NEUROMORPHIC
Computing and Engineering

PAPER

ETLP: event-based three-factor local plasticity for online learning with neuromorphic hardware

Fernando M Quintana[1,*], **Fernando Perez-Peña**[1], **Pedro L Galindo**[1], **Emre O Neftci**[2], **Elisabetta Chicca**[3] and **Lyes Khacef**[3]

[1] School of Engineering, University of Cádiz, 11519 Puerto Real, Cádiz, Spain
[2] Neuromorphic Software Ecosystems, Peter Grünberg Institute, Forschungszentrum Jülich, Technologie Zentrum Aachen Dennewartstraße 25–27, 52068 Aachen, Germany
[3] Bio-inspired Circuits and Systems (BICS), University of Groningen, Nijenborgh 3, NL-9747AG Groningen, The Netherlands
[*] Author to whom any correspondence should be addressed.

E-mail: fernando.quintana@uca.es

Keywords: brain-inspired computing, spiking neural networks, three-factor local plasticity, online learning, neuromorphic hardware.

OPEN ACCESS

RECEIVED
1 February 2024

REVISED
8 July 2024

ACCEPTED FOR PUBLICATION
23 July 2024

PUBLISHED
2 August 2024

Abstract

Neuromorphic perception with event-based sensors, asynchronous hardware, and spiking neurons shows promise for real-time, energy-efficient inference in embedded systems. Brain-inspired computing aims to enable adaptation to changes at the edge with online learning. However, the parallel and distributed architectures of neuromorphic hardware based on co-localized compute and memory imposes locality constraints to the on-chip learning rules. We propose the event-based three-factor local plasticity (ETLP) rule that uses the pre-synaptic spike trace, the post-synaptic membrane voltage and a third factor in the form of projected labels with no error calculation, that also serve as update triggers. ETLP is applied to visual and auditory event-based pattern recognition using feedforward and recurrent spiking neural networks. Compared to back-propagation through time, eProp and DECOLLE, ETLP achieves competitive accuracy with lower computational complexity. We also show that when using local plasticity, threshold adaptation in spiking neurons and a recurrent topology are necessary to learn spatio-temporal patterns with a rich temporal structure. Finally, we provide a proof of concept hardware implementation of ETLP on FPGA to highlight the simplicity of its computational primitives and how they can be mapped into neuromorphic hardware for online learning with real-time interaction and low energy consumption.

1. Introduction

The quest for *intelligence* in embedded systems is becoming necessary to sustain the current and future developments of edge devices. Specifically, the rapid growth of wearable and connected devices brings new challenges for the analysis and classification of the streamed data at the edge, while emerging autonomous systems require adaptability in many application areas such as brain-machine interfaces, AI-powered assistants, or autonomous vehicles. In addition, the current trend points toward increased autonomy and closed-loop feedback, where the results of the data analysis are translated into immediate action within a very tight latency window [52]. Hence, in addition to input (sensors) and output (actuators) modalities, those systems need to combine understanding, reasoning, and decision-making [52], and off-line training techniques cannot account for all possible scenarios. Consequently, there is a need for online learning in embedded systems, i.e. changing the parameters of the deployed neural network on-the-fly to adapt to new sensory inputs, under severe constraints of low latency and low power consumption. Online learning implies that gradients are computed along with the data flow during the forward pass of the neural network, in contrast with offline training techniques like back-propagation through time (BPTT) where the gradients are computed in a separate backward pass which requires extra memory, latency, and energy.

At the same time, artificial intelligence (AI) based on deep learning has led to immense breakthroughs in many application areas such as object detection/recognition and natural language processing thanks to the availability of big databases and the increasing computing power of standard CPU and GPU hardware. However, the sources of this increasing computing performance have been challenged by the end of Dennard scaling [15, 16] in around 2005, resulting in the inability to increase clock frequencies significantly. Today, advances in silicon lithography still enable the miniaturization of electronics, but as transistors reach atomic scale and fabrication costs continue to rise, the classical technological drive that has supported Moore's Law for fifty years is reaching a physical limit and is predicted to flatten by 2025 [55]. Hence, deep learning progress with current models and implementations will be constrained by its computational requirements and will be technically, economically, and environmentally unsustainable [60, 61]. The way forward will thus require simultaneous paradigm shifts in the computational concepts and models as well as the hardware architectures, by specializing the architecture to the task [10, 55].

Therefore, in contrast to standard von Neumann computing architectures where processing and memory are separated, neuromorphic architectures get inspiration from the biological nervous system and propose parallel and distributed implementations of synapses and neurons where computing and memory are co-localized and processing is asynchronous [9, 37]. Neuromorphic implementations have grown in the past thirty years to cover a wide range of hardware [2, 54], from novel materials and devices [6, 48] to analog [9, 26, 50] and digital CMOS technologies [22, 40, 59]. Recent works have shown important gains in computation delay and energy-efficiency when combining event-based sensing (sensor level), asynchronous processing (hardware level), and spike-based computing (algorithmic level) in neuromorphic systems [8, 14, 41, 62]. On the other hand, adaptation with online learning is still ongoing research because neuromorphic hardware constraints prevent the use of exact gradient-based optimization with (BP) and impose the use of local plasticity [29, 43, 64].

Gradient-based learning [31] has been recently applied to offline training of spiking neural networks (SNNs), where the BPTT algorithm coupled with surrogate gradients (approximation of the activation function in the backward pass to enable differentiation and gradients flow) is used to solve the (1) temporal credit assignment by unrolling the SNN like in standard recurrent neural networks (RNNs) [43], as well the (2) spatial credit assignment by assigning the 'blame' to each neuron across the layers with respect to the objective function to optimize. These operations require global computations that are not biologically plausible [4, 47] and do not fit with online learning in neuromorphic hardware. Specifically:

- BPTT is non-local in time because it keeps all the network activities for the duration of each sample, which requires a large memory footprint, and it operates in two phases, which adds latency in the backward pass.
- BPTT is non-local in space because it back-propagates the gradient back from the output layer to all the other layers. It adds further latency and creates a locking effect [12] since a synapse cannot be updated until the feedforward pass, error evaluation, and backward pass are completed. It also creates a weight transport problem [33] since the error signal arriving at some neuron in a hidden layer must be multiplied by the strength of that neuron's forward synaptic connections to the source of the error, which requires extra circuitry and consumes energy [28].

The lack of locality in time and space makes BPTT unsuitable for the hardware constraints of neuromorphic implementations for which latency, efficiency, and scalability are of primary concern.

Recently, intensive research in neuromorphic computing has been dedicated to bridging the gap between gradient-based learning [31, 32] and local synaptic plasticity [24, 35, 36, 38, 58] by reducing the non-local information requirements. However, these local plasticity rules often resulted in a cost in accuracy for complex pattern recognition tasks [17]. On the one hand, temporal credit assignment can be approximated by using eligibility traces [3, 63] that solve the distal reward problem by bridging the temporal gap between the network output and the feedback signal that may arrive later in time [27]. This mechanism is used in the eProp learning rule [3]. On the other hand, several strategies are explored for solving the spatial credit assignment by using feedback alignment [33], direct feedback alignment [44] and random error BP [42]. These approaches partially solve the spatial credit assignment problem [17], since they still suffer from the locking effect. This constraint is further relaxed by replacing the global loss with several local loss functions [28, 39, 43] as in the DECOLLE learning rule [28] or by using fixed random learning signals as in the direct random target projection (DRTP) learning rule [19]. Other local learning rules have been proposed for biological plausibility such as the prescribed error sensitivity (PES) learning rule [18, 34], or from a close biological inspiration using two-factor plasticity mechanisms which are inherently compatible with neuromorphic circuits [29].

In this work, we present the event-based three-factor local plasticity (ETLP) rule for online learning scenarios with spike-based neuromorphic hardware where a teaching signal can be provided to the network.

The long-term objective is to enable intelligent embedded systems to adapt to new classes or new patterns and improve their performance after deployment. ETLP is local in time by using spike traces [3], and local in space by using DRTP [19] which means that no error is explicitly calculated. Instead, the labels are directly provided to each neuron in the form of a third-factor that triggers the weight update asynchronously. We present in the next section the algorithmic formulation of ETLP, then apply it to visual and auditory event-based pattern recognition problems with different levels of temporal information. We explore the impact of an adaptive threshold mechanism in spiking neurons as well as explicit recurrence in the network and compare ETLP to eProp [3], DECOLLE [28] and BPTT in terms of accuracy and computational complexity. Finally, we demonstrate a proof of concept hardware implementation of ETLP on field-programmable gate array (FPGA) and discuss possible improvements for online learning in real-world applications.

2. Methods

To achieve online learning based solely on local information, ETLP is divided into two main parts: (1) the online calculation of the gradient to solve the temporal dependency as previously done in other works such as eProp [3] and DECOLLE [28], and (2) the update of the gradient with DRTP [21] to solve the spatial dependency by directly using the labels instead of a global error calculation.

