A feedback control optimizer for online and hardware-aware training of Spiking Neural Networks

Matteo Saponati*, Chiara De Luca*†, Giacomo Indiveri*, *Senior Member, IEEE*, and Benjamin Grewe*‡
*Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland
†Digital Society Initiative, University of Zurich, Switzerland
‡AI Center, ETH, Zürich, Switzerland

2025 Neuro Inspired Computational Elements (NICE) | 979-8-3315-0302-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/NICE65350.2025.11065428

***Abstract*—Unlike traditional artificial neural networks (ANNs), biological neuronal networks solve complex cognitive tasks with sparse neuronal activity, recurrent connections, and local learning rules. These mechanisms serve as design principles in Neuromorphic computing, which addresses the critical challenge of energy consumption in modern computing. However, most mixed-signal neuromorphic devices rely on semi- or unsupervised learning rules, which are ineffective for optimizing hardware in supervised learning tasks. This lack of scalable solutions for on-chip learning restricts the potential of mixed-signal devices to enable sustainable, intelligent edge systems. To address these challenges, we present a novel learning algorithm for Spiking Neural Networks (SNNs) on mixed-signal devices that integrates spike-based weight updates with feedback control signals. In our framework, a spiking controller generates feedback signals to guide SNN activity and drive weight updates, enabling scalable and local on-chip learning. We first evaluate the algorithm on various classification tasks, demonstrating that single-layer SNNs trained with feedback control achieve performance comparable to artificial neural networks (ANNs). We then assess its implementation on mixed-signal neuromorphic devices by testing network performance in continuous online learning scenarios and evaluating resilience to hyperparameter mismatches. Our results show that the feedback control optimizer is compatible with neuromorphic applications, advancing the potential for scalable, on-chip learning solutions in edge applications.**

***Index Terms*—online learning, spiking neural networks, control theory, neuromorphic computing, mixed-signal devices**

I. Introduction

Nowadays, deep ANNs have become a cornerstone of modern artificial intelligence [1]. This has been driven by advancements in hardware acceleration (i.e., GPUs, TPUs, and NPUs) and intensive training procedures with backpropagation of error [2] over hundreds to thousands of computing machines. While backpropagation allows for massive parallelization of training, it also requires intensive memory access by (1) storing and retrieving intermediate state variables for automatic differentiation, (2) encoding error information with separate states, and (3) updating every parameter during every backward pass [3, 4]. This approach makes ANN ideal for tasks like image classification, speech recognition, and large-scale language modeling [5–7], where massive memory access and training on large, pre-collected datasets in controlled settings are possible (offline learning). On the other hand, these models are unsuitable for applications where power efficiency, continuous, and adaptive optimization are key requirements (online learning) [8]. This online learning approach is particularly critical for systems operating in an always-on mode, enabling them to adapt continuously to changing environments in edge applications such as autonomous systems, IoT devices, and personalized services.

A change of paradigm is thus needed to address such challenges. The field of neuromorphic computing has embraced this shift by drawing inspiration from the brain's remarkable ability to learn continuously, efficiently, and in a distributed manner [9]. Biological neural networks operate with local learning mechanisms that are adaptive, scalable, and optimized for real-time functionality [10, 11]. These networks use spike-based communication and feedback loops, enabling efficient, on-the-fly learning without the need for massive data storage or global error propagation [9]. The challenge of designing systems that meet the power and adaptability needs of edge devices calls for more than just energy-efficient hardware; it requires brain-inspired algorithms that fully exploit the capabilities of neuromorphic hardware.

However, current learning algorithms for mixed-signal devices face significant limitations in both performance and expressivity. Indeed, bio-inspired learning in neuromorphic hardware has been limited to unsupervised rules [12–17], which often require extensive hyperparameter tuning and the integration of multiple hard-coded stabilization mechanisms to function effectively [18, 19]. Furthermore, there is limited theoretical understanding of how to extend these mechanisms to multi-layered spiking neural networks (SNNs) [20]. Moreover, computer in-the-loop (ITL) methods with surrogate gradients have shown promise for training multi-layer SNNs on mixed-signal devices[21]. However, ITL approaches require storing the computation graph of the SNN, assuming idealized dynamics, computing the weight updates with backpropagation through time (BPTT), and reconfiguring the network at every training step. This process is energy-intensive and unsuitable for online learning in adaptable devices where energy efficiency is critical. As yet, training SNNs online and directly on neuromorphic devices remains a challenging engineering and computational problem.

In this work, we address this challenge by proposing a novel optimization algorithm that integrates spike-based local learning rules with feedback control signals. Our approach is inspired by recent work on biologically plausible credit assignment with control [22–26]. In this family of models, a control module is used to steer network activities toward target outputs and generate learning signals for effective credit assignment in deep ANNs [23]. Here, we implement these mechanisms in an SNN, introducing a novel architecture with a control module of spiking neurons. In our model, both inference and learning are mediated by the recurrent connections between these two modules. The error information is encoded in the temporal dynamics of synaptic currents, eliminating the need to store intermediate network states. This enables neurons to adjust synaptic weights online, removing the requirement for separate training and inference phases. Additionally, weight updates follow a fully local learning rule guided by the control signals, without relying on nested gradient computations typical of backpropagation.

First, we train SNNs on multiple supervised learning tasks with our feedback control optimizer. Second, we demonstrate that our SNN feedback control algorithm is suitable for online learning, and we validate its performance by emulating typical hardware constraints (i.e., number of connections and hardware substrate mismatch). In conclusion, this study demonstrates the potentiation of this novel hardware-aware algorithm to revolutionize the scalability of neuromorphic devices.

