Training Spiking Neural Networks Using Combined Learning Approaches

Daniel Elbrecht
Department of EECS
University of Central Florida
Orlando, Florida, USA
delbrecht1@knights.ucf.edu

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

Shruti R. Kulkarni
Computer Science and Mathematics
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
kulkarnisr@ornl.gov

J. Parker Mitchell
Computer Science and Mathematics
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
mitchelljp1@ornl.gov

Catherine D. Schuman
Computer Science and Mathematics
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
schumancd@ornl.gov

***Abstract*—Spiking neural networks (SNNs), the class of neural networks used in neuromorphic computing, are difficult to train using traditional back-propagation techniques. Spike timing-dependent plasticity (STDP) is a biologically inspired learning mechanism that can be used to train SNNs. Evolutionary algorithms have also been demonstrated as a method for training SNNs. In this work, we explore the relationship between these two training methodologies. We evaluate STDP and evolutionary optimization as standalone methods for training networks, and also evaluate a combined approach where STDP weight updates are applied within an evolutionary algorithm. We also apply Bayesian hyperparameter optimization as a meta learner for each of the algorithms. We find that STDP by itself is not an ideal learning rule for randomly connected networks, while the inclusion of STDP within an evolutionary algorithm leads to similar performance, with a few interesting differences. This study suggests future work in understanding the relationship between network topology and learning rules.**

***Index Terms*—spiking neural networks, spike-timing dependent plasticity, evolutionary algorithms, Bayesian optimization**

I. Introduction

With the looming end of Moore's law and the rise of big data, neuromorphic computing systems offer a compelling path forward for low power, efficient machine learning. Neuromorphic systems, especially those based on spiking neural networks (SNNs), offer the ability to perform more neuroscience-inspired machine learning. Many neuromorphic systems, including Intel's Loihi [1], include on-chip learning mechanisms in the form of synaptic plasticity. There is tremendous promise for utilizing these synaptic plasticity mechanisms for both supervised and unsupervised learning. In addition, on-chip plasticity mechanisms may have also improved noise and fault tolerance, but these features have not been fully explored in the literature.

One of the most common learning mechanisms implemented in neuromorphic hardware is spike-timing-dependent plasticity (STDP) [2]. Multiple types of STDP have been explored in the literature, including both unsupervised and supervised approaches. There has been some success with utilizing STDP as a standalone learning mechanism, especially on tasks such as image classification, but it is not clear how extensible it is to new applications or how much hand-tuning is required for STDP to work for a given application. For example, it is not clear whether STDP will work "out-of-the-box" on new applications, or whether significant tuning of the network itself and other relevant hyperparameters will be required. That is, in addition to defining hyperparameters of the STDP process itself, there are also other aspects that must be defined in order to utilize STDP, including the structure of network (numbers of neurons and connectivity) and the input encoding approach utilized for the network. As such, several questions emerge: how useful is STDP for new tasks? Can it be easily applied, or is significant tuning of network design and hyperparameters required for it to be useful? Moreover, what benefits does it provide over other learning approaches?

In this work, we seek to address some of these questions for a particular supervised STDP learning rule. We present approaches utilizing evolutionary algorithms and Bayesian optimization for auto-tuning the SNN structure and hyperparameters for supervised STDP-trained networks. We compare the performance of this supervised STDP process as a standalone learning procedure (with randomly generated network structures) with the performance of these hyperparameter and network design algorithms operating on top of the supervised

STDP learning approach. We also compare the performance of these approaches without the STDP learning rule and determine the effect of the inclusion of STDP as part of the learning process on training and testing accuracy, network size and structure, and network resiliency. We show that STDP as a standalone learning process (even with hyperparameter optimization) is significantly outperformed by the evolutionary approach. However, we also show that the inclusion of STDP with the evolutionary optimization can improve the generalization ability of the resulting networks. We also show that networks produced by the evolutionary and STDP process tend to produce smaller networks than those produced by the evolutionary process alone. Finally, we examine the resiliency characteristics of these different training algorithms with respect to synaptic failures.

