Error-triggered Three-Factor Learning Dynamics for Crossbar Arrays

Melika Payvand[1,*], Mohammed E. Fouda[2,*], Fadi Kurdahi[2], Ahmed Eltawil[2], and Emre O. Neftci[3]
[1]Institute of neuroinformatics, University and ETH of Zurich, Zurich, Switzerland.
[2]Electrical Engineering and Computer Science Dept., UC Irvine, Irvine, CA 92697-2625 USA
[3] Cognitive Sciences Dept. and Computer Science Dept., UC Irvine, Irvine, CA 92697-2625 USA

***Abstract*—Recent breakthroughs suggest that local, approximate gradient descent learning is compatible with Spiking Neural Networks (SNNs). Although SNNs can be scalably implemented using neuromorphic VLSI, an architecture, that can learn *in situ* as accurately as conventional processors, is still missing. Here, we propose a subthreshold circuit architecture designed through insights obtained from machine learning and computational neuroscience that could achieve such accuracy. Using a surrogate gradient learning framework, we derive local, error-triggered learning dynamics compatible with crossbar arrays and the temporal dynamics of SNNs. The derivation reveals that circuits used for inference and training dynamics can be shared, which simplifies the circuit and suppresses the effects of fabrication mismatch. We present SPICE simulations on XFAB 180nm process, as well as large-scale simulations of the spiking neural networks on event-based benchmarks, including a gesture recognition task. Our results show that the number of updates can be reduced hundred-fold compared to the standard rule while achieving performances that are on par with the state-of-the-art.**

I. Introduction

The implementation of learning dynamics as synaptic plasticity in neuromorphic hardware can lead to highly efficient, lifelong learning systems [1]–[5]. While gradient Backpropagation (BP) is the workhorse for training nearly all deep neural network architectures, it is incompatible with neuromorphic hardware because it is not spatially and temporally local [6]. Recent work addresses this problem using Surrogate Gradient (SG) learning [7]. SGs use a differentiable surrogate network to compute weight updates in a local fashion, and formulate the updates as three-factor synaptic plasticity rules. The SG approach reveals from first principles the mathematical nature of the three factors, and a learning dynamic that is continuous in time. While temporal continuity is a plausible property in the brain, while being able to perform a large number of weight updates (writes) which can be energetically expensive in hardware [2].

Here, we demonstrate a crossbar based neuromorphic architecture that efficiently implements SG learning as a three-factor plasticity rule. The problem of continuous updates is solved by triggering weight updates asynchronously when the error exceeds a threshold. We propose subthreshold analog circuits that efficiently implement the neural dynamics and error-triggered updates. We find that the circuits for learning and inference can be shared, which further reduces the circuit complexity, and suppresses mismatch in the peripheral circuits. Taken together, our results demonstrate that the additional circuit complexity for efficient learning with spiking neurons is small compared to a conventional artificial neural network, and could enable efficient spatiotemporal pattern learning in memristor-based crossbar arrays.

II. Neural Network Model

The proposed model consists of networks of plastic integrate-and-fire neurons. Here, the models are formalized in discrete-time to make the equivalence with classical artificial neural networks more explicit. However, these dynamics can also be written in continuous-time without any conceptual changes. The neuron and synapse dynamics are:

$$
\begin{aligned}
U_i^l[n] &= \sum_j W_{ij}^l P_j^l[n] - \delta R_i^l[n], \quad S_i^l[n] = \Theta(U_i^l[n]) \qquad (1)\\
P_j^l[n+1] &= \alpha_j^l P_j^l[n] + Q_j^l[n],\\
Q_j^l[n+1] &= \beta_j^l Q_j^l[n] + S_j^{l-1}[n],\\
R_i^l[n+1] &= \gamma_i^l R_i^l[n] + S_i^l[n].
\end{aligned}
$$

