To Spike or Not to Spike, that is the Question

1st Sanaz Mahmoodi Takaghaj*
Department of Computer Science and Engineering, and
School of Engineering Design and Innovation
Penn State University
University Park, PA USA
sxm788@psu.edu
* Author to whom any correspondence should be addressed.

2nd Jack Sampson
Department of Computer Science and Engineering
Penn State University
University Park, PA USA
jms1257@psu.edu

***Abstract*—Neuromorphic computing has recently gained momentum with the emergence of various neuromorphic processors. As the field advances, there is an increasing focus on developing training methods that can effectively leverage the unique properties of spiking neural networks (SNNs). SNNs emulate the temporal dynamics of biological neurons, making them particularly well-suited for real-time, event-driven processing. To fully harness the potential of SNNs across different neuromorphic platforms, effective training methodologies are essential. In SNNs, learning rules are based on neurons' spiking behavior, that is, if and when spikes are generated due to a neuron's membrane potential exceeding that neuron's spiking threshold, and this spike timing encodes vital information. However, the threshold is generally treated as a hyperparameter, and incorrect selection can lead to neurons that do not spike for large portions of the training process, hindering the effective rate of learning.**

This work focuses on the significance of learning neuron thresholds alongside weights in SNNs. Our results suggest that promoting threshold from a hyperparameter to a trainable parameter effectively addresses the issue of dead neurons during training. This leads to a more robust training algorithm, resulting in improved convergence, increased test accuracy, and a substantial reduction in the number of training epochs required to achieve viable accuracy on spatiotemporal datasets such as NMNIST, DVS128, and Spiking Heidelberg Digits (SHD), with up to 30% training speed-up and up to 2% higher accuracy on these datasets.

***Index Terms*—Neuromorphic Computing, Spiking Neural Networks, Adaptive Threshold Learning, Robust Training**

I. Introduction

Neuromorphic computing is inspired by biological neural networks, which are known for their high energy efficiency in processing stimulus signals and communication. The emergence of DVS cameras, also referred to as "silicon retina" [1] and "silicon cochlea" devices [2], coupled with the use of implementable, dense on-chip artificial synapses [3]–[5], has propelled the field of neuromorphic engineering towards end-to-end event-based models where the data from event-based input devices would be processed via Spiking Neural Networks (SNNs) mapped on neuromorphic hardware fabrics [6]–[10]. SNNs, in their general form, are composed of Leaky Integrate and Fire (LIF) neurons that have states, and, compared to conventional Artificial Neural Network (ANNs), more closely mimic biological neural networks. LIF neurons are standard neuron models that can be efficiently simulated. SNNs distribute data in the temporal domain rather than the activation amplitude and, when implemented effectively, they will have lower overall memory usage. Beside having state variables (synaptic response current and membrane potential), SNNs also incorporate time as another parameter in their computations. Neuron dynamics and threshold levels specify the timing of spike generation. Since these spikes are sparse, discrete 0/1 events, SNNs are more energy efficient in their computation, and can be implemented in extremely low-power neuromorphic hardware with lower memory access frequency.

However, a key roadblock remains for this SNN vision of the future: SNN training is not as mature or reliable as DNN training. Bio-plausible unsupervised training algorithms, such as Hebbian learning and Spike Timing Dependent Plasticity (STDP) [11], [12], excel at extracting low-level features but often face challenges with generalization and scalability, limiting their application to shallow SNNs [13], [14]. In reinforcement learning [15], [16], synaptic weights are subjected to random perturbations to gauge changes in output error. If the error decreases, the alteration is accepted; otherwise, it is rejected. With large networks, reinforcement learning becomes challenging because the effect of adjusting one weight is overshadowed by noise from others, necessitating numerous trials for meaningful learning. Due to the remarkable success of error backpropagation in training DNNs, most recent strategies for training SNNs prominently leverage gradient calculations and error backpropagation, achieved through either converting pre-trained ANNs to SNNs [17], [18] or directly training SNNs via error backpropagation [20]–[26]. Directly applying error backpropagation to SNNs is however challenging due to the non-differentiable nature of spiking functions. To address this issue, surrogate gradients [25], [27], [28] are used to approximate the spike activation derivative as a continuous function, enabling end-to-end training of SNNs.