2.1. Neuron and synapse model

The spiking neuron model used in this paper is the adaptive leaky-integrate and fire (ALIF) with a soft-reset mechanism proposed in [3].

$$A_j(t) = v_{th} + \theta a_j(t) \tag{1a}$$
$$a_j(t) = \gamma a_j(t-1) + s_j(t-1) \tag{1b}$$
$$v_j(t) = \alpha v_j(t-1) + \sum_i W_{ij} x_i(t) - s_j(t-1)\, v_{th} \tag{1c}$$
$$s_j(t) = H\big(v_j(t) - A_j(t)\big) \tag{1d}$$

where v_j describes the voltage of neuron j with a decay constant α, W_{ij} the synaptic weight between neuron j and input i, x_i the input from the neuron i of the previous layer, s_j the spike state (0 or 1) of the neuron j, H the activation function, that in our case is the step function and A_j the instantaneous threshold obtained from the baseline threshold (v_{th}) and a trace (a_j) produced by the activity of neuron j, with a scaling factor θ and a decay constant γ.

2.2. Temporal dependency

Typically, the training of SNN is based on many-to-many (sequence-to-sequence) models, where an error is calculated at each time step and then the gradient is calculated based on the combination of these errors [64]. In our case, in order to perform an event-driven update, we have used a many-to-one model where only the result of the network obtained at a specific time step is taken into account. This is shown in the computational graph in figure 1, where the gradient is computed at timestep t. The gradient obtained from the ALIF has only the addition of a threshold-dependent trace in the pre-synaptic spike trace, compared to a leaky-integrated and fire (LIF) [3]. Therefore, for simplicity in this section, the calculation of the gradient is based on a LIF neuron.

In classification problems, the cross-entropy loss function is often used. However, since this depends on the softmax activation function which is not local (it depends on all the output values of the neurons for the final calculation), we have used the mean squared error (MSE). Consequently, the gradient of the error with respect to the network weights (according to equation (1) and its computational graph associated on figure 1) would be as in equation (2*b*), where the final gradient is the sum over the gradients on all timesteps,

$$\frac{\mathrm{d}E}{\mathrm{d}W} = \frac{\partial E}{\partial V(t)}\frac{\partial V(t)}{\partial W} + \frac{\partial E}{\partial V(t-1)}\frac{\partial V(t-1)}{\partial W} + \cdots + \frac{\partial E}{\partial V(0)}\frac{\partial V(0)}{\partial W} \tag{2a}$$
$$\frac{\mathrm{d}E}{\mathrm{d}W} = \frac{\partial E}{\partial V(t)} I(t) + \frac{\partial E}{\partial V(t-1)} I(t-1) + \cdots + \frac{\partial E}{\partial V(0)} I(0)\,. \tag{2b}$$

Following equation (1) and using the MSE error, the gradients of the network are as follows:

$$\begin{array}{ll} \frac{\partial S(t)}{\partial V(t)} = \phi(V(t) - A(t)) = \varphi(t) & \frac{\partial V(t)}{\partial V(t-1)} = \alpha \\ \frac{\partial E}{\partial S(t)} = S(t) - S^* & \frac{\partial V(t)}{\partial W} = I(t) \end{array} \tag{3}$$

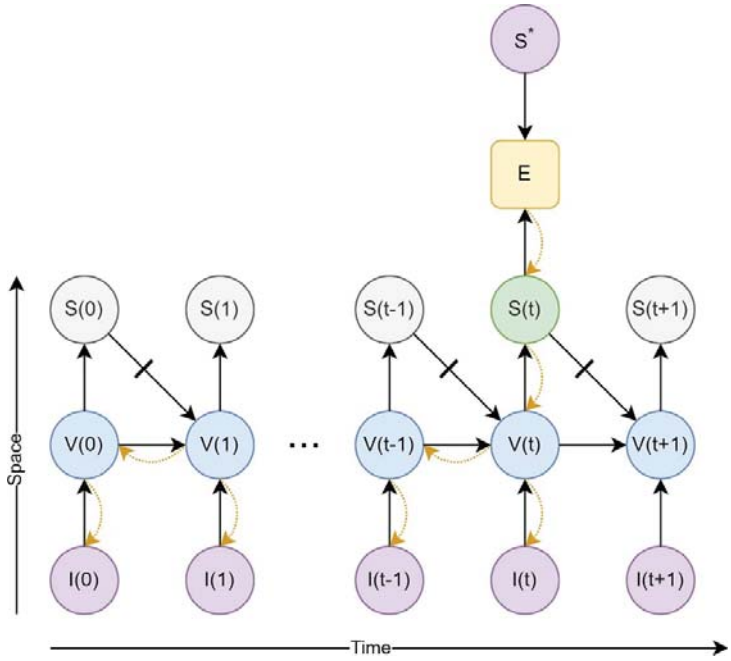


Figure 1. Computational graph for a five time steps simulation on a 1-layer SNN using soft-reset LIF neurons and back-propagation through time. Where I, V, and S are vectors representing the neuron input, voltage and output spike, respectively. S^* is the target output and E is the error. The black arrows represent the forward pass of the data, while the curved orange arrows represent the gradient backward pass. The neuron resets is detached from the computational graph in the gradient calculation.

where $\phi(x)$ is the surrogate gradient of the Heaviside function (H), (for simplicity we will rename $\phi(V(t)-A(t))$ as $\varphi(t)$), S^* the target output, $S(t)$ the output spike of the neuron and I(t) the input at time t.

Substituting equation (3) on equation (2*b*) as following on equations (4*a*)–(4*d*), the gradient of the error in each timestep can be calculated using a pre-synaptic ($\varepsilon_{\text{pre}}(t)$), post-synaptic ($\varphi(t)$) components, and a teaching signal based on the loss function,

$$\frac{\mathrm{d}E(t)}{\mathrm{d}W} = \frac{\mathrm{d}E(t)}{\mathrm{d}V(t)} I(T) + \cdots + \frac{\mathrm{d}E(t)}{\mathrm{d}V(t)} \frac{\partial V(t)}{\partial V(t-1)} \cdots \frac{\partial V(1)}{\partial V(0)} I(0) \tag{4a}$$

$$\frac{\mathrm{d}E(t)}{\mathrm{d}W} = (S(t)-S^*)\,\varphi(t)\,I(t) + \cdots + (S(t)-S^*)\,\varphi(t)\;I(0)\,\alpha^t \tag{4b}$$

$$\frac{\mathrm{d}E(t)}{\mathrm{d}W} = (S(t)-S^*)\,\varphi(t)\underbrace{\left(I(t)+\alpha I(t-1)+\cdots+\alpha^t I(0)\right)}_{\varepsilon_{\text{pre}}(t)=\sum_{i=0}^{t}\alpha^{t-i}I(i)} \tag{4c}$$

$$\frac{\mathrm{dE}(t)}{\mathrm{dW}} = \underbrace{\varepsilon_{\text{pre}}(t)\varphi(t)}_{e(t)=\varepsilon_{\text{pre}}(t)\varphi(t)}\;(S(t)-S^*). \tag{4d}$$

Assuming that the error is calculated at time $t=T$ and calling $\frac{\mathrm{d}E(t)}{\mathrm{d}V(t)} = (S(t)-S^*)$, the gradient of the error with respect to the weights in a period of time T is defined as an equation of three factors (equation (4*d*)) which we will call: (1) pre-synaptic spike trace (blue), (2) surrogate gradient of the post-synaptic voltage (green) and (3) external signal (red). The pre-synaptic spike trace can be expressed as a low-pass filter of the input spikes to the neuron at each synapse, derived from equation (4*c*) as formulated in equation (5),

$$\varepsilon_{\text{pre}}(t) = \alpha\varepsilon_{\text{pre}}(t-1) + I(t). \tag{5}$$

Since threshold adaptation must be taken into account for the gradient computation of the ALIF model, the product of (1) the pre-synaptic spike trace and (2) the surrogate gradient of the post-synaptic voltage is modified according to equation (6) [3], where $e(t)$ is a function of (1) the pre-synaptic spike trace $\varepsilon(t)$, (2)

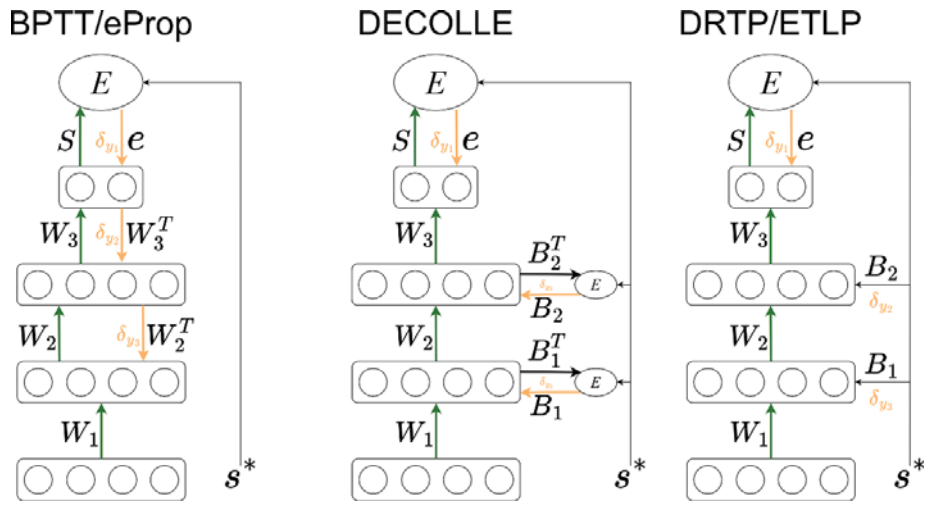


Figure 2. Weight update mechanism on the spatial domain for BPTT, eProp, DECOLLE, DRTP, and ETLP. In the case of BPTT and eProp, an error is calculated using the output layer and the target S^*, the error is then propagated backward through the layers using the transpose of the synaptic weights. In the case of DECOLLE, it uses local losses per layer, calculating an independent loss function per each. On ETLP, the temporal dependency is solved by means of using eligibility traces but needs to resolve the spatial dependency between layers (green flow). On DRTP and ETLP, the error is only calculated at the output layer, the weights in the hidden layer are updated using directly the target, using a random fixed matrix.

the surrogate gradient of the post-synaptic voltage ψ and (3) the threshold adaptation trace $a(t)$ at time step t, θ is the increase in the adaptive threshold, and γ is the threshold decay constant,

$$\begin{aligned} \varepsilon_{\text{adapt}}(t) &= \varepsilon_{\text{pre}}(t)\,\varphi(t) + (\gamma - \varphi(t)\,\theta)\,\varepsilon_{\text{adapt}}(t-1) \\ e(t) &= \varphi(t)\left(\varepsilon_{\text{pre}}(t) - \theta\varepsilon_{\text{adapt}}(t)\right). \end{aligned} \tag{6}$$

2.3. Spatial dependency

The back-propagation of the errors in calculating the network gradients means that there is a spatial dependency in training. This dependency results in non-local update to each neuron or synapse for two main reasons: (1) weight update locking and (2) weight transport problem as explained in section 1).