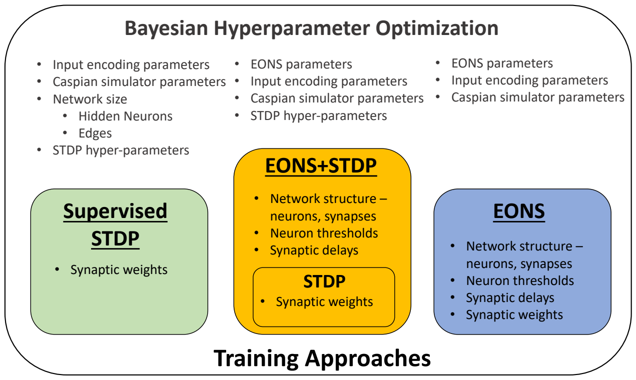
II. Background and Related Work

A. Plasticity-based learning

Spike timing-dependent plasticity (STDP) is a biologically inspired learning rule for synaptic weight updates in SNNs [3]. This learning rule is based on observed phenomena of synaptic strengthening and weakening in brains. While many variations of STDP exist, the underlying rule is that a synapse between pre and post-synaptic neurons will be strengthened if firing at the pre-synaptic neuron caused firing at the post-synaptic neuron. Conversely, synapses are weakened if firing at the pre-synaptic neuron does not induce firing at the post-synaptic neuron. STDP can either be supervised or unsupervised. In the original, unsupervised form, the weight updates are the same regardless of the correctness of the network output. However, STDP can be made to be ”supervised” in a variety of ways. In this work, we utilize reward modulated STDP as the supervised STDP learning approach. In this case, the direction of the weight update (strengthened or weakened) is dependent on the network output [4]. Firing patterns which result in correct predictions will be strengthened, while those that result in incorrect predictions are weakened. STDP is not the only biologically inspired learning rule; others such as spike driven synaptic plasticity (SDSP) base weight changes on a combination of spike times and neuron voltages [5].

B. SNN Training and Architecture Search

Supervised training in SNNs is also done in a variety of other ways, including gradient descent or backpropagation-style approaches [6]–[9], which are adapted in some way to accommodate for the spiking, non-differentiability of neurons in the SNN. Many of these approaches do not take advantage of the full capability of SNNs, particularly in the training and use of synaptic or axonal delays in SNNs. Moreover, with each of these approaches, selecting the appropriate network architecture or hyperparameters is still an issue. Evolutionary optimization-based approaches can overcome some of these issues by designing all aspects of the SNNs, including parameters such as synaptic or axonal delay, as well as network structure, such as number of neurons and synapses [10]–[13]. However, evolutionary approaches also include algorithmic



Bayesian Hyperparameter Optimization
Input encoding parameters
Caspian simulator parameters
Network size
Hidden Neurons
Edges
STDP hyper-parameters
EONS parameters
Input encoding parameters
Caspian simulator parameters
STDP hyper-parameters
EONS parameters
Input encoding parameters
Caspian simulator parameters
Supervised STDP
Synaptic weights
EONS+STDP
Network structure – neurons, synapses
Neuron thresholds
Synaptic delays
STDP
Synaptic weights
EONS
Network structure – neurons, synapses
Neuron thresholds
Synaptic delays
Synaptic weights
Training Approaches

Fig. 1. We compare three approaches in this work: STDP only, evolutionary optimization (EONS) only, and STDP and EONS together. For each of these approaches, we utilize a Bayesian hyperparameter optimization approach to tune the hyperparameters. This figure shows the relationship between the three approaches.

hyperparameters that, once optimized, could provide better performance.

One common approach for hyperparameter optimization in the deep learning literature, Bayesian optimization, has also been applied to SNNs. It has been successfully applied to optimize hyperparameters of backpropagation-style algorithms, binary networks [14], [15] and liquid state machines [16], as well as in neuromorphic computing systems with algorithmic hyperparameters for the evolutionary optimization process, input/output encoding, as well as hardware hyperparameters [17], [18].