where $U_i^l[n]$ is the membrane potential of neuron i at layer l at time step n, W^l is the synaptic weight matrix between layer $l-1$ and l, and S_i^l is the binary output of this neuron. Θ is the step function, *i.e.* ($S_i^l[n]=1$ when $U_i^l[n] \leq 0$). The constants α_j^l, γ_j^l and β_j^l capture the decay dynamics of the membrane potential U_i^l, the refractory (resetting) state R_i^l and the synaptic state Q_i^l and can be related to time constants in leaky integrate-and-fire neurons. The indices in the time constants, j and l, reflect the circuit-to-circuit variability in these parameters due to fabrication mismatch. States P and Q describe the traces of the membrane and the current-based synapse, respectively. R is a refractory state that resets and inhibits the neuron after it has emitted a spike, and δ is the constant that controls its magnitude. Note that Eq. (1) is equivalent to a discrete-time version of the Spike Response Model (SRM)$_0$ with linear filters [8]. This SNN and the ensuing learning dynamics can be transformed into a standard binary neural network by setting all $\alpha=0$, replacing all $P[n]$ with $S[n-1]$ and dropping Q and R.

III. Surrogate Gradient Learning

Assuming a global cost function $\mathcal{L}$, the gradients with respect to the weights in layer l are formulated as three factors

$$
\frac{\partial}{\partial W_{ij}^l}\mathcal{L} = \frac{\partial}{\partial S_i^l}\mathcal{L}\frac{\partial}{\partial U_i^l}S_i^l\frac{\partial}{\partial W_{ij}^l}U_i^l \qquad (2)
$$

The rightmost factor describes the change of the membrane potential changes with the weight W_{ij}^l. This term is equal to $P_j^l[n] - \delta \frac{\partial}{\partial W_{ij}^l} R_i^l[n]$ for the neuron defined by Eq. (1). The term with R involves a dependence of the past spiking activity of the neuron, which significantly complicates the learning dynamics. Fortunately, this dependence can be ignored during learning without empirical loss in performance [9]. The middle factor is the change in spiking state as a function of the membrane potential, *i.e.* the derivative of Θ. Θ is non-differentiable but can be replaced by a smooth sigmoidal or piecewise constant function in the learning rule [7]. Our experiments make use of a piecewise linear function, such that middle factor becomes the box function: $\frac{\partial}{\partial U_i^l} S_i^l := B(U_i) = 1$ if $u_- < U_i < u_+$ and 0 otherwise. The leftmost factor describes how the change in the spiking state affects the loss. It is commonly called the local error (or the "delta") and is typically computed using gradient BP. We assume for the moment that these local errors are available and denote them err_i^l, and revisit this point in Sec. III-B. The weight updates become:

$$\Delta W_{ij}^l = -\eta \frac{\partial}{\partial W_{ij}^l} \mathcal{L} = -\eta \, err_i^l P_j^l, \text{ if } u_- < U_i < u_+, \quad (3)$$

where η is the learning rate.

A. Error-triggered Learning

For most interesting cost functions, errors must be computed extrinsically and communicated to the neuron. To make this communication efficient, we encode errors using positive and negative events as follows:

$$E_i^l = sign(err_i^l)[|err_i^l| - \theta]^+ \quad (4)$$

where $\theta \in \mathbb{R}$ is a constant or slowly varying error threshold and $[\cdot]^+$ is the recitifed linear function. Using this encoding, the parameter update rule becomes:

$$\Delta W_{ij}^l = -\theta E_i^l P_j^l B(U_i^l) \quad (5)$$

where θ is called the stop-learning threshold (η was folded into θ). Thus, an update takes place on an error of magnitude θ and if $B(U_i^l) = 1$. The sign of the weight update is $-E_i^l$ and its magnitude θP_j^l. Provided that the layer-wide update magnitude can be modulated proportionally to θ, this learning rule implies two comparisons and an addition (subtraction).