II. The feedback control optimizer

A. The network architecture

We built the architecture of the feedback control optimizer on a simple computational primitive: (1) an Integrate-and-Fire (LIF) output neuron, and (2) a controller module of one excitatory (positive) and one inhibitory (negative) LIF control neuron (Figure 1a, left). During training, the neuron receives external inputs (bottom-up), while the controller module receives external inputs from target sources (top-down). Each bottom-up input is associated with spiking patterns from top-down inputs representing a target activity for the output neuron. The controller module is only active during training and sends feedback signals proportional to the activity differences between the output neuron and the target sources. These feedback signals drive the neuron to match the target activities during training and provide a learning signal for credit assignment on the feedforward synapses [23, 26]. This computational unit can be directly implemented on neuromorphic devices and scaled to neural networks for on-chip programming of SNNs (Figure 1a). In this scenario, the feedback signals to the output neuron are also propagated to every layer, guiding the entire network toward the target activities.

In this work, we restricted the implementation to a one-layer network. Accordingly, the network comprises n output neurons (one neuron per input class). The controller module comprises n pairs of positive and negative control neurons associated with each output neuron. First, the dynamics of the membrane voltage and the spiking activity of the output neurons are described by the following equations,

$$\begin{cases} \mathbf{v}_t^{\text{out}} = \alpha\, \mathbf{v}_{t-1}^{\text{out}} - v_{th}\, \mathbf{s}_{t-1}^{\text{out}} + \mathbf{I}_t^{\text{ff}} + \mathbf{I}_t^{\text{fb}} \\ \mathbf{s}_t^{\text{out}} = H(\mathbf{v}_t^{\text{out}} - v_{th}) \,, \end{cases} \tag{1}$$

where $\alpha = 1 - \Delta t/\tau_m$ is the decay factor given by the membrane time constant τ_m and the timestep size Δt, v_{th} is the voltage threshold, $\mathbf{v}_t \in \mathbb{R}^n$ is the vector of membrane voltages of the network, $\mathbf{s}_t \in \mathbb{R}^n$ is the vector of binary spiking output of the network such that $s_{i,t} \in [0,1]$ for every i-th neuron, and $H(\cdot)$ is the Heaviside function. Here, $\mathbf{I}_t^{\text{ff}}$ and $\mathbf{I}_t^{\text{fb}}$ represent the synaptic currents of the external inputs and the feedback inputs from the control module, as described by the following equations,

$$\begin{cases} \mathbf{I}_t^{\text{ff}} = \beta\, \mathbf{I}_{t-1}^{\text{ff}} + \mathbf{W}\mathbf{s}_t^{\text{in}} \\ \mathbf{I}_t^{\text{fb}} = \beta\, \mathbf{I}_{t-1}^{\text{fb}} + \mathbf{Q}_p \mathbf{s}_{t-1}^{p} + \mathbf{Q}_n \mathbf{s}_{t-1}^{n} \,, \end{cases} \tag{2}$$

where $\beta = 1 - \Delta t/\tau_c$ is the decay factor given by the synaptic time constant τ_c and the timestep size Δt, $\mathbf{s}_t^{\text{in}} \in \mathbb{R}^m$ is the vector of spiking activity from the pre-synaptic inputs to the output neurons, $\mathbf{W} \in \mathbb{R}^{n,m}$ is the set of feedforward weights from the m pre-synaptic inputs to the n output neurons, $\mathbf{s}_t^p \in \mathbb{R}^n$ and $\mathbf{s}_t^n \in \mathbb{R}^n$ are the vectors of spiking activities of the positive and negative control neurons, and $\mathbf{Q}^p \in \mathbb{R}^{n,n}$ and $\mathbf{Q}^n \in \mathbb{R}^{n,n}$ the respective feedback weights. The feedforward currents $\mathbf{I}_t^{\text{ff}}$ accumulate over time the weighted spiking activity of the feedforward inputs with a decaying memory dictated by τ_c. Similarly, the feedback currents $\mathbf{I}_t^{\text{fb}}$ accumulate the weighted spiking activity of the control neurons.

Next, we modeled both control neurons as LIF neurons receiving spikes from the output neurons of the network and from the target sources (Figure 1a). The dynamics of the membrane voltage and the spiking activity of the control neurons are described by the following equations,

$$\begin{cases} \mathbf{u}_t^p = \gamma\, \mathbf{u}_{t-1}^p - u_{th}\, \mathbf{s}_{t-1}^p + \mathbf{J}_t^{\text{fb}} - \mathbf{J}_t^{\text{ff}} \\ \mathbf{u}_t^n = \gamma\, \mathbf{u}_{t-1}^n - u_{th}\, \mathbf{s}_{t-1}^n - \mathbf{J}_t^{\text{fb}} + \mathbf{J}_t^{\text{ff}} \\ \mathbf{s}_t^p = H(\mathbf{u}_t^p - u_{th}) \\ \mathbf{s}_t^n = H(\mathbf{u}_t^n - u_{th}) \,, \end{cases} \tag{3}$$

where $\gamma = 1 - \Delta t/\tau_u$ is the decay factor given by the membrane time constant τ_u and the timestep size Δt, u_{th} is the voltage threshold of the control neurons, $\mathbf{u}_t^p \in \mathbb{R}^n$ and $\mathbf{u}_t^n \in \mathbb{R}^n$ are, respectively, the vector of membrane voltages of the positive and negative control neurons, and $\mathbf{s}_t^p \in \mathbb{R}^n$ and $\mathbf{s}_t^n \in \mathbb{R}^n$ the vector of binary spiking output of the positive and negative control neurons. The spikes from the output neurons and target sources are accumulated into two respective synaptic current variables $\mathbf{J}_t^{\text{ff}}$ and $\mathbf{J}_t^{\text{fb}}$, as follows,

$$\begin{cases} \mathbf{J}_t^{\text{ff}} = \beta\, \mathbf{J}_{t-1}^{\text{ff}} + \mathbf{s}_t^{\text{out}} \\ \mathbf{J}_t^{\text{fb}} = \beta\, \mathbf{J}_{t-1}^{\text{fb}} + \mathbf{s}_t^{\text{trg}} \,, \end{cases} \tag{4}$$