Spiking threshold is a crucial hyperparameter with respect to the activity of a neuron, determining the level of input a neuron must receive before it can generate an output spike. Determining optimal spiking threshold levels that allow neurons to emit enough spikes and, in some cases, precisely timing their emission is a critical task in training SNNs, affecting training convergence rate, inference latency, and accuracy. Increasing the threshold can result in more precise coding of

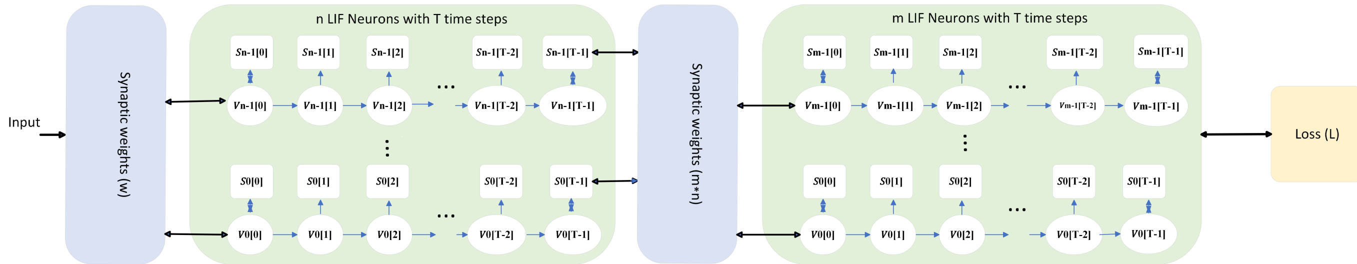


Fig. 1: Illustration of our (two-layer) SNN architecture with spatiotemporal error backpropagation. Arrows pointing to the right and top indicate the forward path, while arrows pointing to the left and bottom represent the backward path.

information, with only the most salient features represented by spikes. Decreasing the threshold can lead to more distributed information coding, with more neurons contributing to the representation of each input feature. In backpropagation-based SNN training, achieving the right balance is crucial to avoid the "dead neuron" problem. This issue arises when neurons fail to spike during the presentation of an input or input batch, leading to a lack of learning under many learning rules [22], [23], [25]. The effectiveness of any SNN training algorithm is therefore sensitive to neuron threshold values, and appropriate threshold values for a given network architecture/dataset are often discovered through offline grid search [18], [19].

In this study, we investigate how incorporating in-loop threshold learning can reduce the number of dead neurons, leading to faster training and improved inference accuracy. We demonstrate that the adaptive threshold learning can achieve up to a 30% speed-up in training (measured by the number of epochs before convergence) and up to a 2% increase in accuracy on neuromorphic datasets NMNIST, DVS128, and Spiking Heidelberg Digits (SHD). The main contribution of this work is an in-depth analysis of the training dynamics and the "dead neuron" problem in training SNNs with surrogate gradient-based error backpropagation.

II. Related work

Several methods have been explored to integrate threshold adjustment into SNNs. The threshold balancing method [18], [19] employs a grid search of thresholds to find the best threshold values for ANN to SNN conversion. Another approach involves the use of adaptive spiking neuron models [29]–[31] and Spike Frequency Adaptation (SFA) mechanisms [32]. These techniques are used to transiently increase the neuron's spiking threshold, thus reducing the neuron's firing rate over time. A comparable strategy is found in the threshold regularization method [25]. This approach increases the thresholds of highly active neurons, making them less responsive to input stimuli, while allowing less active neurons to react more easily to subsequent stimuli. The threshold annealing technique [33] increases the threshold level over time, using a small threshold in early epochs and a larger threshold in later epochs. Additionally, some methods like activity regularization [21], [28] have proposed lower bound and higher bound values on the neuronal spike counts to control spiking activity levels in SNNs. More recent approaches focus on threshold optimization [20], [29], [34], [35]. These techniques utilizes gradient descent to find the threshold values that minimize a loss function. While improved performance has been reported, there hasn't been a dedicated study on the dynamics of threshold learning and its impact on SNN training through error backpropagation.