B. Local Losses and Local Errors

Up to now, we have side-stepped the calculation of $err[n]_i^l$. If l is not the output layer, then computing this term requires solving a deep credit assignment problem. Gradient BP can solve this, but is not compatible with a physical implementation of the neural network [6]. Several approximations have emerged recently to solve this, such as feedback alignment [10]–[12], and local losses defined for each layer [**?**], [**?**], [13]. For classification, local losses can be local classifiers (using output labels) [**?**], and supervised clustering, which perform on par and sometimes better than BP in classical ML benchmark tasks [13]. For all experiments used in this work, we use a layer-wise local classifier using a mean-squared error loss defined as $\mathcal{L}_i^l = || \sum_{k=1}^{C} (J_{ik}^l S_k^l - \hat{y}_k)||_2$, where J_{ik}^l is a random, fixed matrix, $\hat{y}_k$ are one-hot encoded labels, and C is the number of classes. The gradients of $\mathcal{L}_i^l$ involve backpropagation within the time step n and thus requires the symmetric transpose, $J^{l,T}$. If this symmetric transpose is available, then $\mathcal{L}$ can be optimized directly. To account for the case where J^T is unavailable, for example in mixed signal systems, we train through feedback alignment using another random matrix H^l [10] whose elements are equal to $H_{ij}^l = J_{ij}^{l,T} \omega_{ij}^l$ with Gaussian distributed $\omega_{ij}^l \sim N(1, \frac{1}{2})$, where T indicates transpose. Weight updates are achieved through stochastic gradient descent (SGD). We note that an error can be computed with any loss function (*e.g.* mean-squared error or cross entropy) provided there is no temporal dependency, i.e. $\mathcal{L}[n]$ does not depend on variables in time step $n-1$. If such temporal dependencies exist, for example with Van Rossum distance [9], the learning rule becomes considerably more complex and Eq. (3). The matrices J^l and H^l can be very large, especially in the case of convolutional networks. One solution to the memory footprint of J^l is to generate these matrices on the fly using a random number generator [**?**]. Another solution is to define J^l as a sparse, binary matrix [6]. Using a binary matrix would further reduce the need for multiplications in the computation of err_i.

C. Hardware Realization with Memristor Crossbar Arrays

The emerging technologies, such as memristors (RRAMs), phase change memory, and spin transfer torque-RAM in addition to other MOS realizations such as floating gate transistors, assembled as crossbar array enable the VMM operation to be completed in a single step. This is unlike other hardware solutions which requires $N \times M$ steps where N and M are the size of the weight matrix. These emerging technologies implement only positive weight (excitatory connections), however, the negative weights (inhibitory connections) are necessary. There are two ways to realize the positive and negative weights [14]; 1) balanced realization where two devices are needed to implement the weight value which is stored in the devices' conductances where $w = G^+ - G^-$. If the G^+ is greater/less than G^-, it represents positive/negative weight, respectively. 2) Unbalanced realization where one device is used to implement the weight value with a common reference conductance G_{ref}, set to the mid value of the conductance range. Thus, the weight value is represented as $w = G - G_{ref}$. If the G is greater/less than G_{ref}, it represents positive/negative weight, respectively. In this work, we use the unbalanced realization since it saves the area and power in expense of using half of the device's dynamic range. Thus, the memristive SNN can be written as

$$U_i^l[n] = \sum_j \left(G_{ij}^l - G_{ref} \right) P_j^l[n]. \quad (6)$$

By following the same analysis in section III-A, the conductance update model is the same as Eq. (3). The general architecture of the proposed dynamics is shown in Fig. 1.