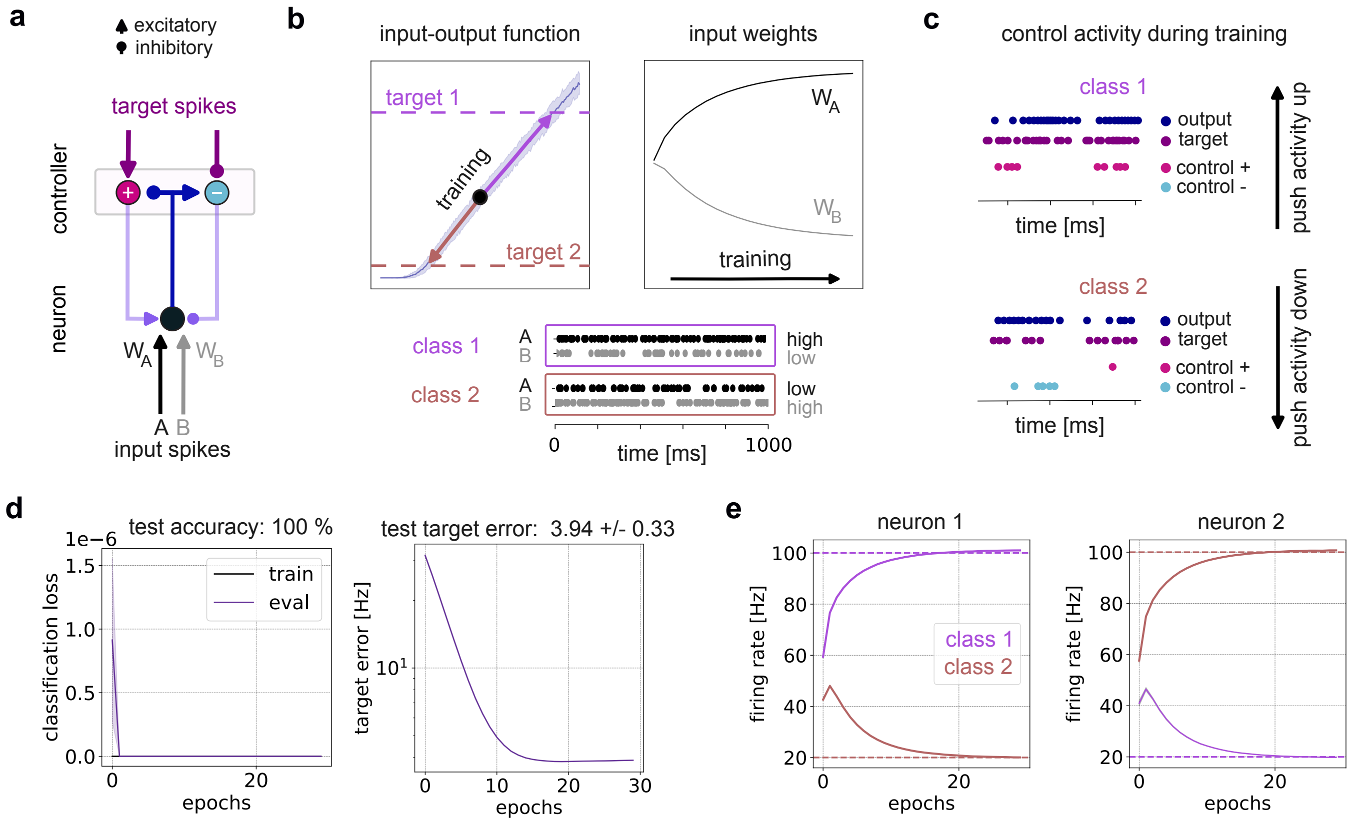

Fig. 1. **Illustration of the feedback control optimizer and binary classification task**. **a**) Illustration of the feedback control architecture. A neuron receives external input spikes (black and grey arrows) associated with learnable synaptic weights (W_A and W_B) and, in turn, sends output spikes to the control module (blue arrows). The neuron is paired with a positive (magenta) and a negative (light blue) control neuron. The positive and negative control neurons receive inhibitory and excitatory activity from the output neuron, respectively. The control neurons also receive external target spikes (purple arrows) and, in turn, send feedback spikes (light purple arrows). **b**) Illustration of the binary classification task. Bottom) The activity of the pre-synaptic inputs A and B for class 1 (pink) and 2 (orange), respectively. Top-left) The input-output function (f-f curve) of the neuron. The neuron has two target activities corresponding to the two classes (target 1 and target 2) and is trained to match these targets when inputs from the respective classes are presented. Top-right) Evolution of the learnable synaptic weights W_A and W_B during training. **c**) Illustration of the feedback control algorithm. Example spiking activities of the neuron, the targets, and the positive and negative control for an example input from class 1 (top) and class 2 (bottom). **d**) Left: the average cross-entropy loss during training (black) and validation (purple). Right: the mean absolute difference between output and target activity during validation. **e**) Average firing rate during validation for inputs from class 1 (black) and class 2 (light red). Left: neuron encoding class 1. Right: neuron encoding class 2. We calculate the mean and standard deviation across multiple simulations with different random seeds (see Methods).

where $\beta = 1 - \Delta t/\tau_c$ is the same decay factor as in Equation (2), $\mathbf{s}_t^{\text{out}} \in \mathbb{R}^n$ is the vector of spiking activity from the output neurons of the network (see Equation (1)), and $\mathbf{s}_t^{\text{trg}} \in \mathbb{R}^n$ is the vector of spiking activity from the target sources. We refer to the next section for a detailed description of the optimization algorithm.