III. Approach

In this work, we combine STDP with evolutionary optimization-based neural architecture search and parameter optimization, along with Bayesian optimization for algorithmic hyperparameter optimization. We compare this combined approach with a standalone STDP and a standalone evolutionary optimization approach, both of which have also undergone Bayesian hyperparameter optimization. These three approaches are shown in Figure 1, which emphasizes how they are nested together to perform different optimizations. We describe each of the components of this approach in the following subsections.

A. Spiking Neural Network Model

We use a leaky integrate and fire neuron model, with simple weighted synapses and discrete time delays on the synapses. We utilize a recurrent network architecture. In this recurrent architecture, any neuron may be connected to any other neuron in the network, including input and output neurons. We specify the number of input neurons, hidden neurons, output neurons, and the number of synapses to utilize and then construct the network by initializing the neurons and then randomly selecting two neurons to connect with a synapse until the desired number of synapses is reached. All SNNs are implemented and evaluated using the Caspian neuromorphic processor [19], an FPGA-based neuromorphic implementation with a hardware accurate simulator written in C++.

In this work, we utilize datasets in which the input values are floating point or integer values that must be encoded as spikes. Input encoding (from value to spike) is either done through a combination of binning and rate-based encoding, which are described in more detail in [20], or using temporal coding, in which the time of the spike encodes the value. We focus in this work on classification tasks with discrete outputs; as such, each output class is represented by an output neuron and we decode the spikes on those neurons to find the appropriate classification value. Discrete output decoding is done by selecting the neuron which has fired the most over the time (a winner-take-all approach) or temporal coding, in which the output neuron that spikes first is selected as the output class. Determining the appropriate encoding/decoding scheme and their associated hyperparameters are an important selection when utilizing any SNN approach and can have a significant impact on the performance of the algorithm. As such, we utilize hyperparameter optimization (described below) to find the best performing encoding/decoding approaches for each algorithm.

B. STDP learning rule

We utilize a supervised spike timing-dependent plasticity rule which updates all synapses within an SNN, based on the discrete firing times of the neurons, similar to the rule demonstrated here [4]. For synapse weight w, our update rule is as follows:

$$\Delta w = \frac{\alpha * sgn(w)}{t_{post} - t_{pre} + 0.5}$$

where t_{pre} is the time the pre-synaptic spike arrives at the post-synaptic neuron, t_{post} is the spike time of the post-synaptic neuron, α is the learning rate, and sgn(w) is the sign, positive or negative, of the current weight value. In order to perform supervised STDP, we use one two α values, α_r(reward) and α_p(punish), based on whether the network classification is correct. α_r is always positive, to drive the weight update in the normal direction, while α_p is always negative, to cause an anti-STDP weight update. Since the firing times for neurons in the Caspian processor are all discrete integers, we include the 0.5 constant in the denominator to prevent divide by zero when $t_{pre} = t_{post}$, while also ensuring symmetry of the magnitude of weight updates when $t_{pre} > t_{post}$ or $t_{pre} < t_{post}$. This update occurs for each pair of pre and post synaptic spikes which fall within a defined window. Weight updates are computed and applied after each training example is evaluated by the spiking neural network.

C. EONS

One of the key issues with utilizing a learning procedure such as STDP is how to select the network structure and other network parameters such as delays (in our case, on the synapses) or neuron thresholds. The network structure as well as the other network parameters are often hand-tuned by the researcher for a given task. To automate this process, we utilize Evolutionary Optimization for Neuromorphic Systems (EONS) [12], [13] to automatically design the network structure and network parameters of synaptic delay and neuron thresholds. EONS can be utilized as a standalone network design process that tunes all aspects of the network, or it can be used in combination with other training or learning approaches. In this work, we utilize EONS both as a standalone approach for determining all aspects of the network (structure and parameters), as well as in combination with STDP (where EONS defines the structure and non-weight parameters).