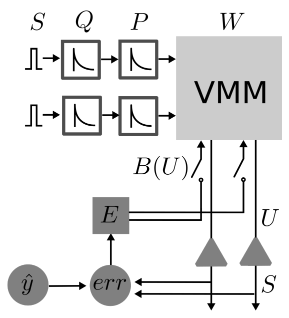

Fig. 1. Architecture of the Three-Factor Error-Triggered Rule. Input spikes S are integrated through P. The vector P is then multiplied with W resulting in U. Output spikes S are then compared with local targets $\hat{y}$ and bipolar error events E are fed back to each neuron. Updates are made if $u_- < U < u_+$. R is omitted in this diagram

D. Inference and Learning Circuits

Our circuit implementation of the spiking neural network differs from classical ones. Generally, the rows of crossbar arrays are driven by (spikes) and integration takes place at each column [15]. While this is beneficial in reducing read power, it renders learning more difficult because the variables necessary for learning in SNNs are not local to the crossbar. Instead, we use the crossbar as a vector-matrix multiplication of pre-synaptic trace vectors P^l and synaptic weight matrices W^l. Using this strategy, a single trace P^l_i per neuron supports both inference and learning. Furthermore, this property means that learning is immune to the mismatch in P^l, and can even exploit this variation for reducing the loss. Fig. 2 depicts the details of the learning circuits in a crossbar-like architecture which is compatible with the address-event representation (AER) as the conventional scheme for communication between neuronal cores in many neuromorphic chips [16]. In this circuit, only P is shown and $\alpha_Q = 0$. This type of architecture includes multi-T/1R [17]. The traces P are generated through a Differential-Pair Integrator (DPI) circuit which generates a tunable exponential response at each input event in the form of a sub-threshold current [18]. The current is linearly converted to voltage using pseudo resistors in the I-to-V block highlighted in the red box in Fig. 2. The exponentially decaying voltage is buffered and drives the entire crossbar row in accordance with Eq. (1).

For every neuron, different voltages (corresponding to P_j) are applied to the top electrode of the corresponding memristive device whose bottom electrode is pinned by the crossbar front-end highlighted in yellow (Fig. 2). This block pins the entire column to a reference voltage ($Vref$) and reads out the sum of the currents generated by the application of Ps across the memristors in the column. As a result, a voltage is developed on the gate of the M1 connected to a differential pair which re-normalizes the sum of the currents from the crossbar to I_{norm}. This ensures that the currents remain in the sub-threshold regime for the next stage of the computation which is the ternary error generation as is specified in equation (4). This is done through the Variable Width Bump (VWBump) circuit that compares I_{nn} to the target $\hat{y}$, with a stop region. Thus, the VWBump circuit output indicates the sign of the weight update (up or down) or stop-learning (no update). The circuit (not shown) is based on the bump circuit [19], which consists of a differential pair for the comparison and a current correlator for the stop region, and is modified to have a tunable stop-learning region [20]. The boundaries of this region play the role of θ in (4). The output of the block is plotted in the inset of Fig. 2, which shows the Up, Down and STOP outputs.

The Up and Down signals trigger the oscillators highlighted in blue which generate the bipolar E_i events. According to Eq. (5), the magnitude of the weight update is P_j, and thus P_j must be sampled at the onset of E_i. To do so, we regenerate the exponential current in the entire row by propagating pbias shown in the DPI circuit block (green) and sample it by the up and down events. This is done through the sampling circuit which consists of two PMOS transistors in series connected to the up/down events and pbias respectively. The NMOS transistor is biased to generate a current much smaller than that of the DPI and as a result, the higher the DPI current, the higher the input of the following inverter during the event pulse, and thus it takes longer for the NMOS to discharge that node. This results in a pulse width varying linearly with P_j, in agreement with Eq. (5). The linear pulse width can be approximated with multiple pulses which results in a linear conductance update in memristive devices [21].