B. Optimization with spiking feedback control

In this work, we use the feedback control optimizer to train SNNs on classification tasks. We use the firing rates of the output neurons $\mathbf{r}^{\text{out}} \in \mathbb{R}^n$ as the readout variables for classification,

$$\mathbf{r}^{\text{out}} = \frac{1}{T} \sum_{t=0}^{T} \mathbf{s}_t^{\text{out}} , \tag{5}$$

where $\mathbf{s}_t^{\text{out}}$ is the spiking activity from the output neurons (see Equation (1)). We formalize a standard classification problem by defining a cross-entropy objective,

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) \equiv -\sum_i \mathbf{y}_i \log(\hat{\mathbf{y}}_i) , \tag{6}$$

where $\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{r}_i^{\text{out}}) \in \mathbb{R}^n$ is the vector of predictions from the network for the i-th sample, and $\mathbf{y}_i \in \mathbb{R}^n$ the one-hot encoded true labels. For every i-th sample, the true labels $\mathbf{y}_i$ are associated with a set of target firing rates $\mathbf{r}_i^{\text{trg}} \in \mathbb{R}^n$ where the correct class has a higher target activity than the incorrect classes. We quantify the difference between target and output activities - the target error - with a complementary objective function,

$$\mathcal{L}_{\text{trg}}(\mathbf{r}_i^{\text{trg}}, \mathbf{r}_i^{\text{out}}) = \sum_i \left| \mathbf{r}_i^{\text{trg}} - \mathbf{r}_i^{\text{out}} \right| , \tag{7}$$

that is, the mean absolute difference between the target activity $\mathbf{r}^{\text{trg}}$ and the output firing rate of the network $\mathbf{r}^{\text{out}}$. Since the target activities hold the same structure as the one-hot encoded labels, optimizing the network for the objective in Equation (7) implicitly minimizes the classification loss in Equation (6) [23, 25]. In our framework, we optimize the network for the objective (6) with a feedback control approach [23]. The control system defined in Equation (3) implements a spiking PI controller that sends feedback proportional to an estimate of the difference between the target activity $\mathbf{r}^{\text{trg}}$ and the output activity $\mathbf{r}^{\text{out}}$. Indeed, the spiking activity $\mathbf{s}_t^{\text{trg}}$ is modeled as an homogeneous Poisson process with rate $\mathbf{r}^{\text{trg}}$, such that $\mathbb{E}[\mathbf{s}_t^{\text{trg}}] = \mathbf{r}^{\text{trg}}\Delta t$, where Δt is the timestep size. It follows from Equation (4) that the expected value of the current $\mathbf{J}_t^{\text{fb}}$ can be derived as,

$$\begin{aligned}
\lim_{t\to\infty} \mathbb{E}[\mathbf{J}_t^{\text{fb}}] &= \lim_{t\to\infty} \mathbb{E}\Big[\sum_{t'<t} \beta^{t-t'} \mathbf{s}_{t'}^{\text{trg}} + \beta^t \mathbf{J}_0^{\text{fb}}\Big] \\
&= \lim_{t\to\infty} \Big(\sum_{t'<t} \beta^{t-t'} \mathbf{r}^{\text{trg}}\,\Delta t + \beta^t \mathbb{E}[\mathbf{J}_0^{\text{fb}}]\Big) \\
&= \lim_{t\to\infty} \Big(\mathbf{r}^{\text{trg}}\,\Delta t \sum_{t'<t} \beta^{t-t'} + \beta^t \mathbb{E}[\mathbf{J}_0^{\text{fb}}]\Big) \\
&= \tau_c\,\Delta t\,\mathbf{r}^{\text{trg}}\,,
\end{aligned} \tag{8}$$

which is proportional to the target firing rate $\mathbf{r}^{\text{trg}}$ as $t \to \infty$. Furthermore, this current implements an exponential moving average of the underlying firing rate, with decay factor $\beta^{t-t'}$ determined by τ_c (see Equation (4)). When this time constant is longer than the average inter-spike interval $1/\mathbf{r}^{\text{trg}}$, then the current $\mathbf{J}_t^{\text{fb}}$ approximates the moving average of the underlying firing rate, that is,

$$\mathbf{J}_t^{\text{fb}} \propto\approx \mathbf{r}^{\text{trg}} \quad \text{for } t \gg \tau_c\,. \tag{9}$$

Similarly, the input spiking activity $\mathbf{s}_t^{\text{in}}$ follows a homogeneous Poisson process with rate $\mathbf{r}^{\text{in}}$, where $\mathbb{E}[\mathbf{s}_t^{\text{in}}] = \mathbf{r}^{\text{in}}\Delta t$. The same analysis can be then applied to the synaptic current $\mathbf{I}_t^{\text{ff}}$ in Equation (2), yielding

$$\mathbf{I}_t^{\text{ff}} \propto\approx \mathbf{r}^{\text{in}} \quad \text{for } t \gg \tau_c\,. \tag{10}$$

Under the assumptions that (1) τ_c is larger than the average inter-spike interval of both control and output neurons, and (2) the reset dynamics are negligible due to $\tau_u < \tau_c$, the synaptic currents in Equation (2) and Equation (4) approximates the underlying firing rate as follows,

$$\mathbf{J}_t^{\text{ff}} \propto\approx \mathbf{r}^{\text{out}} \ ; \ \mathbf{I}_t^{\text{fb}} \propto\approx \mathbf{r}^{\text{p}} + \mathbf{r}^{\text{n}}\,, \tag{11}$$

where $\mathbf{r}^{\text{p}}$ and $\mathbf{r}^{\text{n}}$ is the firing rate of the positive and negative control, respectively. Therefore, Equations (3) represent two spiking PI controllers driven by the estimated difference between the target and output firing rates.

The output neurons use feedback signals from the control module to guide learning [23]. Specifically, the difference in neuronal activity with feedback $\mathbf{r}^{\text{fb}}$ versus without feedback $\mathbf{r}^{\text{ff}}$ encodes error information which drives weight updates according to

$$\Delta\mathbf{W}\big|_t = (\mathbf{r}_t^{\text{fb}} - \mathbf{r}_t^{\text{ff}})\,{\mathbf{r}_t^{\text{in}}}^\top\,, \tag{12}$$

where $\mathbf{r}^{\text{in}}$ is the firing rates of the pre-synaptic neurons. The firing rate of a LIF neuron is approximately proportional to the total input current $\mathbf{I}_t^{\text{tot}}$ $\mathbf{r}_t$ as follows,