DRTP solves this problem by using one-hot-encoded targets in supervised classification problems as a proxy for the sign of the error [21]. Therefore, instead of waiting for the data to propagate forward through the layers to calculate the global error which would be then back-propagated, the targets (i.e. labels) are used directly (figure 2). Together with feedback alignment using fixed random weights from the teaching neurons in the gradient computation, DRTP solves both spatial dependency problems (i.e. update locking and weight transport) [21]. Since there is no propagation of gradient information between layers, the gradient computation is performed in a similar way in each layer, so the computational graph (figure 1) is the same across layers. The difference is that, instead of using the gradient of the error, the projected labels are used. As a result, in contrast with eProp, there is no need for a synaptic eligibility trace which we define as a low-pass filter of the Hebbian component (i.e. pre- and pos-synaptic information) of the learning rule [23].

2.4. ETLP

ETLP is based on a many-to-one learning architecture [64], where the error is calculated at a specific time step of an input streaming data (figure 1), that can be defined as a random point in an input sequence of spikes, as defined in section 2.2. Learning is made using target labels with one-hot encoding, representing the targets as teaching neurons. Since the targets are represented as a one-hot-encoded vector, and the learning algorithm is updated using a many-to-one architecture (as exposed in section 2.2), they can be represented as teaching neurons that fire an event at the end of a time window. The teaching neurons have synapses that connect them to the output neurons and the neurons in the hidden layers (figure 3), firing at a certain frequency (e.g. 100 Hz), obtaining a fully event-based supervised learning technique.

2.4.1. Synapse traces

In the case of an ALIF neuron, each synapse requires two traces: (1) the pre-synaptic spike trace (equation (7)) which holds the pre-synaptic activity information, where $I(t)$ represent the input spike and α the neuron time constant, and (2) the threshold adaptation trace (equation (9)) which holds the threshold value depending on the post-synaptic activity. In the case of a LIF, only a trace of the pre-synaptic spikes is

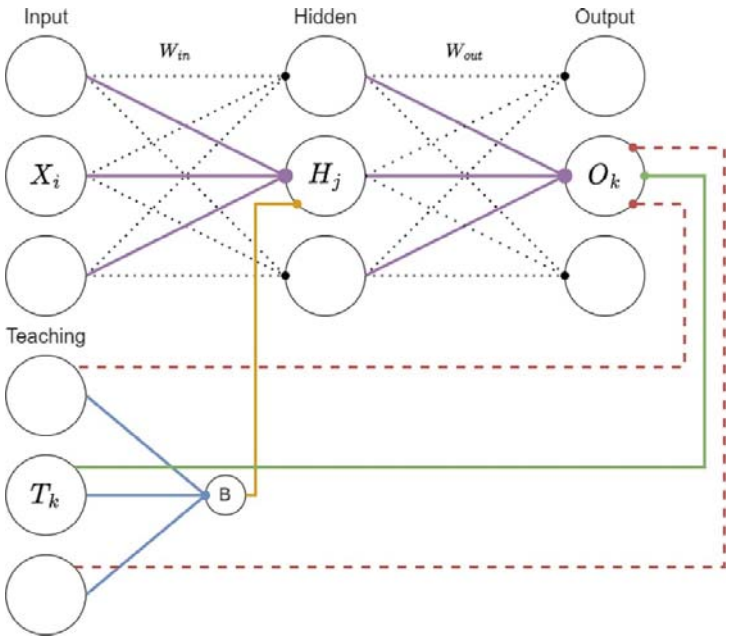


Figure 3. Feedforward SNN with ETLP learning. The network consists of a hidden layer, an output layer, and teaching neurons. The output layer is connected to the teaching neurons by both excitatory (green) and inhibitory (dashed red) synapses. The learning layer is also connected to the group of recurrent neurons via a random weight matrix B. When a class is active, the neuron in the learning layer starts to emit spikes at a certain frequency (e.g. 100 Hz), thus modifying the synapses of the neurons according to the equations (11) and (12).

required. Finally, $e(t)$ in equation (10) consists of a combination of the pre-synaptic spike trace, the surrogate gradient of the post-synaptic voltage (equation (8)), and the threshold adaptation trace when using the ALIF neuron,

$$\varepsilon_{\text{pre}}(t) = \alpha\varepsilon_{\text{pre}}(t-1) + I(t) \tag{7}$$

$$\varphi(t) = \gamma\max(0, 1 - |V(t) - A(t)|) \tag{8}$$

$$\varepsilon_{\text{adapt}}(t) = \varepsilon_{\text{pre}}(t)\varphi(t) + (\gamma - \varphi(t)\theta)\varepsilon_{\text{adapt}}(t-1) \tag{9}$$

$$e(t) = \varphi(t)\left(\varepsilon_{\text{pre}}(t) - \theta\varepsilon_{\text{adapt}}(t)\right). \tag{10}$$

Synapses update is event-driven, i.e. each time the postsynaptic (hidden or output) neuron receives an input spike from a master neuron. In the case of hidden layers, the update is performed based on the equation (11), where I represents the input from the teaching neurons and e the product of surrogate gradient and pre-synaptic and threshold adaptation traces. Thus, by using only local information per synapse (pre-synaptic and post-synaptic values), as well as a firing signal from a neuron, learning only takes place when an activation signal is received from the learning neurons.

2.4.2. Synaptic plasticity

The synapse update is event-driven, i.e. each time the post-synaptic neuron (hidden or output) receives a spike from a teaching neuron. In the case of the hidden layers, the update is made based on equation (11), where I represents the input from the teaching neurons. Thus, ETLP learning uses local information per synapse (pre-synaptic spike trace, surrogate gradient of the post-synaptic voltage, and teaching neurons weights) and is triggered when a spike from a teaching neuron is received. Otherwise, the weights of the network are not updated. For the output layer, two different types of connections are made between the teaching neurons and the output layer. Each teaching neuron is connected in a one-to-one fashion with an excitatory synapse with the output neurons and with the other output neurons with an inhibitory synapse. As a result, the output neuron receives an input current I that each time it receives an input spike from the teaching neuron, applying equation (12), where e_{out} is the function over the pre-synaptic spike trace, the surrogate gradient of the post-synaptic voltage and the threshold adaptation trace of the output neuron, and s_{out} is the spike of the output neuron,

$$W(t+1)_{\text{in/rec}} = W(t)_{\text{in/rec}} + \gamma I(t)\, e(t)_{\text{in/rec}} \tag{11}$$

Algorithm 1. ETLP training for LIF neuron.

for $t = 1, \ldots, T$ **do**
  for $l = 1, \ldots, L$ **do**
    $s_l^t \leftarrow LIF_l(s_{l-1}^t)$
    $\varepsilon_l^t \leftarrow \alpha\varepsilon_l^{t-1} + s_{l-1}^t$
    if *teachSig* **then** $\rightarrow$ *learning mode*
      $\varphi_l^t \leftarrow \gamma max\left(0, 1 - \left|V_l^t - Vth_l^t\right|\right)$
      $\epsilon_l^t \leftarrow \varepsilon_l^t \varphi_l^t$
      if $l = L$ **then**
        $W_l^t \leftarrow W_l^{t-1} + \lambda(s_l^t - y^*)\epsilon_l^t$
      else
        $W_l^t \leftarrow W_l^{t-1} + \lambda y^* F_l \epsilon_l^t$
      end If
    else
      $\rightarrow$ *inference mode*
    end If
  end for
end for

$$W(t+1)_{\text{out}} = W(t)_{\text{out}} + \gamma \frac{2s(t)_{\text{out}} - I(t) - 1}{2} e(t)_{\text{out}}. \tag{12}$$

2.4.3. Training

The training procedure of ETLP is described on algorithm 1. At each timestep, when a teacher signal from the teacher neurons is available (i.e. learning mode), a teacher spike train is produced based on a Poisson distribution with a specific frequency and triggers the online learning. The pre-synaptic trace is calculated and, in case the learning is triggered, the eligibility trace is calculated and the synaptic weight is updated as described in the algorithm 1, where φ_l^t represents the pre-synaptic trace, φ_l^t the surrogate gradient, ϵ_l^t the eligibility trace and F_l the fixed random weight that connects the target with the layer l. When no teacher signal is available, no weight update is performed (i.e. inference mode).