Fig. 3 illustrates the Spectre results of the above circuits designed in XFAB 180nm process. With every input event, the DPI current (and therefore EPSP) undergoes a near instantaneous jump and decays exponentially. The EPSP is buffered and applied to the memristive device whose other side is pinned at $vdd/2$ (0.9V). $Vmmr$, the voltage drop across the device, follows the EPSP except for the time it is being programmed. $VNrn$, marked on Fig. 2, is used to mirror the normalized crossbar current (I_{nn}) to the bump circuit and is shown in green in Fig. 3. In the beginning, while the EPSP is low, I_{nn} is lower than the target, therefore, the UP output of the VWBump circuit is high and the UP events are generated through the oscillator. As the neuron gets closer to the target (because of the integrated input events), the STOP output of VWBump is flipped to high and the event generation stops. The UP events sample the EPSP to change the synaptic weight correspondingly. While the EPSP is low, no programming pulse is generated. For higher values of the EPSP, the pulse width is higher and it falls as the EPSP decays. This is highlighted in the inset of the $Vmmr$. Note that the memristor model (and thus the synaptic update) is not included in the circuit simulations and we are only showing the programming conditions which would cause the conductance change based on the online learning algorithm.

IV. Large-scale Simulation Experiments

An important feature of the used learning rule is its scalability to multilayer networks with very small loss of performance compared to a standard deep neural network when using idealized dynamics. To demonstrate this, we simulate the learning dynamics for classification in large-scale, multilayer spiking networks. The GPU simulations focus on event-based datasets acquired using a neuromorphic sensor, namely the N-MNIST and DVS Gestures dataset for demonstrating the learning model. Both datasets were pre-processed as in [**?**]. The N-MNIST network is fully connected (1000–1000–1000),

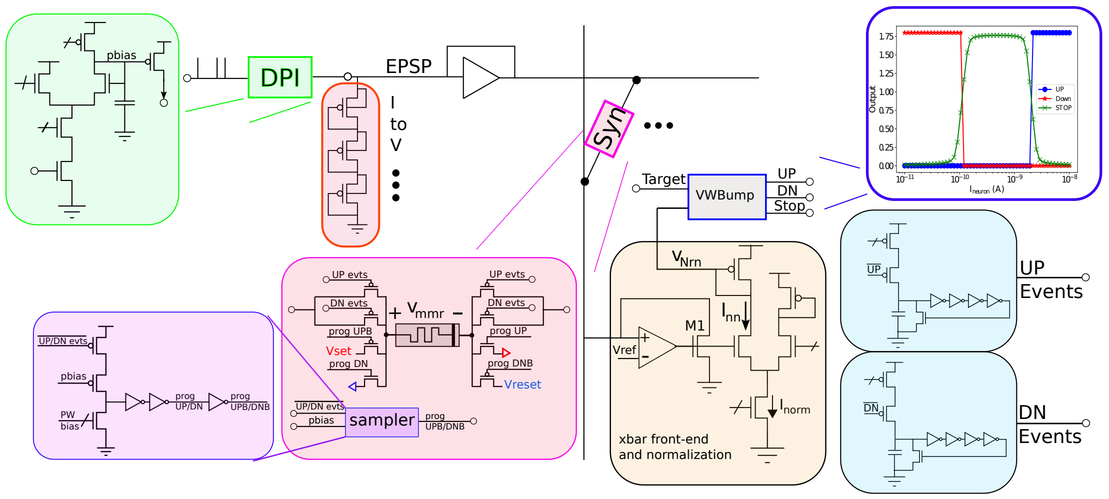

Fig. 2. Details of the architecture and learning circuits. Green: DPI circuit generating P in the current form. Red: Pseudo resistors converting input current into a voltage driving the crossbar array. Pink: Synapse with the controlling switches. Purple: Sampling circuitry generating pulses to program the devices. Yellow: Crossbar front-end and normalization of the crossbar current. Dark blue: Bump circuit comparing the crossbar current to a target and generating the direction of the error. Light blue: Bidirectional neuron producing up and down events.