III. SNN Training using spatiotemporal backpropagation and threshold learning

This section describes the dynamics of Leaky Integrate-and-Fire (LIF) neurons and their training. A LIF neuron i in layer l has two state variables: a synaptic response current $I_i^l[t]$ and a membrane potential $V_i^l[t]$. The synaptic response current is generated by a leaky integrator which integrates the incoming spikes from a stimuli $S_j^{l-1}[t]$. This current then passes through another leaky integrator to produce the membrane potential $V_i^l[t]$. When the membrane potential reaches a spiking threshold level Th_i^l, the neuron sends out a spike $S_i^l[t]$ and then resets to V_{Rest}. A unit step function models the spiking function:

$$S_i^l[t] = \begin{cases} 1, & \text{if } V_i^l[t] \geq Th_i^l \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The spiking function $S_i^l[t]$ is non-differentiable and poses a problem for gradient calculations in error backpropagation. To overcome this issue, a surrogate function is used to approximate the derivative of the spiking function:

$$\frac{\mathrm{d}S_i^l[t]}{\mathrm{d}V_i^l[t]} = \frac{s}{\tau} e^{-\frac{\left|V_i^l[t] - Th_i^l\right|}{\tau}} \tag{2}$$

$$\frac{\mathrm{d}S_i^l[t]}{\mathrm{d}Th_i^l} = -\frac{s}{\tau} e^{-\frac{\left|V_i^l[t] - Th_i^l\right|}{\tau}} \tag{3}$$

where s is a scaling factor, and τ is the steepness parameter ($\tau \to 0$). Each neuron is composed of synapses with associated synaptic weights w_{ji} and adjustable spiking thresholds Th_i^l. Fig. 1 illustrates our fully connected SNN topology, featuring only two layers for simplicity. In each layer, the membrane potential of each neuron $V_i[t]$ updates every discrete time step $t \in \{0, 1, \cdots T-1\}$ for a finite number of time steps T. $S_i[t]$ denotes the spike generated by the neuron i at time t. After the final layer, a loss function calculates the difference between the output spike rates and the target spike rate. We first calculate the gradient of L with respect to V_i^l and Th_i^l, and then use

TABLE I: Top-1 Test Accuracy Scores

Dataset	Training Method	Architecture	Accuracy
NMNIST	Baseline	34x34x2-500-500-10	98.89%
	Rouser	34x34x2-500-500-10	**99.21%**
DVS128	Baseline	128x128x2-64-11	84.72%
	Rouser	128x128x2-64-11	**86.46%**
SHD	Baseline	700-200-200-20	Failed: Training non-convergent
	Rouser	700-200-200-20	**78.14%**

TABLE II: Hyperparameters

Symbols	Description	Value
Th_{init}	Initial threshold	1.25
s	Scaling factor	1.5
τ	Steepness parameter	3.75
lr_W	Synaptic weight's learning rate	0.001
lr_{Th}	Thrshold's learning rate	0.001
-	Current decay	0.75
-	Voltage decay	0.97
V_{Rest}	Resting voltage	0
L	Loss function	MSE
-	Optimizer	Adam
-	Weight initialization	Kaiming

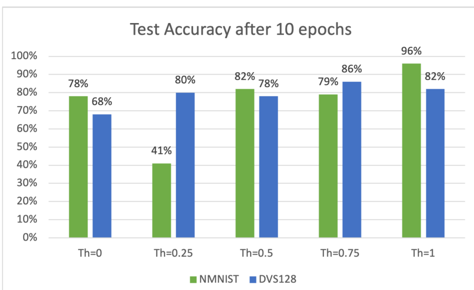


Fig. 2: Threshold grid search on NMNIST and DVS128.

the chain rule to find the update rules for the synaptic weights w_{ji} and the thresholds Th_i^l respectively:

$$w_{ji} = w_{ji} - lr_W * \frac{\partial L}{\partial V_i^l[t]} * \frac{\partial V_i^l[t]}{\partial w_{ji}} \quad (4)$$

$$Th_i^l = Th_i^l - lr_{Th} * \frac{\partial L}{\partial Th_i^l} \quad (5)$$

The learning rate for synaptic weights is denoted as lr_W, while the learning rate for thresholds is lr_{Th}.

IV. Experimental Setup

We used SLAYER from the Lava-DL library [36] as the baseline for training SNNs with error backpropagation. We then extended SLAYER by incorporating threshold learning through error backpropagation (Rouser). Our evaluation is based on the NMNIST [37], DVS128 [38], and Spiking Heidelberg Digits (SHD) [39] datasets.