3. Experiments and results

ETLP is designed for online learning in edge pattern recognition applications where there is a need for adaptation to new classes and/or new users. In the real-world scenario, a SNN will be pre-trained to some initial performance at initial deployment on the neuromorphic chip, and ETLP will be automatically triggered for on-the-fly adaptation to new patterns when a teaching signal is provided by the user. Weights adaptation can occur in the last few classification layers, while feature extraction layers can be frozen [57]. Nevertheless, in order to benchmark and compare ETLP with other learning rules, we simplify the problem and train a relatively small fully-connected (feedforward and recurrent) SNN which is equivalent to the classification layers in the real-world scenario, then we quantify its accuracy when learning from scratch as a proof of convergence.

To test the performance of ETLP, the rule was compared with BPTT (non-local in time and space), eProp (local in time but non-local in space), and DECOLLE (uses local losses) on two different datasets. The final accuracy calculation is based on a rate decoding (i.e. the winning neuron is the one firing most during the sample presentation) of the final layer's output spikes.

The first dataset is neuromorphic-MNIST (N-MNIST) [46], recorded by moving the ATIS sensor over MNIST samples (10 classes) in three saccades of 100 ms each. The resulting information is therefore mainly spatial, where the temporal structure is exclusively related to the movements of the sensor. The second dataset is spiking Heidelberg digits (SHD) [11], a set of 10 000 spoken digit audios from 0 to 9 in both English and German (20 classes). The dataset was generated using an artificial cochlea, producing spikes in 700 input channels. In contrast with N-MNIST, SHD contains both a spatial and a rich temporal structure [49].

For N-MNIST, we used only the first saccade of about 100 ms with a time step of 1 ms, resulting in 100 time steps per sample. For SHD, we limited the samples duration to 1 s with a time step of 10 ms to result in 100 time steps per sample, similar to N-MNIST. This time binning provides enough temporal information to reach a state-of-the-art performance in accuracy [5].

The hyper-parameters used on the simulations for both datasets are shown in table 1, where dt is the algorithmic timestep, lr the learning rate, τ_{m} the membrane decay time constant, τ_{a} the threshold adaptation

Table 1. Hyperparameter used on SHD and N-MNIST datasets.

Parameter	N-MNIST	SHD
dt	1 ms	10 ms
Learning rate	$5e^{-4}$	$5e^{-4}$
τ_m	80 ms	1000 ms
τ_a	10 ms	1000 ms
ν_{th}	1	1
Batch size	128	128
Hidden size	200	450
Refractory period (time steps)	5	5

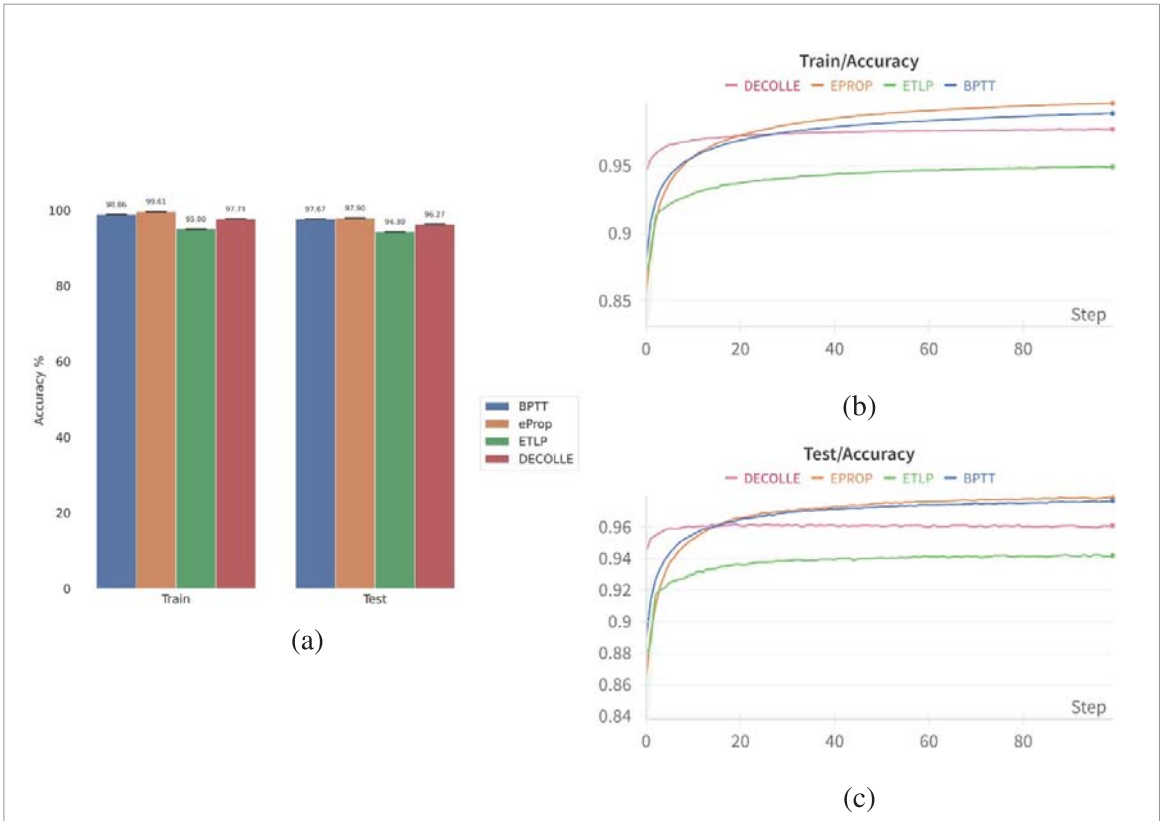


Figure 4. Accuracy of BPTT, eProp, DECOLLE and ETLP on N-MNIST using a feedforward network of LIF neurons. Figure (a) shows a comparison in accuracy between the training and test datasets, and figures (b) and (c) show the accuracy over all the epochs for training and test, respectively.

decay time constant, ν_{th} the base threshold. The PyTorch software implementation is publicly available on GitHub (https://github.com/ferqui/ETLP). All the results presented in this section are averaged over three runs with different random initialization of the synaptic weights.

3.1. N-MNIST

For the N-MNIST dataset, a feedforward network of LIF neurons with 2064 input neurons ($32 \times 32 = 1032$ neurons for each polarity), 200 hidden neurons, and 10 output neurons were used.

Indeed, explicit recurrence is not needed when the information in the data are mostly spatial [5].

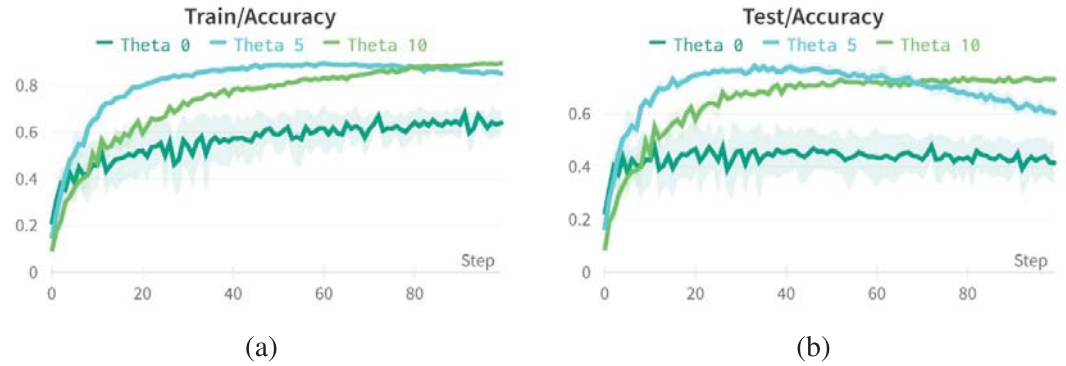
The results on N-MNIST dataset for training and test sets are shown in figure 4 ETLP shows an accuracy of 94.30% on the test set, compared to 97.90%, 97.67% and 96.27% for eProp, BPTT and DECOLLE respectively. BPTT results are on par with the state-of-the-art with a similar topology [5, 49], while eProp reaches a slightly higher accuracy which is probably due to hyper-parameters. On the other hand, ETLP loses 3.60% compared to eProp and 1.97% compared to DECOLLE.

3.2. SHD

For the SHD dataset, since it has more temporal dependency than N-MNIST [49], we tested both feedforward and recurrent networks with all-to-all explicit recurrent connections in the hidden layer. Both

Table 2. Test accuracy of θ parameter exploration on BPTT, eProp, DECOLLE and ETLP. On SHD dataset with a recurrent spiking neural network (RSNN).

	$\theta = 0$		$\theta = 5$		$\theta = 10$	
Learning method	μ	σ	μ	σ	μ	σ
BPTT	75.23	0.44	64.43	0.68	23.96	0.67
eProp	51.68	1.10	80.79	0.39	49.94	1.98
DECOLLE	45,91	2,25	62,01	0,61	60,66	1,68
ETLP	48.53	0.64	78.71	1.49	74.59	0.44



(a) (b)

Figure 5. Accuracy for different threshold adaptation on ETLP in SHD dataset with the recurrent network.

networks used 700 input neurons, 450 hidden neurons and 20 output neurons. Two different neuron models were explored: LIF and ALIF.