EONS is a genetic or evolutionary algorithm-based approach for designing SNNs for neuromorphic deployment. EONS utilizes a direct representation of the SNN in its population, so every aspect of the network is directly represented in the individual stored in the population. In the case of the EONS, the representation is the network itself (not a traditional genomic representation). During the EONS process, each network in the population is evaluated using a fitness function and assigned a score. The scores are used to perform selection, wherein better performing networks are preferentially selected to serve as parents for the next generation. Reproduction operations are performed on this pool of parent networks to produce child networks for the next generation. EONS utilizes both crossover (on the networks themselves), as well as mutations (e.g., adding or deleting a neuron, adding or deleting a synapse, changing a parameter value, etc.) to produce child networks. EONS can be easily combined with other training or learning mechanisms. In this case, we combine EONS with the STDP learning approach by including the STDP learning as part of the fitness evaluation process.

D. Bayesian Hyperparameter Optimization

We utilize iterative Bayesian-based technique to perform hyperparameter optimization for our algorithms (EONS and STDP), as well as hyperparameters associated with the Caspian hardware (such as parameter value ranges) and hyperparameters associated with how input encoding from values to spikes works. Bayesian optimization is a technique suitable for optimizing black-box and expensive objective functions. It builds a posterior distribution using a likelihood model (observations) and Gaussian processes as a *priori*. A surrogate model is optimized and directs the search path toward estimating the unknown black-box function (i.e., the accuracy of the network) while determining the best next point (set of hyperparameters) to evaluate at each step. In each iteration, the surrogate model is calculated to ensure both exploration and exploitation of the search space.

In this work, we optimize hyperparameters for each combination of algorithms shown in Figure 1 (i.e., Bayesian optimization is used as the outer optimization approach for EONS alone, STDP alone, and EONS+STDP approaches). This step is performed to determine the optimal hyperparameters for each approach and to perform fair comparison between them (rather than tailoring the hyperparameters for one approach and using that on all approaches). For each of these individual

TABLE I
OPTIMIZED HYPERPARAMETERS

Parameter category	Parameter	EONS	STDP	EONS+STDP
EONS	population size	800	1000	1100
	crossover rate	0.5	-	0.7
	mutation rate	0.7	-	0.5
Encoding	spike count	12	10	10
	interval	7	6	7
	bins	8	-	8
	time to first spike	-	True	-
Caspian	max synapse weight	31	63	63
	max synapse delay	15	3	15
	max neuron threshold	15	31	15
STDP	window	-	5	12
	α_r	-	0.6	0.5
	α_p	-	0.6	0.5
Random network generation	hidden neurons	-	20	-
	synapses	-	140	-

algorithmic approaches, the Bayesian optimization approach begins by evaluating two sets of hyperparameters. The results from those two observations are used to build the initial posterior Gaussian process. In each iteration of Bayesian search, a surrogate model (called the acquisition function) is built based on the Gaussian distribution. The optimum point of the acquisition function is the set of hyperparameters to evaluate in the next iteration. In each iteration, the added observations reduce the uncertainties of the prior Gaussian distribution and helps predicting the actual trend for the performance of the network. The process is repeated at least for 50 iterations for each algorithmic approach in order to determine well-performing hyperparameters for each approach.

IV. RESULTS

We focused our evaluation on the wine dataset from scikit-learn, which has 13 input attributes and 3 output classes. This is a small dataset that is easily evaluated and allows for large-scale evaluation of these different algorithmic approaches. In the future, we intend to investigate the effect on other datasets.

A. Hyperparameter Optimization Results

As noted in Section III, we utilize Bayesian optimization to optimize the hyperparameters for each approach. The full set of hyperparameters optimized is shown in Table I, where hyperparameters that are not utilized for that algorithmic approach are left as dashes. Note that there are five types of hyperparameters:

- EONS: Hyperparameters for the evolutionary or genetic approach. These are the population size, the crossover rate, and the mutation rate.
- Spike encoding: Hyperparameters associated with how input values are encoded into spikes.
- Caspian: Hyperparameters associated with the neuromorphic hardware utilized. These include maximum values of synaptic weight, synaptic delay, and neuron threshold, each of which are related to the precision of the value. The maximum weight value for Caspian also determines the minimum weight value, which is the negation of the maximum weight value.
- STDP: Hyperparameters that are used to govern how the supervised STDP process behaves.
- Random network generation: Hyperparameters that determine the size of the initial randomized networks.