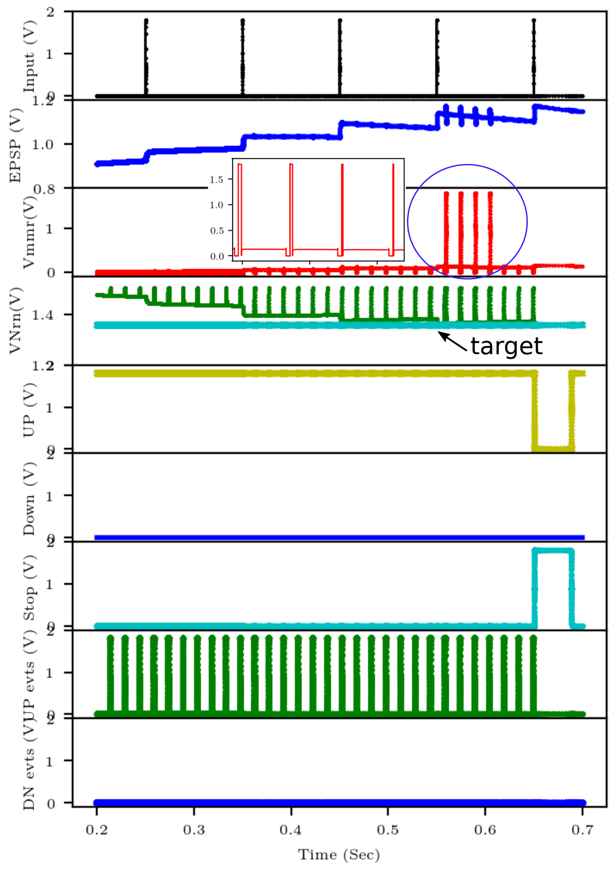

Fig. 3. SPICE simulation results of the learning circuits generating the appropriate programming pulses across the memristive device ($Vmmr$) depending on the value of the $EPSP$ at the onset of the error events.

while the DVS Gestures network is convolutional (64c7-128c7-128c7). For practical reasons, the neural networks were trained in minibatches of 72 (DVS Gestures) and 100 (N-MNIST). We note that the choice of using minibatches is advantageous when using GPUs to simulate the dynamics and is not specific to Eq. (3), and can also be used for gradient backpropagation through time and spike-timing dependent plastisity. The parameters of our model are similar to that of [?] except that the time constants were randomized. In our experiments, we used a proportional controller to adjust θ such as the average error spike rate $\langle E \rangle$ remains stable. The column writes indicates an upper bound on the number of weight writes. It is an approximate upper boundary, as the effect of $B(U)$ has not been taken into account. These results in Tab. I demonstrate a small loss in accuracy across the two tasks when updates are error-triggered. As comparison, published work on DVS Gestures with spiking neurons trained with backpropagation achieved 5.41% [22] and 6.36% [23] error rates and 1.3% [24].

TABLE I
Recognition Error in Idealized Spiking Neural Network Simulations Averaged over 5 Runs.

	DVSGesture		**N-MNIST**	
$\langle E \rangle$	Error	Writes	Error	Writes
Cont.	3.82%	1M	2.3%	1.5M
50Hz	4.22%	50k	2.31%	75k
10Hz	6.25%	10k	2.71%	45k

V. Conclusion

In this article, we demonstrated an error-triggered learning rule that is particularly well-suited for implementation in crossbars. Our implementation leverages the linear property of the subthreshold dynamics, such that the memory required for computing the gradients (i.e. the synaptic traces) grows linearly with the neurons (hence one P_j per input neuron). By updating weights asynchronously (when errors occur), the number of weight writes can be drastically reduced. The proposed learning rule has the same computational footprint as error-modulated Spike Time Dependent Plasticity (STDP) but is functionally different in that there is no acausal part, the updates are triggered on errors if the membrane potential is close to the firing threshold (rather than post-synaptic spike STDP). A more detailed comparison of the scaling of this family of learning rules is provided in [?].

Our proposed implementation still requires a programming circuitry (8 transistors) per synapse along with transistors which switch the memristive device in an out of the network for read/programming. However, the transistors can take advantage of the technology scaling (in contrast to the capacitors whose area do not change as much with the scaling of the nodes).