The NMNIST dataset is a spiking version of the MNIST dataset, where spikes are generated by displaying images on a neuromorphic vision camera equipped with ATIS (Asynchronous Time-based Image Sensor). It contains 60,000 training images, each consisting of 300 time samples, and an additional set of 10,000 test images used to evaluate accuracy.

The DVS128 dataset is an event-based dataset captured using the Dynamic Vision Sensor (DVS) camera. It comprises recordings of 11 hand gestures and consists of ≈20,000-40,000 events per gesture class. The dataset includes 1176 samples with varying temporal dimension allocated for training and 288 samples designated for testing.

The SHD dataset is an audio-based dataset consisting of spoken digits ranging from zero to nine in both the English and German languages. The audio waveforms have been converted into spike trains using an artificial model of the inner ear. The SHD dataset comprises 8,156 training samples and 2,264 test samples with varying time spans.

V. Evaluation

To assess the impact of various threshold levels on the training of SNNs through error backpropagation, we first conducted an experiment using both the NMNIST and DVS128 datasets, systematically varying the threshold values for evaluation while keeping all other hyperparameters and the loss function constant. Fig. 2 shows that SNN training convergence is highly sensitive to the threshold level, and the optimal threshold value differs between these two datasets.

Table I shows the top-1 accuracy of Rouser (with threshold learning) in comparison to the baseline (without threshold learning), along with the network architecture utilized in the study. Table II presents the hyperparameters that were utilized in the study. In addition to the improvement in accuracy, we observed greater robustness in Rouser, particularly during training on SHD.

A. *Rouser's Learning Dynamic*

To better understand the effectiveness of Rouser in SNN training and to gain insights into its convergence latency, we monitored changes in synaptic weights during the training process. Figure 3 illustrates these changes over time in

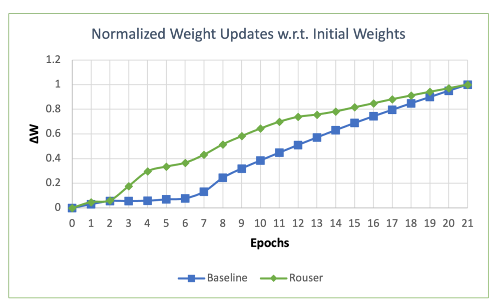
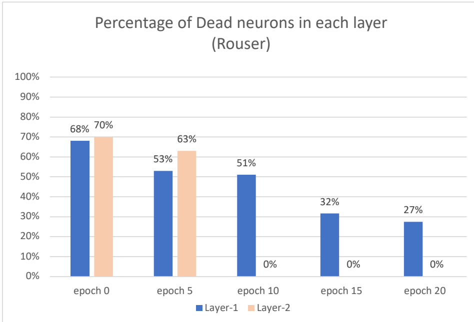
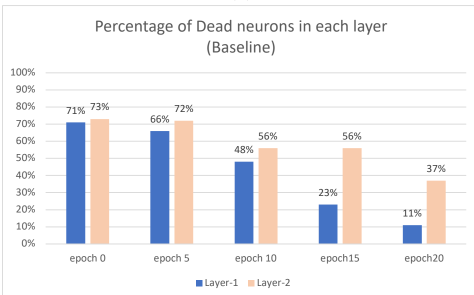


Fig. 3: Weight updates wrt. initial weights during training.

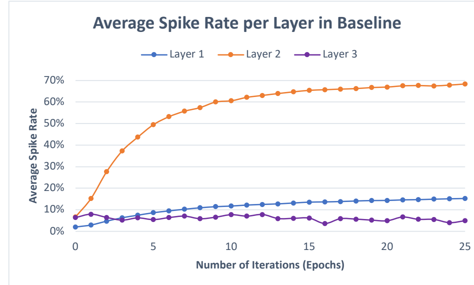


(a)

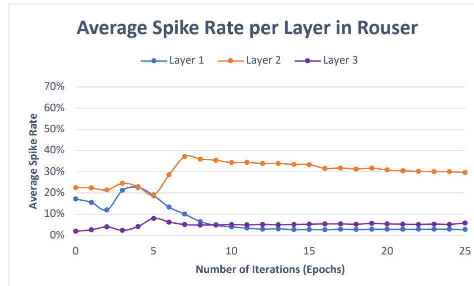


(b)

Fig. 4: The percentage of "dead neurons" for each layer during training on NMNIST.