Table I also gives the best values found using the Bayesian optimization process for each of the three algorithmic approaches, EONS, STDP, and EONS+STDP. There are a few interesting things to note about these values. First, for both algorithmic approaches that utilize STDP, higher maximum weight values, allowing for a weight range of -63 to 63, as compared with the EONS approach, which suffices with a weight range of -31 to 31. Because the particular hardware implementation utilized here (Caspian) has integer weight values, we speculate that the wider weight range is required to be able to recognize and utilize the STDP updates. If a smaller weight range were imposed, there would be less room for STDP weight updates to make a difference in performance.

Also interesting to note is that the encoding approach utilized for both EONS and EONS+STDP are very similar, while the STDP approach alone uses a very different encoding approach with "time-to-first-spike" encoding. This could indicate (along with the results below), that the optimization process has automatically tuned the hyperparameters for EONS+STDP more for EONS than the embedded STDP process.

B. Training and Testing Performance

Utilizing the best performing hyperparameters for each algorithm (as listed in Table I), we then ran each of the algorithms for 50 training epochs. In the case of the EONS approaches, each epoch is one generation, while for the STDP only approach, each epoch is one pass through the data for each network in the "population." Figure 2 shows the average best accuracy over the course of the training epochs for each of these approaches. As we can see in these plots, the STDP approach does not appear to improve much over the course of training and is not able to pull itself out of the local optima in which only two of the three labels are classified correctly in the wine dataset. The initial networks for the STDP only approach perform better than the initial networks for the EONS

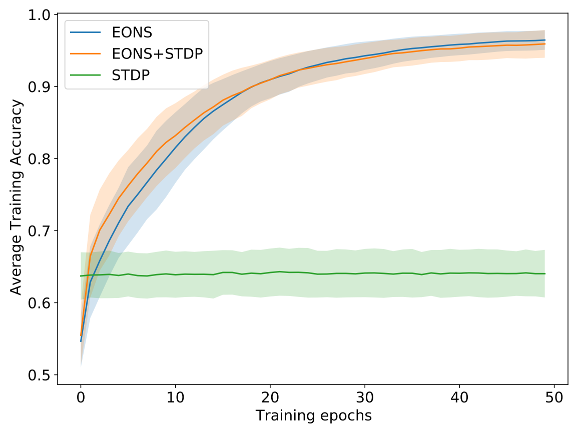


Fig. 2. Average training trajectories for each of the studied algorithms. Shaded areas indicate standard deviation.

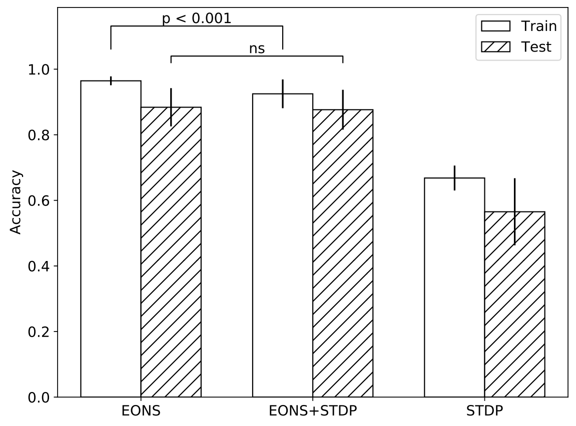


Fig. 3. Train and test accuracy for various algorithms with optimized hyperparameters. Error bars on each indicate the variation in performance. Statistical significance calculated using independent t-test.

and EONS+STDP approaches, but are quickly outperformed. Both the EONS and EONS+STDP training trajectories are very similar, indicating that the inclusion of STDP has little effect on the training performance.