Despite of the huge benefit of the crossbar array structure, the memristor devices suffer from many challenges that might affect the performance unless taken into consideration in the training such as asymmetric nonlinearity, precision and retention [25]. Fortunately, online learning helps with other problems such as sneak path (i.e wire resistance) and endurance. With error-triggered learning rule, only selected devices are updated which leads to extending the lifetime of the devices and less write energy consumption. The aforementioned non-idealities will be considered in our future work.

VI. Acknowledgements

We acknowledge Giacomo Indiveri for fruitful discussions on the learning circuits. This work was supported by the National Science Foundation under grant 1652159 (EON); the Korean Institute of Science and Technology 1652159 (EON); the National Science Foundation under grant 1640081 (EON, MF); and the Nanoelectronics Research Corporation (NERC), a wholly-owned subsidiary of the Semiconductor Research Corporation (SRC), through Extremely Energy Efficient Collective Electronics (EXCEL), an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.003 (EON, MF); and Toshiba Corporation (MP).

References

[1] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. PP, no. 99, pp. 1–1, 2018.

[2] E. O. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," *iScience*, vol. 5, pp. 52–68, jul 2018.

[3] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol, "A 0.086-mm ^{2}12.7-pJ/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS," *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 1, pp. 145–158, 2018.

[4] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in neuroscience*, vol. 9, 2015.

[5] S. Friedmann, J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier, "Demonstrating hybrid learning in a flexible neuromorphic hardware system," *IEEE transactions on biomedical circuits and systems*, vol. 11, no. 1, pp. 128–142, 2017.

[6] P. Baldi, P. Sadowski, and Z. Lu, "Learning in the machine: The symmetries of the deep learning channel," *Neural Networks*, vol. 95, pp. 110–133, 2017.

[7] 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.

[8] W. Gerstner and W. Kistler, *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[9] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multi-layer spiking neural networks," *arXiv preprint arXiv:1705.11146*, 2017.

[10] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, vol. 7, 2016.

[11] E. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers in Neuroscience*, vol. 11, p. 324, Jun 2017.

[12] A. Nø kland, "Direct feedback alignment provides learning in deep neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 1037–1045.

[13] A. Nøkland and L. H. Eidnes, "Training neural networks with local error signals," *arXiv preprint arXiv:1901.06656*, 2019.

[14] M. E. Fouda, E. Neftci, A. Eltawil, and F. Kurdahi, "Independent component analysis using rrams," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 611–615, Nov 2018.

[15] P.-Y. Chen, B. Lin, I.-T. Wang, T.-H. Hou, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, and S. Yu, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 194–199.

[16] S. Deiss, R. Douglas, and A. Whatley, "A pulse-coded communications infrastructure for neuromorphic systems," in *Pulsed Neural Networks*, W. Maass and C. Bishop, Eds. MIT Press, 1998, ch. 6, pp. 157–78.

[17] M. Payvand, M. V. Nair, L. K. Müller, and G. Indiveri, "A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: From mitigation to exploitation," *Faraday Discussions*, vol. 213, pp. 487–510, 2019.

[18] C. Bartolozzi and G. Indiveri, "Synaptic dynamics in analog VLSI," *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, Oct 2007.

[19] T. Delbruck, ""Bump" circuits for computing similarity and dissimilarity of analog voltages," in *Proc. Int. Joint Conf. Neural Networks*, Jul 1991, pp. I–475–479.

[20] M. Payvand and G. Indiveri, "Spike-based plasticity circuits for always-on on-line learning in neuromorphic systems," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[21] J. Frascaroli, S. Brivio, E. Covi, and S. Spiga, "Evidence of soft bound behaviour in analogue memristive devices for neuromorphic computing," *Scientific reports*, vol. 8, no. 1, p. 7178, 2018.

[22] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *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.

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

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

[25] M. Fouda, F. Kurdahi, A. Eltawil, and E. Neftci, "Spiking neural networks for inference and learning: A memristor-based design perspective," *arXiv preprint arXiv:1909.01771*, 2019.