For the feedforward topology, a LIF model was used. ETLP has an accuracy of 59.19%, compared to 66.33%, 63.04%, and 58.55% for BPTT, eProp, and DECOLLE respectively, as shown in figure 6.

For the recurrent topology, an initial exploration over the θ parameter of the adaptive threshold has been made. Several simulations have been performed for each rule and for each θ parameter (0, 5, and 10) to see how it affects the accuracy. Table 2 shows the accuracy for each learning rule and θ parameter, where the θ value used for the final results is marked in green.

In the case of ETLP, a higher maximum accuracy is obtained with $\theta = 5$. However, it can be seen in figure 5 that this accuracy then decreases with the epochs, going below the accuracy with $\theta = 10$. Hence, $\theta = 10$ is taken as the best value because it is more important to have a convergence with more data than to reach a maximum accuracy at a particular epoch.

The final results obtained on SHD dataset using a recurrent topology are shown in figure 6 based on the best θ parameter obtained per learning rule. ETLP ($\theta = 10$) achieves a test accuracy of 74.59%, showing a slight loss compared to BPTT ($\theta = 0$) that reaches 75.23% and a consequent loss of 6.20% compared to eProp that reaches 80.79%. However, it has an increase of 12.58% compared to DECOLLE. We observe that adding explicit recurrence in the hidden layer of the network increases accuracy by a considerable amount of 8.90% for BPTT, 17.75% for eProp, 3.46% for DECOLLE, and 15.40% for ETLP. We also observe that the BPTT offline learning reaches a better accuracy without threshold adaptation (i.e. $\theta = 0$), while it is required for the online learning of eProp (although spatially non-local), DECOLLE, and ETLP.

It is to note that, similar to [49], we did not reach the state-of-the-art accuracy with BPTT that reaches 80.41% with a similar topology [5]. This can be explained by the different training strategies where the output layer in [5] is not spiking, as well as the chosen hyper-parameters that were not tuned specifically for BPTT. Hence, BPTT reaches in principle a better accuracy than both eProp and ETLP.

ETLP proposes a method for local training of multi-layer SNNs, i.e. not only the output layer but also the hidden layers. To validate the credit assignment of ETLP for hidden layers, we performed an experiment on the SHD dataset where our network was trained with both a shallow (only the output layer's weights are updated) and a standard (both the output and hidden layers weights are updated simultaneously) strategy. The results show an increase in accuracy of 43% in the test set when all the weights are updated using ETLP, as shown in figure 7.

To test the application of ETLP on deeper networks with different parameters, a hyper-parameter optimization (HPO) was made for networks between 0 and 10 layers for the N-MNIST and SHD datasets

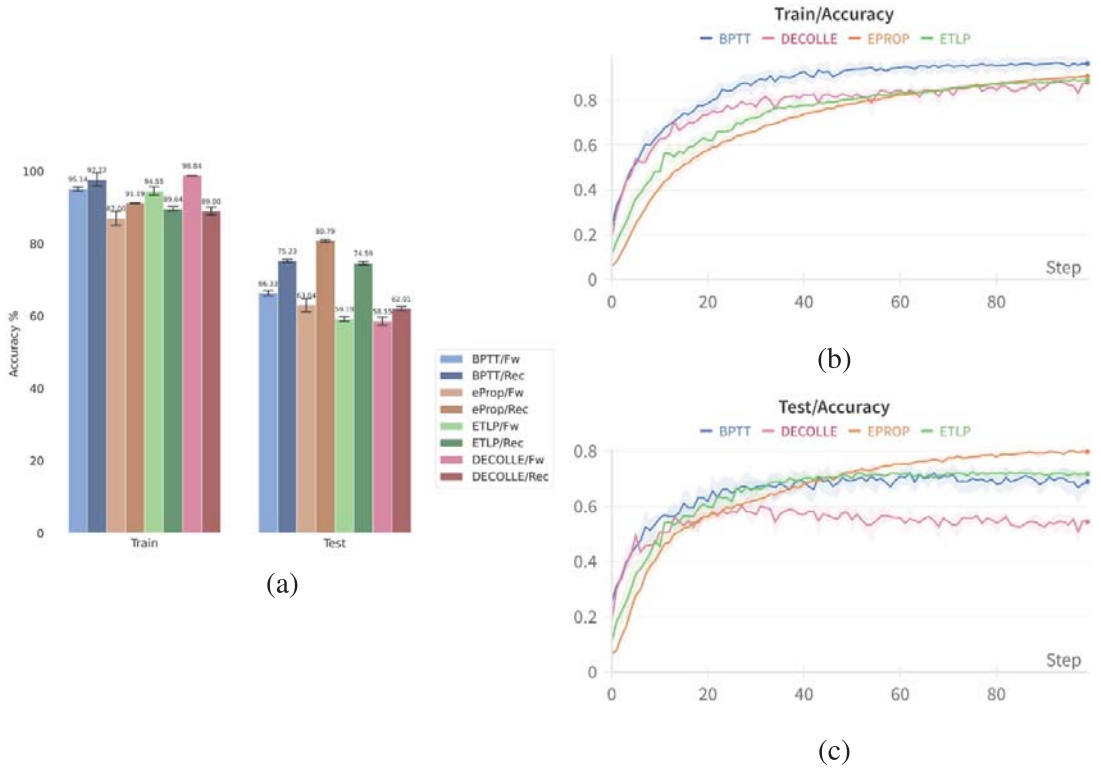


Figure 6. Accuracy of BPTT, eProp, DECOLLE, and ETLP on SHD. Figure (a) shows a comparison in accuracy between the feedforward network with LIF neurons and the recurrent network with ALIF neurons. Figures (b) and (c) shows the accuracy of the recurrent network for each learning rule on the training and test datasets, respectively, with the best θ value obtained for each one.

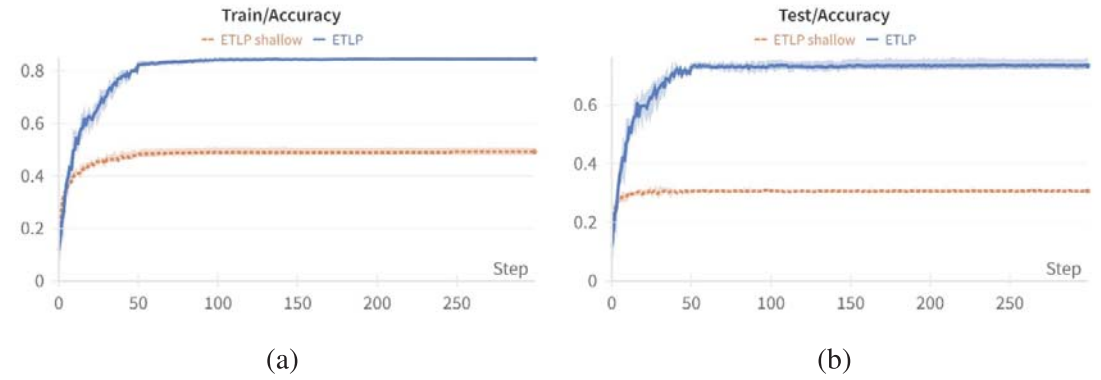


Figure 7. Accuracy obtained using ETLP training on the output layer only (ETLP shallow, dashed line) or both the output and hidden layers (ETLP, continuous line) in a two-layer network.

using feedforward LIF neurons. During HPO the optimal parameter set found (table 3) was tested on the test set with different initialization seeds. The results for the N-MNIST dataset are shown in figure 8.

4. Hardware implementation

ETLP could be easily implemented on a neuromorphic hardware, resulting in a low-latency and low-power consumption alternative for online learning. [20] offers a similar approach, where an on-chip eProp algorithm for learning has been implemented. In this paper, as a proof of concept, a field-programmable gate array (FPGA) implementation of the ETLP module has been designed to calculate the weight update based on [51], as shown in figure 9. The module receives the following signals as input for the gradient calculation:

Table 3. Hyperparameters founded for N-MNIST dataset for ETLP.