$$\mathbf{r}_t \equiv \mathbf{r}_t(\mathbf{I}_t^{\text{tot}}) \approx k\left(\mathbf{I}_t^{\text{tot}} - \bar{\mathbf{I}}\right) \quad \text{if} \quad \mathbf{I}_t^{\text{tot}} > \bar{\mathbf{I}}\,, \tag{13}$$

where k is a proportionality constant, $\mathbf{I}_t^{\text{tot}}$ is the total synaptic currents, and $\bar{\mathbf{I}}$ is the minimal threshold current that drives post-synaptic spikes. Substituting this into the weight update equation, we obtain

$$\begin{aligned}
\Delta\mathbf{W}\big|_t &= (\mathbf{r}_t^{\text{fb}} - \mathbf{r}_t^{\text{ff}})\,\mathbf{s}_t^\top \\
&\approx k(\mathbf{I}_t^{\text{fb}} + \mathbf{I}_t^{\text{ff}} - \bar{\mathbf{I}} - \mathbf{I}_t^{\text{ff}} - \bar{\mathbf{I}})\,\mathbf{s}_t^\top \\
&= k\,\mathbf{I}_t^{\text{fb}}\,\mathbf{s}_t^\top\,,
\end{aligned} \tag{14}$$

and thus the following learning rule,

$$\mathbf{W}\big|_t = \mathbf{W}\big|_{t-1} + \eta\,\mathbf{I}_t^{\text{fb}}\,\mathbf{s}_t^\top \tag{15}$$

where η is the learning rate. From (11), it follows that weight updates are driven by and proportional to the firing activity of the control neurons.

III. Results

A. Classification tasks with the feedback control optimizer

In the following, we demonstrate that the feedback control optimizer effectively trains single-layer SNN networks. To do so, we evaluate the algorithm's performances on two different datasets (see Datasets). We first demonstrate the dynamics of the feedback control with a binary classification task. In this setting, one output neuron receives spikes from two pre-synaptic inputs A (black) and B (grey) (Figure 1b). The inputs belong to two different classes: class 1 where input A fires at 100 Hz and input B at 50 Hz, and class 2 where input A fires at 50 Hz and input B at 100 Hz. The neuron is trained to reach specific target activities for each class (100 Hz for class 1 and 20 Hz for class 2). During training, the control neurons integrate the difference between the output neuron's activity (blue) and the target spikes (purple) into their membrane potential. When this potential reaches a threshold, the control neurons generate positive or negative feedback, adjusting the output neuron's activity to better match the target (Figure 1c). During validation and testing, the control module is inactive, and the output neuron relies solely on its learned synaptic weights W_A and W_B to reach the target activities. The feedback control drives the neuron to achieve zero classification loss during training and zero validation error in a few epochs, reaching 100% accuracy during the whole training process (Figure 1d). Consequently, the difference between the output and target activities without feedback control consistently decreases across epochs (Figure 1d). The neuron progressively reaches the target activities without feedback control for both class types when trained to encode for either class 1 or class 2 (Figure 1e). These results demonstrate that the control system can precisely drive the neuron activity toward target firing rates. The output neuron changes its synaptic weights proportionally to the feedback and learns to reach the target without the feedback.