(a)



(b)

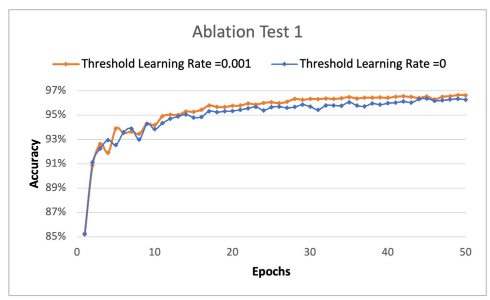
Fig. 5: Average spike rates.

comparison to the initial weights. We observed that Rouser exhibited greater weight changes than the baseline, indicating a faster training process.

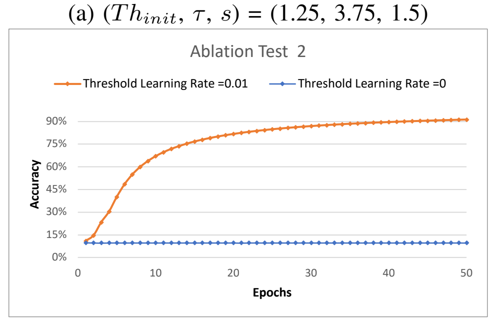
We conducted a further investigation by examining the percentage of dead neurons in each layer during training. Fig. 4 shows a lower count of "dead neurons" in Rouser than the baseline. When the threshold level is high and neurons operate in a sub-threshold region, they have a near-zero gradient, and don't emit any spikes. This results in dead neurons, which are consequently unable to contribute to the learning process. By dynamically adjusting the threshold levels, Rouser enables a greater number of neurons to generate spikes and actively participate in error backpropagation. Additionally, in Fig. 5, we present the average spike rate for NMNIST dataset. Initially, Rouser shows a higher average spike rate in layers 1 and 2 compared to the baseline. However, as training progresses, Rouser shows a reduction in spike rates. The average spike rate in layer 3 remains comparable, likely due to the use of the same loss function (MSE of the spike rate) for both training methods.

B. Ablation Studies

To determine whether the performance gains observed with Rouser are primarily due to threshold learning and not influenced by variations in experiemnts, we conducted ablation studies on Rouser. In these studies, we set the threshold learning rate to zero and ensured that identical hyperparameters and initial weights were used across experiments, while keeping all other variables such as network architecture unchanged. We then increased the threshold learning rates to assess its impact. We conducted these studies under two hyperparameter settings and observed that threshold learning rate of zero resulted in lower accuracy as shown in Fig. 6a, and an inability to train in Fig. 6b.



(a) (Th_{init}, τ, s) = (1.25, 3.75, 1.5)



(b) (Th_{init}, τ, s) = (0.25, 3, 3)

Fig. 6: Ablation studies on NMNIST

VI. Conclusion

In this work, we analyzed Rouser, a method for adaptively learning the threshold of neurons during SNN training, aimed at "rousing" dead neurons. By experimenting with three datasets—NMNIST, DVS128, and SHD—we demonstrated the effectiveness of this technique in enhancing network performance in terms of training latency and testing accuracy. We showed that Rouser's adaptive threshold learning consistently achieved high accuracy and robustness to variations in datasets and initial hyperparameters. Furthermore, Rouser significantly reduced the number of dead neurons during training while better regulating neuron spiking activity, resulting in faster training and improved accuracy.

References

[1] M. Mahowald, "The Silicon Retina," in *An Analog VLSI System for Stereoscopic Vision*, Springer US, Boston, MA, 1994, pp. 4–65.

[2] L. Watts, D. A. Kerns, R. F. Lyon, and C. A. Mead, "Improved implementation of the silicon cochlea," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 5, pp. 692–700, 1992, doi: 10.1109/4.133156.

[3] Y. Nishitani, Y. Kaneko, and M. Ueda, "Supervised learning using spike-timing-dependent plasticity of memristive synapses," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 2999–3008, Dec. 2015, doi: 10.1109/TNNLS.2015.2399491.