Parameter	1 layers	2 layers	3 layers	4 layers
Learning rate	$1.333e^{-3}$	$2.94e^{-4}$	$3.71e^{-5}$	$6.336e^{-5}$
τ_m	989.607	172.962	635.850	158.31
τ_o	386.058	995.947	779.788	588.88
Refractory	3	5	2	5
Layer 1 size	X	1024	512	1024
Layer 2 size	X	X	1024	32
Layer 3 size	X	X	X	64
Layer 4 size	X	X	X	X

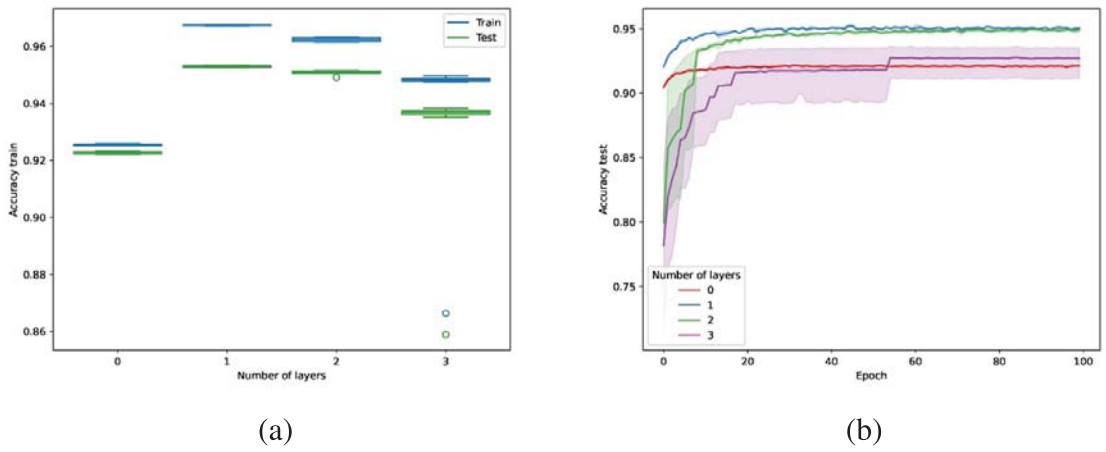


Figure 8. Accuracy obtained using ETLP training for a different number of hidden layers on N-MNIST dataset.

(i) the address of the pre-synaptic neuron that is connected to the synapse to be updated;
(ii) the output of the pre-synaptic neuron (0 or 1) for the pre-synaptic spike trace calculation;
(iii) the voltage value of the post-synaptic neuron for the surrogate voltage calculation;
(iv) the teaching value, is equal to the teaching neurons' weights when they emit a spike which triggers the weight update.

The implemented surrogate gradient shown in figure 10 is the triangular function, defined as $f(x) = \max(0, 1 - |x|)$.

We assume to use some time-multiplexing handled by the upper module but do not go into its architectural details as the ETLP implementation is independent. The ETLP module stores pre-synaptic spike trace (figure 11) values in a BRAM. Each time a step signal is received, the module retrieves the stored trace, calculates the new trace value, and stores it back into memory. At the same time, it calculates the surrogate gradient from the value of the post-synaptic voltage. When both results are obtained, the final gradient is calculated with their product with the signal coming from the teaching neuron. Finally, the module produces a step signal to indicate to the upper module that it can update the corresponding synaptic weight. This module can be optimized for asynchronous updates of the gradient (i.e. only when a spike from the teaching neurons is received).

This module takes three clock cycles to perform the calculation. Figure 12 shows a time diagram of the process. The module would be in the 'initial' state until a signal from *step_I* is received. Then, it gathers the pre-synaptic spike trace from memory ('read' state), calculates the gradient value, and writes the new pre-synaptic spike trace into memory ('write' state). Finally it produces an output signal to notify the upper module that the gradient value is available and changes the state back to 'init'.

The module has been synthesized on a Nexys4 DDR (xc7a100tcsg324-1) FPGA with the resource consumption shown in table 4. A power estimation has been carried out at different clock frequencies. With a 100 MHz clock, the device's static and dynamic power consumption are approximately 97 mW and 53 mW respectively, where 47 mW corresponds to I/O. With a 10 MHz clock, the static power consumption is the same while the dynamic power consumption is reduced to 5 mW, where 4.5 mW corresponds to I/O. Finally, with a 1MHz clock, the dynamic power consumption is negligible, since the value is smaller than the resolution provided by the tool (below 1 mW).

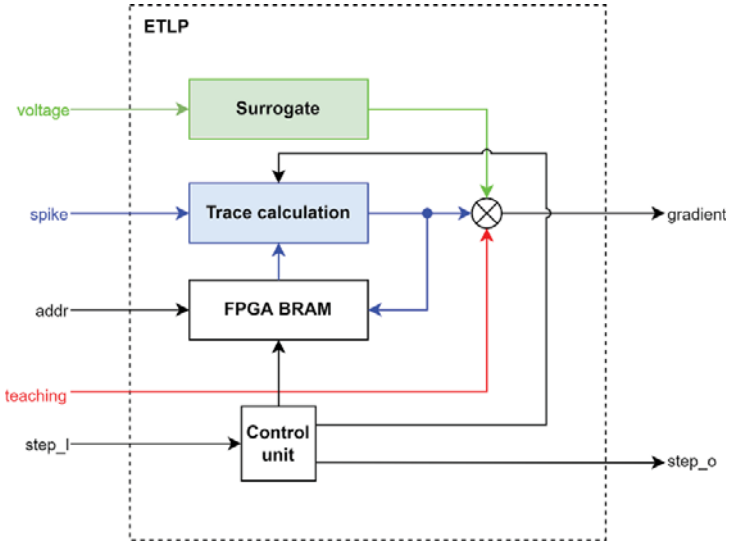


Figure 9. FPGA implementation for ETLP module. The control unit triggers the calculation of the new pre-synaptic spike trace value and stores it in memory. Once the gradient is calculated, it sends the control signal to the upper module so that it can read the calculated gradient value.

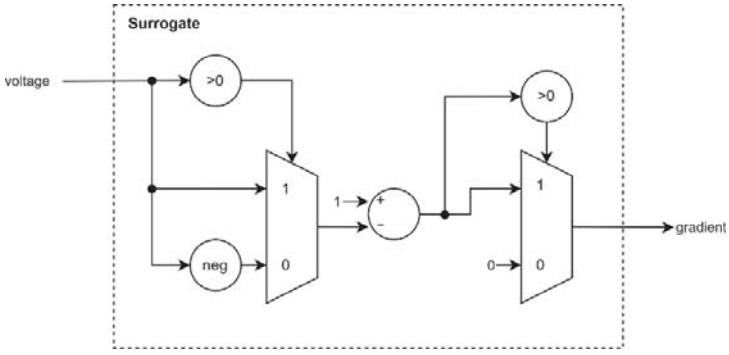


Figure 10. Triangular function surrogate gradient implementation on FPGA.

Interestingly, when using one module per neuron, a 1 MHz clock is sufficient to update a neuron in the hidden layer of our recurrent SNN with SHD, since the number of required clock cycles per second is (700 [synapses from input layer neurons] + 450 [synapses from hidden layer neurons) * 100 [updates per second] * 3 [clock cycles per updates] = 345 000, meaning a clock frequency of 345 KHz. We can therefore expect a low-power consumption for the full network, especially when designing an application-specific integrated circuit (ASIC) with a similar architecture.

5. Discussion and conclusion

In this work, we have proposed the ETLP rule, a local learning rule in time and space that is triggered asynchronously based on the availability of the target or label. These features satisfy the hardware constraints of neuromorphic chips where computing and memory are co-localized. This learning rule then could be implemented into different types of neuromorphic hardware platforms from digital [20, 22, 40, 59] to analog [9, 26, 50], where the restriction of locality and online weights updates are imperative. We compared ETLP to eProp (non-local in space), DECOLLE (uses local losses), and BPTT (non-local in time and space), and achieved a test accuracy of 94.30% on N-MNIST with a feedforward network of LIF neurons and 74.59% on SHD with a recurrent network of ALIF neurons. ETLP has a loss of accuracy of 3.60% and 6.20% compared to the best accuracy of eProp on N-MNIST and SHD, respectively, which is reasonable due to its fully event-driven and local plasticity paradigm. However, DECOLLE seems to perform better than ETLP in spatial patterns, with a difference of 1.97% with respect to ETLP. While ETLP performs better on spatio-temporal patterns with a rich temporal structure [5], with an increase of 12.58% in SHD compared to DECOLLE.

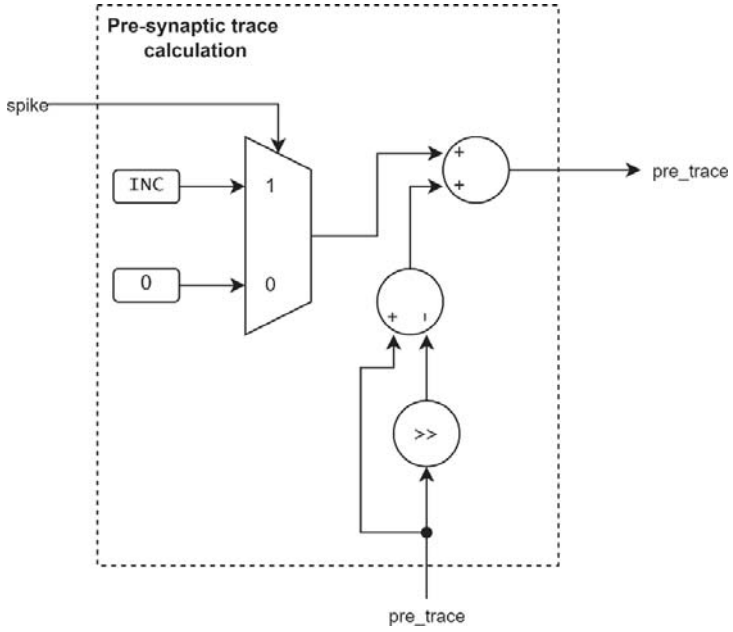


Figure 11. Pre-synaptic spike trace calculation on FPGA. In the case of a pre-synaptic spike arrives a constant is added to the new pre-synaptic spike trace value.