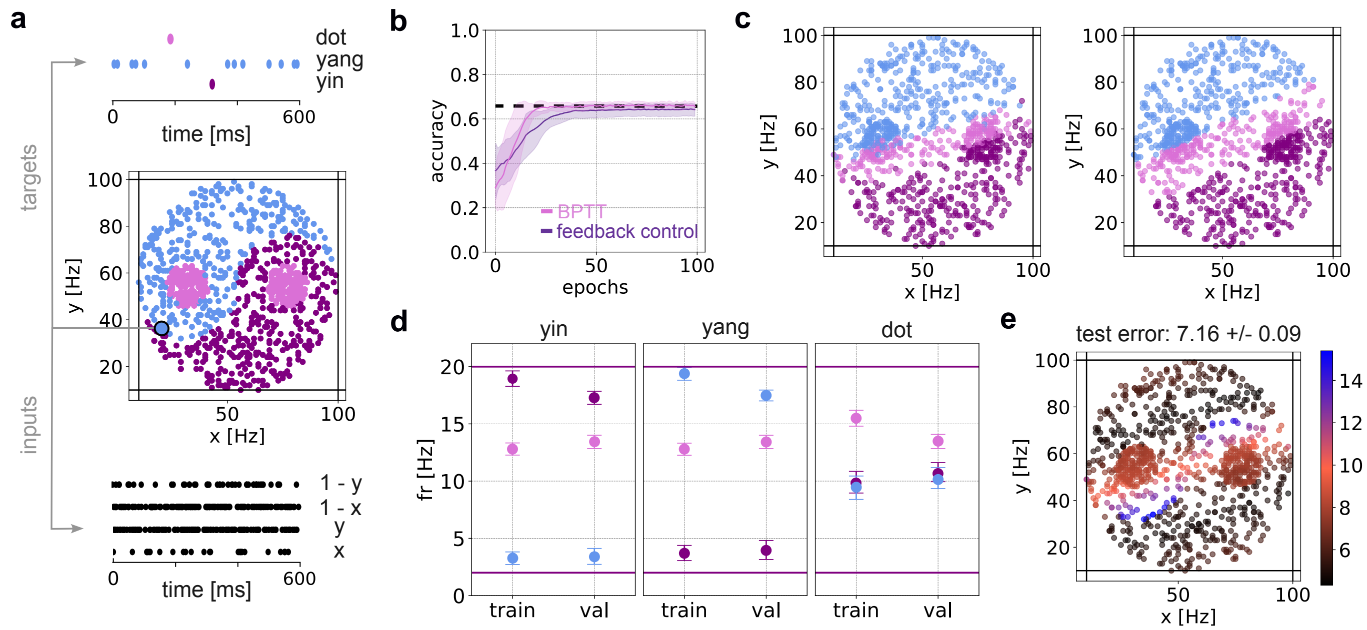

Fig. 2. **Training a single-layer SNN on the spiking Yin-Yang dataset with feedback control**. **a**) Illustration of the spiking Yin-Yang dataset and the corresponding targets. Each dot belongs to the Yin (purple), Yang (azure), or dot (pink) region. Here we illustrate the spiking patterns of the input coordinates (bottom) and target activities (top) of an example dot from the Yang class. **b**) The average validation accuracy for the feedback control network (purple) compared to a network of three leaky-integrator neurons trained with backpropagation-through-time (BPTT). The dashed blue line represents test accuracy for a standard readout layer trained with backpropagation, using firing rates directly as features for each dot's coordinates. **c**) Left) The network predictions on the test set, with a test accuracy of 0.63 ± 0.03. Right) The predictions of a standard ANN on the test set, with a test accuracy of 0.66 ± 0.02. **d**) The mean firing rate of the output neurons at the final epoch for both training (train) and validation (val) inputs across the Yin, Yang, and Dot classes. Each output neuron is color-coded according to the class it represents: purple for Yin, azure for Yang, and pink for Dot. **e**) The mean target error on the test set, with dots color-coded by associated target error (see color bar). We calculate the mean and standard deviation across multiple simulations with different random seeds (see Methods).

Next, we train a one-layer SNN on a spike-based version of the Yin-Yang dataset [27]. This dataset comprises dots from the Yin, Yang, and Dot regions. The position of each dot is determined by four coordinates x, y, $1-x$, and $1-y$. We encode these features as firing rates between 10 and 100 Hz, where each coordinate is represented by a homogeneous Poisson process with the given rate (Figure 2a). We train the three output neurons to achieve a target firing rate of 20 Hz for the correct class and 2 Hz otherwise (Figure 2a). We train the network with the feedback control optimizer for 100 epochs with a batch size of 20. We benchmark our results against (1) a single-layer network of leaky integrator units trained with backpropagation-through-time (BPTT) and (2) a linear readout layer using the 4 input firing rates as representations of each dot position. The SNN trained with feedback control achieves a comparable accuracy to the other two models trained with backpropagation (Figure 2b). Furthermore, the SNN produces similar predictions on the test set as those expected from a linear readout model [27] (Figure 2c). Accordingly, the neurons encoding the Yin/Yang classes show a firing rate closer to the high target for inputs of its respective class and close to the low target for inputs from the opposite class (Figure 2d). In contrast, the neuron encoding the Dot class shows similar firing rates across all classes, with a slight increase for Dot inputs. We note that during training (with active control) the output neurons reach the target firing rates more effectively than during validation (without active control). The target error is higher in the regions where the model provides wrong class predictions, in particular at the borders between the Ying and Yang regions (Figure 2e).

In summary, these results show that the feedback control optimizer effectively trains single-layer SNNs. The network matches the performance of backpropagation-trained models with neurons approaching the target firing rates when the feedback is active (training) and inactive (validation). The next step for testing the applicability of this optimizer for mixed-signal devices is to examine its performances under two important conditions: (1) online learning with a continuous input stream and (2) hyperparameter mismatch with analog components.

B. Online learning with feedback control

In this section, we test the first condition and we simulate the performance of the models during online learning. In this setting, the network receives inputs and targets sequentially without resetting the dynamics of both the output and the control neurons (see *Online learning* section in Numerical simulations). Moreover, the synaptic weights w_{ij} are updated continuously during training examples, each time a pre-synaptic spike s_j reaches the post-synaptic neuron (Figure 3a).

