plank, C. D. Schuman, G. R. Bruer, A. W. Disney, M. E. Dean, and G. S. Rose, "A comparison of neuromorphic classification tasks," in *Proceedings of the International Conference on Neuromorphic Systems*. ACM, 2018, p. 12.

[16] C. D. Schuman, J. S. Najem, R. Weiss, N. Skuda, A. Belianinov, P. Collier, S. Sarles, G. Rose *et al.*, "Biomimetic, soft-material synapse for neuromorphic computing: from device to network," Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep., 2019.

[17] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[18] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[19] J. Chung and T. Shin, "Simplifying deep neural networks for neuromorphic architectures," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[20] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[21] M. Dimovska and T. Johnston, "A novel pruning method for convolutional neural networks based off identifying critical filters," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*. ACM, 2019, p. 63.

[22] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[23] Y. Ji, Y. Zhang, S. Li, P. Chi, C. Jiang, P. Qu, Y. Xie, and W. Chen, "Neutrams: Neural network transformation and co-design under neuromorphic hardware constraints," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Press, 2016, p. 21.

[24] J. Lin, Z. Zhu, Y. Wang, and Y. Xie, "Learning the sparsity for reram: mapping and pruning sparse neural network for reram based accelerator," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 2019, pp. 639–644.

[25] D. S. Yeung, J.-C. Li, W. W. Ng, and P. P. Chan, "Mlpnn training via a multiobjective optimization of training error and stochastic sensitivity," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 5, pp. 978–992, 2015.

[26] S. Yuan, G. Deng, Q. Feng, P. Zheng, and T. Song, "Multi-objective evolutionary algorithm based on decomposition for energy-aware scheduling in heterogeneous computing systems." *J. UCS*, vol. 23, no. 7, pp. 636–651, 2017.

[27] W. Olin-Ammentorp, K. Beckmann, C. D. Schuman, J. S. Plank, and N. C. Cady, "Stochasticity and robustness in spiking neural networks," *arXiv preprint arXiv:1906.02796*, 2019.

[28] C. D. Schuman, J. D. Birdwell, and M. Dean, "Neuroscience-inspired inspired dynamic architectures," in *Proceedings of the 2014 Biomedical Sciences and Engineering Conference*. IEEE, 2014, pp. 1–4.

[29] J. Elliott, M. Hoemmen, and F. Mueller, "Exploiting data representation for fault tolerance," *Journal of computational science*, vol. 14, pp. 51–60, 2016.