Figure 3 shows the training and testing accuracy for each of the three algorithmic approaches. Once again, it is clear that the STDP approach alone performs significantly worse than the algorithmic approaches that utilize evolutionary optimization. However, the difference in performance between EONS and EONS+STDP is worth noting. In particular, there is a statistically significant performance different between the resulting training accuracies of EONS and EONS+STDP, with EONS outperforming the EONS+STDP result in terms of training accuracy. However, the difference in testing performance between EONS and EONS+STDP is *not* significant. As such, this indicates that the inclusion of STDP can improve the generalization ability of the network. With this result, it may be that EONS+STDP networks can take longer (require more training epochs) to achieve the same training accuracy, but that the resulting networks perform better on new data instances.

C. Network Structure Analysis

To understand the difference in performance between networks generated with the three different algorithmic approaches, we first looked at the pruned networks (networks in which all neurons and synapses that would not lead directly to an output are eliminated). The best performing networks for EONS, EONS+STDP, and STDP (where the number of hidden neurons and synapses is determined by the Bayesian optimization approach) in Figures 4, 5, and 6, respectively. As we can see in these figures, the networks produced by the evolutionary process are significantly different than the random structure initialized based on the optimized hyperparameters. Moreover, all of these network structures are significantly different than the traditional feed-forward networks that are often seen in the literature, even for SNNs. In future work, we intend to investigate the performance of these algorithms with more structured, feed-forward style networks.

Another interesting thing to note about the networks produced by the EONS approaches is that the resulting SNNs do not include any hidden neurons. Because EONS allows for connections between any two neurons in the network, including between inputs and inputs, outputs to outputs, and outputs to inputs, EONS will often train networks that do not require hidden neurons at all to perform classifications or control tasks. Instead, the inputs and outputs themselves will be used for computation as well.

Finally, Figure 7 shows the overall network size differences between networks produced by EONS and EONS+STDP, in terms of the number of neurons (nodes) and number of synapses (edges). As can be seen in this figure, the networks produced by EONS+STDP tend to be smaller than those produced by EONS alone, and that the difference in the network size is statistically significant. This is an interesting result that indicates that beyond improving the generalization ability of the SNNs produced, EONS+STDP networks can also be smaller, and thus are more likely to be more area and energy efficient in a neuromorphic implementation.

D. Resiliency

We examined the resiliency of the networks produced by each of the algorithms. To measure resiliency of a network, we randomly remove some fraction f of synapses in the network, and measure the corresponding change in training accuracy. This process is repeated multiple times for an individual network to obtain and average resiliency to random deletions for that network. The result of the resilience analysis are shown in Figure 8. EONS and EONS+STDP demonstrate very similar resilience curves, with a linear decrease in performance as a function of the fraction of synapses deleted. The STDP trained networks, while starting at a lower accuracy, appear to be

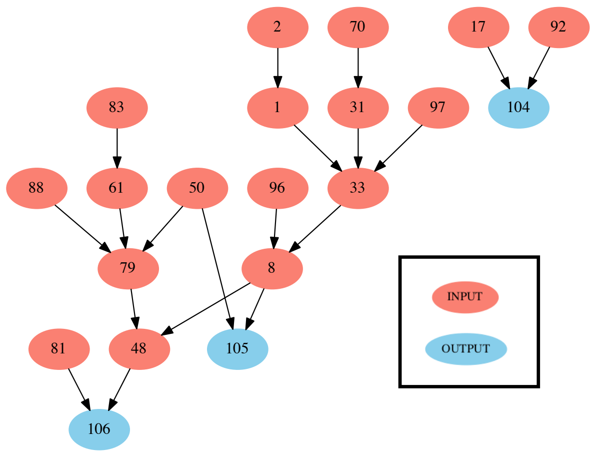


Fig. 4. Best performing network trained by EONS. Numbers correspond to neuron IDs assigned during evolution.