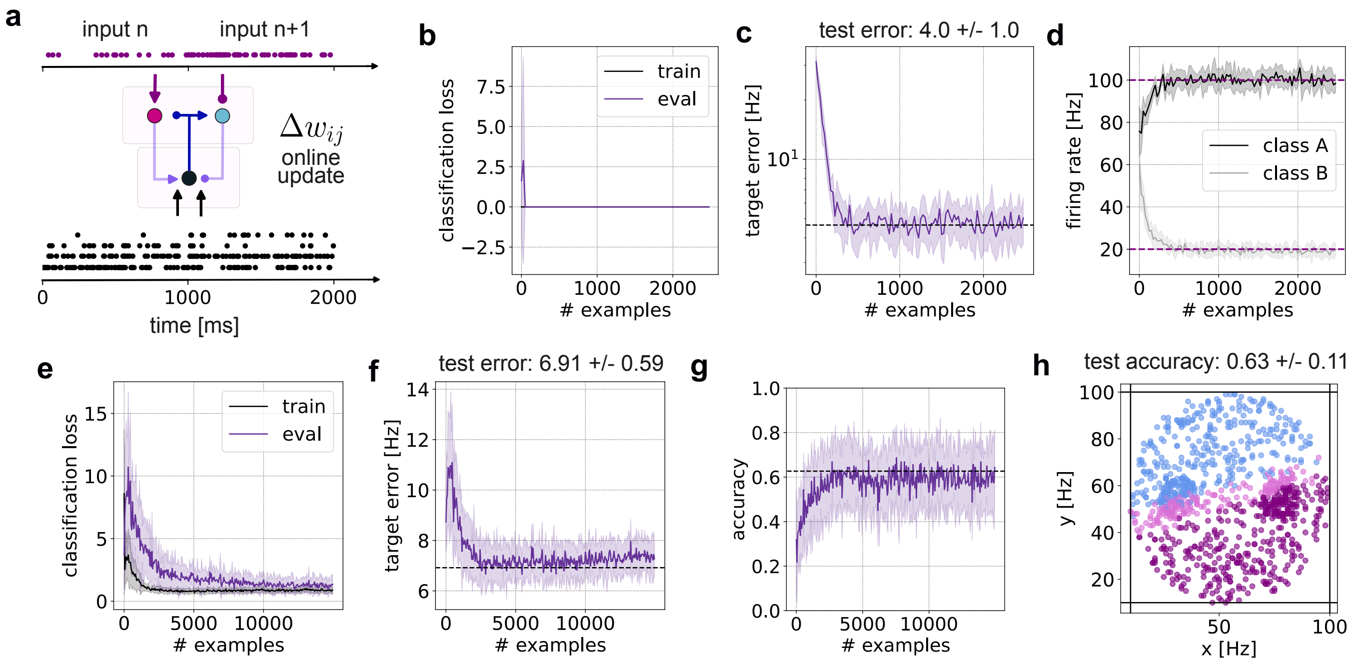

Fig. 3. **Online learning in a single-layer SNN with feedback control**. **a**) Schematic of the online learning setup. (**b-d**) Binary classification tasks. **b**) The average cross-entropy loss during training (black) and validation (purple). We calculate the mean and standard deviation by averaging over 25 sequential inputs. **c**) Same as **b** for the mean absolute difference between output and target activity during validation. **d**) Same as **b** for firing rate during validation for inputs from class A (black) and class B (grey) for a neuron encoding class A. (**e-h**) The Spiking Yin-Yang dataset: **e**) The average cross-entropy loss during training (black) and validation (purple). We calculate the mean and standard deviation by averaging over 50 sequential inputs. **f**) Same as **e** for the mean absolute difference between output and target activity during validation. The test target error is calculated offline over the whole testing set with batch size 20. **g**) Same as **e** for the average classification accuracy during validation. **h**) The average network predictions when testing offline over the whole testing set with batch size 20. We calculate the mean and standard deviation across multiple simulations with different random seeds (see Methods).

When optimizing the network on the binary classification task, we observe that the network achieves zero classification loss during training and zero validation loss after training on the first 500 input examples (Figure 3b). Similarly to our results in Figure 1, the target error decreases during training and reaches comparable performances to the offline training (Figure 3c). On average, the neuron consistently reaches the target activities (Figure 3d). Furthermore, we show in Figure 3e that the classification and validation error on the spiking Yin-Yang dataset also decreases when training the network online. The network reaches the same average accuracy as in the offline case after the first 50000 examples (Figure 3f). Accordingly, the predictions of the network are consistent with offline training (Figure 3g). The target error decreases during online training with a target error on the test set that is quantitatively comparable with offline training (Figure 3h).

In conclusion, these results show that the feedback control optimizer is suitable for online learning applications. This capability arises from the single-phase structure of the learning rule in Equation 15, which is driven by feedback control. Our results indicate that embedding error information directly into synaptic currents is essential for simultaneously adjusting neuronal activity and controlling weight updates.

C. Impact of device mismatch on classification performances

Finally, we evaluate the second condition for hardware implementation by analyzing the performance of the feedback control optimizer under varying levels of device mismatch (see the Methods section Numerical simulations). We train the same network as in Figure 2 introducing simulated device mismatches ranging from 0% to 20%, and quantify how these mismatches affect model performance. Next, we repeat the same analysis using populations of all-to-all connected neurons, simulating p neurons for each output and control neuron. In these simulations, each neuron is independently affected by device mismatch in its model parameter.

Classification accuracy without population averaging remains stable for mismatch levels up to 10%, with significant degradation only at higher levels (Figure 4a). Notably, using populations of just two neurons improves test classification accuracy and restores performance to levels observed without mismatch (Figure 4a). Additionally, the test error shows reduced variability around the median with two-neuron populations (Figure 2b). The validation loss at the end of training remains nearly constant across all tested mismatch levels, comparable to the scenario without population averaging (Figure 4c) Increasing the population size further recovers performance despite device mismatches, with a slight decline in performance for larger populations (Figure 4d).

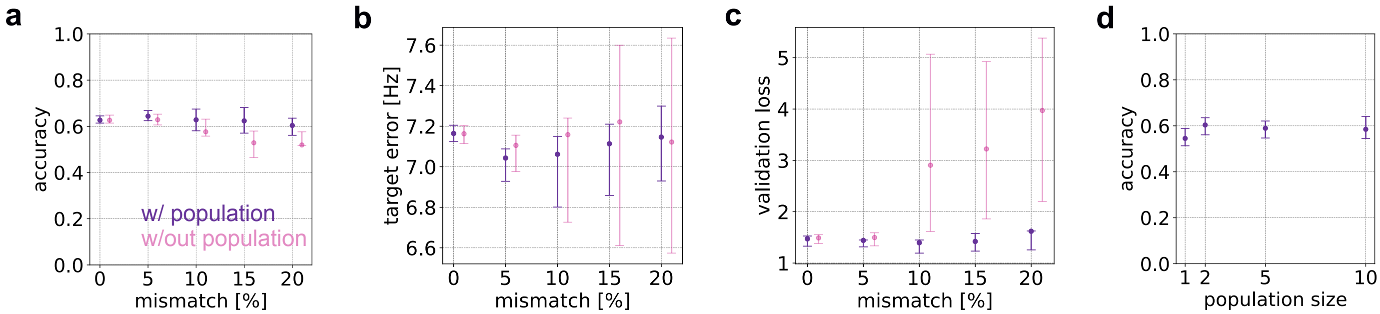

Fig. 4. **Robustness of the feedback control optimizer to device mismatch**. **a**) The test classification accuracy for different values of simulated device mismatch (in percentage). The performance are calculated with and without population averaging with $p = 2$ (purple and pink, respectively). The median and inter-quartile ranges are calculated over 15 simulations with different random seeds. **b**) Same as **a** for the test target error. **c**) Same as **a** for the validation loss. **d**) The test classification accuracy for different population sizes with a given mismatch (20%).

Our robustness test simulations demonstrate that the feedback control optimizer maintains high classification accuracy even under significant device mismatch. Furthermore, introducing neural populations of $p = 2$ fully restores the original performance, aligning well with the capabilities of current neuromorphic technologies [28, 29].