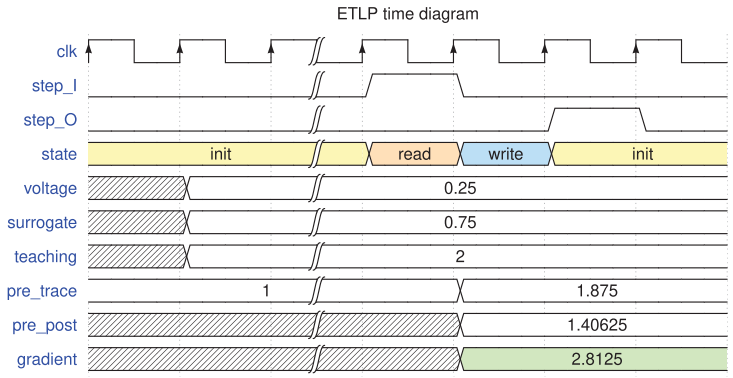


Figure 12. ETLP FPGA implementation time diagram.

Table 4. Resource utilization of ETLP module.

Resource	Used	Available	Util%
Slice LUTs	115	63 400	0.18
Flip Flop	22	126 800	0.02%
DSPs	2	240	0.83%
BRAM tile	1	270	0.37%

Two features have a major impact on the accuracy when learning SHD spatio-temporal patterns that have a rich temporal structure. First, adding an all-to-all explicit recurrence in the hidden layer reduces training over-fitting and highly increases test accuracy by 8.90%, 17.75%, 3.46% and 15.40% for BPTT, eProp, DECOLLE and ETLP, respectively. This confirms the results reported on BPTT in [5]. Second, the threshold adaptation of the ALIF improves the accuracy of ETLP, eProp, and DECOLLE by 26.06%, 29.11%, and 16.1% respectively, while it decreases the accuracy of BPTT by 10.80%. Although we did not extensively explore the reasons behind this behavior, it is clear for ETLP that the threshold adaptation provides an online dynamic mechanism (in addition to the refractory period) to limit the neurons' firing rate. In BPTT, it might not be needed since the error signal is explicitly backpropagated and used in the weights updates. An additional mechanism to explore for ETLP to further reduce the firing rate and potentially gain energy-efficiency [7] is

Table 5. Complexity analysis of the gradient computation over different learning rules with the final accuracy obtained on both datasets. N represents the number of input neurons; M is the number of neurons in a layer; T is the length of the back-propagated sequence; Nr is the number of readout neurons in DECOLLE; p is the ratio of connected neurons/total possible.

Learning	Space	Time	Accuracy N-MNIST	Accuracy SHD
BPTT	$O(NT)$	$O(pNMT)$	97.67%	75.23%
e-Prop	$O(pNM)$	$O(pNM)$	97.9%	80.79%
DECOLLE [28]	$O(1)$	$O(MNr + pNM)$	96.27%	62.01%
ETLP	$O(1)$	$O(pNM)$	94.3%	74.59%

the voltage regularization based on additional plasticity mechanisms such as homeostatic membrane potential-dependent synaptic plasticity [1].

In addition to the accuracy benchmarking, we demonstrated a proof-of-concept hardware implementation on FPGA to show how it is possible to map the local computational primitives of ETLP on digital neuromorphic chips. It is also a way to better understand the simplicity of the plasticity mechanism in hardware, where the required information is two Hebbian pre-synaptic (spike trace) and post-synaptic (voltage put in a simple function) factors [29] and a third factor in the form of an external signal provided by the target or label. Therefore, ETLP is compatible with neuromorphic chips such as Loihi2 which supports three-factor local plasticity on the chip [13, 45]. Table 5 shows a comparison of the computational complexity of BPTT, eProp, DECOLLE [28] and ETLP. We can see that ETLP is indeed local in space like DECOLLE (ETLP uses DRTP while DECOLLE uses local losses) and local in time like eProp (although ETLP needs one spike trace per pre-synaptic neuron only, while eProp needs in addition an eligibility trace per synapse), thus being more computationally efficient than previously proposed plasticity mechanisms.

ETLP builds on previous developments toward more local learning mechanisms and proposes a fully local synaptic plasticity model that fits the hardware constraints of neuromorphic chips for online learning with real-time interaction and low-energy consumption. A good classification accuracy on visual and auditory benchmarks is shown, both starting from a randomly initialized network to show a proof of convergence. ETLP was proven also to train not only the output layer but also the hidden layer of the network with an increase of accuracy of 43% on the auditory dataset, highlighting the effective credit assignment despite the local learning paradigm. ETLP has also been proven to work on multiple hidden layers on the visual dataset. ETLP converges with multiple hidden layers and reaches the best accuracy with a two layers network. It shows that ETLP makes use of the hidden layer to improve accuracy compared to a network without a hidden layer or with a random hidden layer as shown in figures 7 and 8, proving a successful spatial credit assignment despite the lack of backpropagation. Nevertheless, more hidden layers do not improve accuracy with the actual training hyper-parameters (e.g. number of epochs).

Future works will focus on implementing ETLP on a more realistic experimental setup with few-shot learning using convolutional SNNs. To adapt to new classes that have not been seen before, new labels will be provided to the network to automatically trigger plasticity in the plastic synapses. One possibility is to use standard spiking convolutional layers with weight sharing (i.e, weight kernels are shared across locations in the input space) to be selective to the same feature [28, 30], however, that would break the locality of the rule. A time-multiplexed implementation in digital neuromorphic hardware could alleviate the problem, nevertheless increasing the processing and learning latency. When targeting fully parallel implementations, for example in analog neuromorphic hardware [25, 26], a local processing paradigm [29] can be achieved by using locally connected layers where weights are not shared (i.e. a different weight kernel is learned per input location in the input space) [53]. Alternatively, as discussed previously, only the last classification (i.e. fully-connected) layers can be plastic, while feature extraction layers can remain fixed, under the assumption that the features are similar across different classes [57].

An open question is how to design the output layer, either by having more neurons than needed at the beginning or by adding new neurons when needed. Since few-shot learning is based on very few labeled samples, we can train both initial network synaptic weights and ETLP hyper-parameters based on the model-agnostic meta-learning that has been recently applied to SNNs [56].

Data availability statement

No new data were created or analysed in this study.

Acknowledgment

Fernando M Quintana would like to acknowledge the Spanish Ministerio de Universidades for the support trhough FPU Grant (FPU18/04321). This work was part of the project Nemovision PID2019-109465RB-I00 funded by MICIU/AEI /10.13039/501100011033, and (with support from the European Regional Development Fund) MIND-ROB (PID2019-105556GB-C33) from the Ministerio de Ciencia e Innovación, the CogniGron research center and the Ubbo Emmius Funds of the University of Groningen.

ORCID iDs

Fernando M Quintana https://orcid.org/0000-0001-5042-9399
Fernando Perez-Peña https://orcid.org/0000-0003-3586-2930
Pedro L Galindo https://orcid.org/0000-0003-0892-8113
Emre O Neftci https://orcid.org/0000-0002-0332-3273
Elisabetta Chicca https://orcid.org/0000-0002-5518-8990
Lyes Khacef https://orcid.org/0000-0002-4009-174X