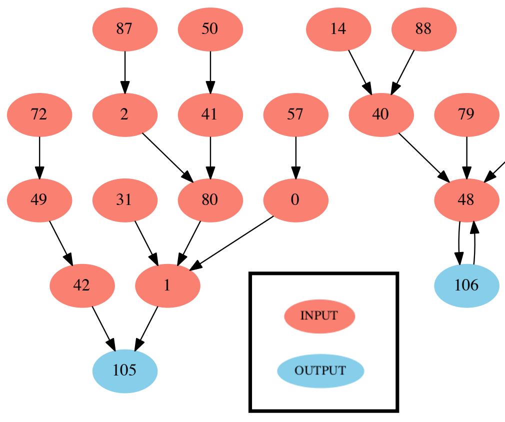


Fig. 5. Best performing network trained by EONS+STDP.

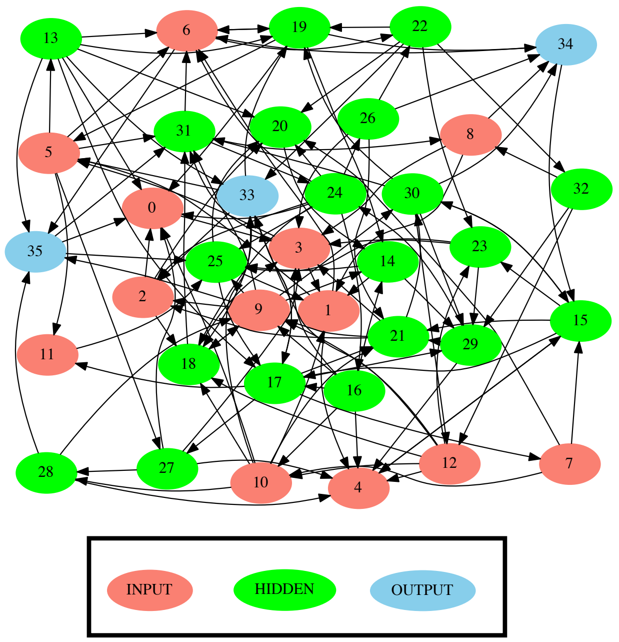


Fig. 6. Best performing network trained by STDP (randomly generated structure).

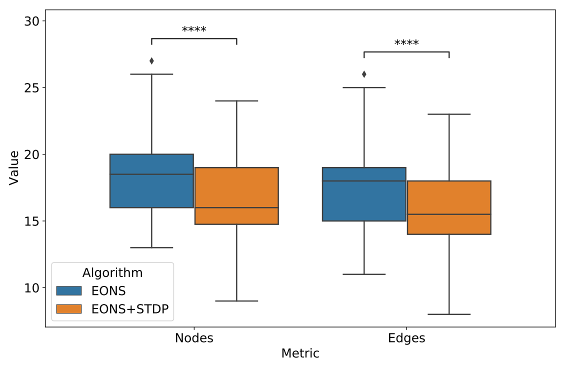


Fig. 7. Average number of nodes and edges in the networks produced by the EONS and EONS+STDP algorithms. Statistical significance calculated using independent t-test. **** indicates $p < 0.0001$.

more resilient to synapse deletions, as the slope of the STDP resilience curve is more gradual.

V. Discussion and Conclusions

Here we hierarchically combine three different types of learning or optimization approaches for spiking neural networks: STDP, EONS (an evolutionary optimization based approach), and Bayesian hyperparameter optimization. In this work, we focus this workflow specifically on understanding whether the evolutionary approach and the Bayesian hyperparameter optimization approach can be used to automate some of the hyperparameter and network design decisions that need to be made when utilizing an algorithm for a new application and determine whether they can improve the performance over "out-of-the-box" network designs and hyperparameter settings.

It is worth noting that we explicitly do not utilize pre-constructed or hand-tuned networks or hyperparameter sets within this work, because we do not believe that is reasonable to expect when applying an algorithm to each new application. However, based on other works in STDP literature, it is clear that with hand-tuning of both the network structure and the hyperparameters, STDP can achieve good results on some applications.