[4] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," *Nature Nanotechnology*, vol. 12, no. 8, pp. 784–789, 2017.

[5] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[6] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015. [Online]. Available: http://dx.doi.org/10.1109/TCAD.2015.2474396

[7] S. Höppner, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, M. Stolba, F. Kelber, B. Vogginger, F. Neumärker, G. Ellguth, *et al.*, "The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing," *arXiv preprint arXiv:2103.08392*, 2021.

[8] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[9] Akida Neural Processor, Brainchip, "Technology," *[Online]*. Available: https://www.brainchip.com/technology. [Accessed: Apr. 2024].

[10] GrAI Matter Labs, "Technology," *[Online]*. Available: https://www.graimatterlabs.ai/technology. [Accessed: Apr. 2024].

[11] R. Kempter, W. Gerstner, and J. L. Van Hemmen, "Hebbian learning and spiking neurons," *Phys. Rev. E*, vol. 59, no. 4, p. 4498, 1999.

[12] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nat. Neurosci.*, vol. 3, no. 9, pp. 919–926, 2000.

[13] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, p. 99, 2015.

[14] P. Ferré, F. Mamalet, and S. J. Thorpe, "Unsupervised feature learning with winner-takes-all based STDP," *Front. Comput. Neurosci.*, vol. 12, p. 24, 2018.

[15] H. S. Seung, "Learning in spiking neural networks by reinforcement of stochastic synaptic transmission," *Neuron*, vol. 40, no. 6, pp. 1063–1073, 2003.

[16] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.

[17] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, pp. 54–66, 2015.

[18] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. 2015 Int. Joint Conf. Neural Networks (IJCNN)*, 2015, pp. 1–8, doi: 10.1109/IJCNN.2015.7280696.

[19] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Front. Neurosci.*, vol. 13, p. 95, 2019.

[20] N. Rathi and K. Roy, "DIET-SNN: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 6, pp. 3174–3182, Jun. 2023, doi: 10.1109/TNNLS.2021.3111897.

[21] N. Perez-Nieves and D. Goodman, "Sparse spiking gradient descent," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 11795–11808, 2021.

[22] S. M. Bohte, J. N. Kok, and J. A. La Poutré, "SpikeProp: backpropagation for networks of spiking neurons," in *Proc. ESANN*, Bruges, 2000, pp. 419–424.

[23] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[24] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[25] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Front. Neurosci.*, vol. 10, p. 508, 2016.

[26] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," *Front. Neurosci.*, vol. 14, p. 424, 2020.

[27] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51-63, Nov. 2019, doi: 10.1109/MSP.2019.2931595.

[28] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Comput.*, vol. 33, no. 4, pp. 899–925, 2021.

[29] B. Yin, F. Corradi, and S. M. Bohté, "Effective and efficient computation with multiple-timescale spiking recurrent neural networks," in *Proc. Int. Conf. Neuromorphic Syst.*, 2020, pp. 1–8.

[30] A. Shaban, S. S. Bezugam, and M. Suri, "An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation," *Nature Commun.*, vol. 12, no. 1, p. 4234, Sep. 2021.

[31] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[32] D. Salaj, A. Subramoney, C. Kraisnikovic, G. Bellec, R. Legenstein, and W. Maass, "Spike frequency adaptation supports network computations on temporally dispersed information," *eLife*, vol. 10, p. e65459, Jul. 2021, doi: 10.7554/eLife.65459. [Online]. Available: https://doi.org/10.7554/eLife.65459

[33] J. K. Eshraghian and W. D. Lu, "The fine line between dead neurons and sparsity in binarized spiking neural networks," *CoRR*, vol. abs/2201.11915, 2022, [Online]. Available: https://arxiv.org/abs/2201.11915.

[34] S. Wang, T. H. Cheng, and M. Lim, "LTMD: Learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, pp. 28350–28362, 2022.

[35] Ahmed Hasssan, Jian Meng, and Jae-Sun Seo, "Spiking Neural Network with Learnable Threshold for Event-based Classification and Object Detection," in *Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2024.

[36] Intel, "Lava Software Framework," *[Online]*. Available: https://lava-nc.org/. [Accessed: Sep. 2023].

[37] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, p. 437, 2015.

[38] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.

[39] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 7, pp. 2744–2757, Jul. 2020.