References

[1] Albers C, Westkott M and Pawelzik K 2016 Learning of precise spike times with homeostatic membrane potential dependent synaptic plasticity *PLoS One* **11** 1–28
[2] Basu A, Deng L, Frenkel C and Zhang X 2022 Spiking neural network integrated circuits: a review of trends and future directions *IEEE Custom Integrated Circuits Conf. (CICC)* pp 1–8
[3] Bellec G, Scherr F, Subramoney A, Hajek E, Salaj D, Legenstein R and Maass W 2020 A solution to the learning dilemma for recurrent networks of spiking neurons *Nat. Commun.* **11** 3625
[4] Bengio Y, Lee D H, Bornschein J and Lin Z 2015 Towards biologically plausible deep learning (arXiv:1502.04156)
[5] Bouanane M S, Cherifi D, Chicca E and Khacef L 2023 Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition *Front. Neurosci.* **17 1244675**
[6] Boybat I, Gallo M, Nandakumar S R, Moraitis T, Parnell T, Tuma T, Rajendran B, Leblebici Y, Sebastian A and Eleftheriou E 2018 Neuromorphic computing with multi-memristive synapses *Nat. Commun.* **9** 2514
[7] Caccavella C, Paredes-Vallés F, Cannici M and Khacef L 2023 Low-power event-based face detection with asynchronous neuromorphic hardware (arXiv:2312.14261 [cs.NE])
[8] Ceolini E, Frenkel C, Shrestha S B, Taverni G, Khacef L, Payvand M and Donati E 2020 Hand-gesture recognition based on emg and event-based camera sensor fusion: a benchmark in neuromorphic computing *Front. Neurosci.* **14** 637
[9] Chicca E, Stefanini F, Bartolozzi C and Indiveri G 2014 Neuromorphic electronic circuits for building autonomous cognitive systems *Proc. IEEE* **102** 1367–88
[10] Christensen D V *et al* 2022 Roadmap on neuromorphic computing and engineering *Neuromorph. Comput. Eng.* **2** 022501
[11] Cramer B, Stradmann Y, Schemmel J and Zenke F 2020 The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **33** 2744–57
[12] Czarnecki W M, Swirszcz G, Jaderberg M, Osindero S and Vinyals K 2017 Understanding synthetic gradients and decoupled neural interfaces *Proc. 34th Int. Conf. on Machine Learning - Volume 70* (JMLR.org) pp 904–12
[13] Davies M *et al* 2018 Loihi: a neuromorphic manycore processor with on-chip learning *IEEE Micro* **38** 82–99
[14] Davies M, Wild A, Orchard G, Sandamirskaya Y, Guerra G A F, Joshi P, Plank P and Risbud S R 2021 Advancing neuromorphic computing with loihi: a survey of results and outlook *Proc. IEEE* **109** 911–34
[15] Dennard R H, Rideout V L, Bassous E and LeBlanc A R 1974 Design of ion-implanted MOSFET's with very small physical dimensions *IEEE J. Solid-State Circuits* **9** 256–68
[16] Dennard R, Gaensslen F, HWA-NIEN Y U, Leo V R, Bassous E and Leblanc A 2007 Design of ion-implanted MOSFET's with very small physical dimensions *IEEE Solid-State Circuits Soc. Newsl.* **12** 38–50
[17] Eshraghian J K, Ward M, Neftci E, Wang X, Lenz G, Dwivedi G, Bennamoun M, Jeong D S and Lu W D 2021 Training spiking neural networks using lessons from deep learning (arXiv:2109.12894 [cs.NE])
[18] Ezra E 2021 *Neuromorphic Engineering: The Scientist's, Algorithm Designer's and Computer Architect's Perspectives on Brain-Inspired Computing* (CRC Press)
[19] Frenkel C, Bol D and Indiveri G 2021 Bottom-up and top-down neural processing systems design: neuromorphic intelligence as the convergence of natural and artificial intelligence (arXiv:2106.01288 [cs.NE])
[20] Frenkel C and Indiveri G 2022 ReckOn: a 28nm Sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales *IEEE Int. Solid- State Circuits Conf. (ISSCC)* vol 65 pp 1–3
[21] Frenkel C, Lefebvre M and Bol D 2021 Learning without feedback: fixed random learning signals allow for feedforward training of deep neural networks *Front. Neurosci.* **15** 20
[22] Frenkel C, Lefebvre M, Legat J -D and Bol D 2019 A 0.086-mm^2 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS *IEEE Trans. Biomed. Circuits Syst.* **13** 145–58
[23] Gerstner W, Lehmann M, Liakoni V, Corneil D and Brea J 2018 Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules *Front. Neural Circuits* **12** 53
[24] Gerstner W, Ritz R and van Hemmen J L 1993 Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns *Biol. Cybern.* **69** 503–15
[25] Hazan A and Ezra T E 2022 Neuromorphic neural engineering framework-inspired online continuous learning with analog circuitry *Appl. Sci.* **12** 4528
[26] Indiveri G *et al* 2011 Neuromorphic silicon neuron circuits *Front. Neurosci.* **5** 1–23
[27] Izhikevich E M 2007 Solving the distal reward problem through linkage of STDP and dopamine signaling *Cereb. Cortex* **17** 2443–52

[28] Kaiser J, Mostafa H and Neftci E 2020 Synaptic plasticity dynamics for deep continuous local learning (DECOLLE) *Front. Neurosci.* **14** 424
[29] Khacef L, Klein P, Cartiglia M, Rubino A, Indiveri G and Chicca E 2023 Spike-based local synaptic plasticity: a survey of computational models and neuromorphic circuits *Neuromorph. Comput. Eng.* **3** 042001
[30] Kheradpisheh S R, Ganjtabesh M, Thorpe S J and Masquelier T 2018 STDP-based spiking deep convolutional neural networks for object recognition *Neural Netw.* **99** 56–67
[31] LeCun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-based learning applied to document recognition *Proc. IEEE* **86** 2278–324
[32] LeCun Y *et al* 2015 Deep learning *Nature* **521** 436–44
[33] Lillicrap T P, Cownden D, Tweed D B and Akerman C 2016 Random synaptic feedback weights support error backpropagation for deep learning *Nat. Commun.* **7** 13276
[34] MacNeil D and Eliasmith C 2011 Fine-tuning and the stability of recurrent neural networks *PLoS One* **6** 1–16
[35] Markram H, Helm P J and Sakmann B 1995 Dendritic calcium transients evoked by single back-propagating action potentials in rat neocortical pyramidal neurons *J. Physiol.* **485** 1–20
[36] McNaughton B L, Douglas R M and Goddard G V 1978 Synaptic enhancement in fascia dentata: cooperativity among coactive afferents *Brain Res.* **157** 277–93
[37] Mead C and Conway L 1980 *Introduction to VLSI Systems* (Addison-Wesley)
[38] Morrison A, Diesmann M and Gerstner W 2008 Phenomenological models of synaptic plasticity based on spike timing *Biol. Cybern.* **98** 459–78
[39] Mostafa H, Ramesh V and Cauwenberghs G 2018 Deep supervised learning using local errors *Front. Neurosci.* **12** 608
[40] Muliukov A R, Rodriguez L, Miramond B, Khacef L, Schmidt J, Berthet Q and Upegui A 2022 A unified software/hardware scalable architecture for brain-inspired computing based on self-organizing neural models *Front. Neurosci.* **16 825879**
[41] Muller-Cleve S F *et al* 2022 Braille letter reading: a benchmark for spatio-temporal pattern recognition on neuromorphic hardware (arXiv:2205.15864)
[42] Neftci E O, Augustine C, Paul S and Detorakis G 2017 Event-driven random back-propagation: enabling neuromorphic deep learning machines *Front. Neurosci.* **11** 324
[43] Neftci E O, Mostafa H and Zenke F 2019 Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks *IEEE Signal Process. Mag.* **36** 51–63
[44] Nøkland A 2016 Direct feedback alignment provides learning in deep neural networks *Advances in Neural Information Processing Systems* vol 29, ed D Lee M Sugiyama U Luxburg I Guyon R Garnett (Curran Associates, Inc.)
[45] Orchard G, Frady E P, Rubin D B D, Sanborn S, Shrestha S B, Sommer F T and Davies M 2021 Efficient neuromorphic signal processing with Loihi 2 *CoRR* (arXiv:2111.03746)
[46] Orchard G, Jayawant A, Cohen G K and Thakor N 2015 Converting static image datasets to spiking neuromorphic datasets using saccades *Front. Neurosci.* **9** 437
[47] Ororbia A G 2023 Brain-inspired machine intelligence: a survey of neurobiologically-plausible credit assignment (arXiv:2312.09257 [cs.NE])
[48] Payvand M, Moro F, Nomura K, Dalgaty T, Vianello E, Nishi Y and Indiveri G 2022 Self-organization of an inhomogeneous memristive hardware for sequence learning *Nat. Commun.* **13** 5793
[49] Perez-Nieves N, Leung V, Dragotti P and Goodman D 2021 Neural heterogeneity promotes robust learning *Nat. Commun.***12** 5791
[50] Qiao N, Mostafa H, Corradi F, Osswald M, Stefanini F, Sumislawska D and Indiveri G 2015 A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128 K synapses *Front. Neurosci.* **9 141**
[51] Quintana F M, Perez-Peña F and Galindo P L 2022 Bio-plausible digital implementation of a reward modulated STDP synapse *Neural Comput. Appl.* **34** 15649–60
[52] Rabaey J M *et al* 2019 AI at the edge - a roadmap *Technical Report* IMEC, KU Leuven, Ghent University, VUB, EPFL, ETH Zurich and UC Berkeley
[53] Saunders D J, Patel D, Hazan H, Siegelmann H T and Kozma R 2019 Locally connected spiking neural networks for unsupervised feature learning *Neural Netw.* **119** 332–40
[54] Schuman C D, Potok T E, Patton R M, Birdwell J D, Dean M E, Rose G S and Plank J S 2017 A survey of neuromorphic computing and neural networks in hardware (arXiv:1705.06963) pp 1–88
[55] Shalf J 2020 The future of computing beyond Moore's Law *Phil. Trans. R. Soc.* A **378** 20190061
[56] Stewart K and Neftci E 2022 Meta-learning spiking neural networks with surrogate gradient descent *CoRR* (arXiv:2201.10777)
[57] Stewart K, Orchard G, Shrestha S B and Neftci E 2020 Online few-shot gesture learning on a neuromorphic processor *IEEE J. Emerg. Sel. Top. Circuits Syst.* **10** 512–21
[58] Stuart G and Sakmann B 1994 Active propagation of somatic action potentials into neocortical pyramidal cell dendrites *Nature* **367** 69–72
[59] Stuijt J, Sifalakis M, Yousefzadeh A and Corradi F 2021 μ brain: an event-driven and fully synthesizable architecture for spiking neural networks *Front. Neurosci.* **15 664208**
[60] Thompson N C, Greenewald K, Lee K and Manso G F 2021 Deep learning's diminishing returns: the cost of improvement is becoming unsustainable *IEEE Spectr.* **58** 50–55
[61] Thompson N C, Greenewald K H, Lee K and Manso G F 2020 The computational limits of deep learning *CoRR* (arXiv: 2007.05558)
[62] Yik J *et al* 2024 NeuroBench: a framework for benchmarking neuromorphic computing algorithms and systems (arXiv: 2304.04640 [cs.AI])
[63] Zenke F and Ganguli S 2018 SuperSpike: supervised learning in multilayer spiking neural networks *Neural Comput.* **30** 1514–41
[64] Zenke F and Neftci E O 2021 Brain-inspired learning on neuromorphic substrates *Proc. IEEE* **109** 935–50