From this work, we can see that STDP is not necessarily an effective learning rule for the types of networks produced through evolutionary optimization, which may contain all types of recurrent connections and arbitrary structures. While we observe that the inclusion of STDP within the evolutionary optimization did not significantly impact performance across the metrics studied in this paper, it did have several noticeable effects, namely in reducing network size and improving gen-

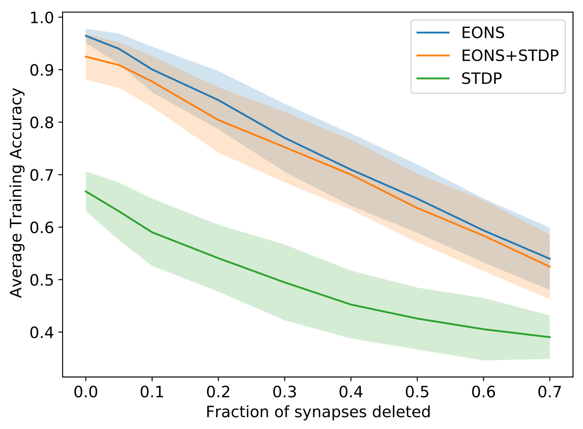


Fig. 8. Resilience of spiking neural networks produced by the studied algorithms

eralization ability. Both of these are desirable characteristics in deployed SNNs.

Here we focus on a particular supervised STDP learning rule. In future work, we intend to investigate the effect of the inclusion of other supervised and unsupervised learning rules with these additional optimization methods. We also intend to investigate the effect of different types of network structures, including more traditional feed-forward neural network structures.

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, and in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC.

This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We would like to thank Chris Layton for his support in our utilization of CADES Cloud.

References

[1] 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.

[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] Y. Dan and M.-m. Poo, "Spike timing-dependent plasticity of neural circuits," *Neuron*, vol. 44, no. 1, pp. 23–30, 2004.

[4] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated stdp," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6178–6190, 2018.

[5] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural computation*, vol. 19, no. 11, pp. 2881–2912, 2007.

[6] S. M. Bohte, J. N. Kok, and J. A. La Poutré, "Spikeprop: backpropagation for networks of spiking neurons." in *ESANN*, vol. 48, 2000, pp. 17–37.

[7] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

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

[9] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.

[10] N. Pavlidis, O. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. Vrahatis, "Spiking neural network training using evolutionary algorithms," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. IEEE, 2005, pp. 2190–2194.

[11] R. Batllori, C. B. Laramee, W. Land, and J. D. Schaffer, "Evolving spiking neural networks for robot control," *Procedia Computer Science*, vol. 6, pp. 329–334, 2011.

[12] 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.

[13] C. D. Schuman, J. P. Mitchell, R. M. Patton, T. E. Potok, and J. S. Plank, "Evolutionary optimization for neuromorphic systems," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–9.

[14] M. Parsa, C. D. Schuman, D. C. Rose, B. Kay, J. P. Mitchell, S. R. Young, R. Dellana, W. Severa, T. E. Potok, K. Roy *et al.*, "Hyperparameter optimization in binary communication networks for neuromorphic deployment," *arXiv preprint arXiv:2005.04171*, 2020.

[15] M. Parsa, A. Ankit, A. Ziabari, and K. Roy, "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, 2019, pp. 1–8.

[16] A. Chernyshev, "Bayesian optimization of spiking neural network parameters to solving the time series classification task," in *Biologically Inspired Cognitive Architectures (BICA) for Young Scientists*. Springer, 2016, pp. 39–45.

[17] M. Parsa, J. P. Mitchell, C. D. Schuman, R. M. Patton, T. E. Potok, and K. Roy, "Bayesian-based hyperparameter optimization for spiking neuromorphic systems," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 4472–4478.

[18] ——, "Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design," *Frontiers in Neuroscience*, vol. 14, p. 667, 2020.

[19] J. P. Mitchell, C. D. Schuman, R. M. Patton, and T. E. Potok, "Caspian: A neuromorphic development platform," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–6.

[20] C. D. Schuman, J. S. Plank, G. Bruer, and J. Anantharaj, "Non-traditional input encoding schemes for spiking neuromorphic systems," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–10.