IV. Discussion

Efficient on-chip optimization algorithms are crucial for unlocking the full potential of mixed-signal neuromorphic devices in real-world applications. When paired with advancements in neuromorphic hardware, these algorithms can surpass standard machine learning methods in both power efficiency and adaptability. Building on recent advances in biologically plausible learning [23, 24, 26], we propose a feedback control optimizer specifically designed for online learning directly on hardware. Specifically, we introduce a novel architectural primitive where a control layer drives SNNs to reach target activities, while weights are optimized with a fully local learning rule. Our approach combines inference and learning in one single phase by relying on the recurrent connections between the network and the control, allowing for online learning on neuromorphic devices. Here, we first validate our method on a spike-based binary classification task. Next, we show that the feedback control optimizer can effectively train single-layer SNNs during both offline and online learning settings. Finally, we quantify the robustness of the algorithm against typical hardware mismatches, showing minimal loss in performance. Notably, our results show that the original performance levels can be fully restored by modestly increasing the number of neurons. Together, our study demonstrates that our feedback control optimizer can train SNNs online while effectively compensating for typical mismatches in mixed-signal neuromorphic devices. This work has the potential to advance the scalability and applicability of neuromorphic hardware significantly.

Previous work has proposed to train neuromorphic devices by taking inspiration from synaptic plasticity mechanisms observed experimentally [30–32]. The close alignment between these biological principles (binary communication, eligibility traces, and voltage dynamics, among others) and the functional design of neuromorphic devices makes these rules particularly well-suited for on-chip applications. While several spike-timing-dependent mechanisms have been successfully implemented on mixed-signal devices [33–36], these models often face limitations in expressivity and typically require additional mechanisms to ensure stability [37].

Alternatively, it is possible to train multi-layered SNNs offline with gradient-based learning by combining BPTT [2, 38] and surrogate gradients [39, 40]. This approach effectively bypasses the on-chip credit assignment problem by leveraging a conventional Von Neumann architecture to compute error backpropagation through automatic differentiation. However, this approach requires storing observables from the mixed-signal hardware over time, maintaining the computational graph, waiting for the forward pass to complete, retrieving stored information to calculate weight updates offline, and then reconfiguring the on-chip weights accordingly [21]. These requirements make this method unsuitable for online, on-chip learning scenarios, where both inference and learning are intended to occur on the device while it is running.

Looking forward, a key direction for future work is training multi-layered SNNS with the feedback control optimizer. This is fundamental for exploiting the scalability of the algorithm, investigating its capabilities on different datasets, and demonstrating the feasibility of on-chip optimization for multi-layered SNNs. Moreover, it is necessary to extend the testing of the algorithm with behavioral simulations and following implementation of the architecture on mixed-signal devices such as the DYNAP-SE [28, 29] and Brainscales [41]. While the optimization of these devices can initially be performed with a computer-in-the-loop setup [21], further testing of the algorithm will pave the way for designing a blueprint for a next-generation neuromorphic device.

V. Methods

Further information on the model hyperparameters and the code to reproduce our results is available at *add code here after the revisions*.

A. Datasets

In this work, we consider two different datasets. The first dataset is a custom binary classification set with 5000 training, 1000 validation, and 1000 test examples. Each sample consists of spiking activity from two inputs, A and B, with firing rates f_A and f_B. The two classes are defined by $f_A > f_B$ for class A and $f_B > f_A$ for class B. We define a pair of firing rates f_1 and f_0 (with $f_1 > f_0$) and assign the high f_1 and the low f_0 firing rate targets to the inputs based on their class. We simulate a homogeneous Poisson process over T timesteps with timestep size Δt, where each input and target samples have dimensions $(2, T)$ and $(1, T)$, respectively. The second dataset we used is a spike-based version of the Yin-Yang dataset [27]. Each sample in the original dataset represents a point in a 2D space encoded by coordinates x, y, $1 - x$, and $1 - y$, and classified into three classes (Yin, Yang, Dot) representing the yin-yang symbol. We preprocess this dataset by mapping each coordinate (values 0 to 1) to a firing rate with possible values f_{min} and f_{max}. We map the one-hot encoding of the correct class to a pair of firing rates f_1 and f_0 (with $f_1 > f_0$) and assign the high target f_1 and the low target f_0 for the correct class and the incorrect classes, respectively. Finally, we simulate a homogenenous Poisson process for each of the four input firing rates and each of the three target firing rates. Each input and target samples have dimensions $(4, T)$ and $(3, T)$, respectively, with 5000 training samples and 1000 each for validation and testing [27].

B. Numerical simulations

1) Offline training: We train the feedback control network with the binary classification dataset for 30 epochs with a batch size of 50 during both training and evaluation, learning rate $\eta = 10^{-5}$, and a total number of timestep $T = 5000$. The feedforward weights $\mathbf{W}$ are initialized randomly from a uniform distribution between $[0, 0.04]$. The feedback weights $\mathbf{Q}_p$ and $\mathbf{Q}_n$ are initialized as the identity matrix $\mathbf{Q}_p = \mathbf{Q}_n = \mathbb{I}_{n,n}$, such that each output neuron receives feedback only from its respective positive and negative control. When using the spiking Yin-Yang dataset, we train the models for 100 epochs with a batch size of 50 during both training and evaluation, learning rate $\eta = 1e10^{-4}$ for the feedback control network ($\eta = 5e10^{-4}$ for the linear leaky readout network), and a total number of timestep $T = 1000$. When using the standard Yin-Yang dataset, we train a linear readout layer for 300 epochs with a batch size of 20 during both training and evaluation, learning rate $\eta = 2e10^{-3}$, and input firing rates estimated from homogeneous Poisson processes with a total number of timestep $T = 1000$. For both types of Yin-Yang datasets, the feedforward synaptic weights $\mathbf{W}$ are initialized randomly from a Gaussian distribution with zero mean and $1/\sqrt{d_{in}}$ standard deviation (where $d_{in} = 4$). We set the timestep size $\Delta t = 1$ ms for every simulation. We randomly initialize the training, validation, test sets, and feedforward weights with a different seed. We estimate the average performance metrics and network firing rates across different random seeds (5 seeds for the binary classification tasks, 15 seeds for the Ying-Yang and Spiking Yin-Yang datasets).

2) Online learning: We train the models with a batch size of 1 (input and target samples are selected at random and processed sequentially) without resetting the model state variables between samples. Moreover, the synaptic weights $\mathbf{W}$ are updated continuously each time a pre-synaptic spike reaches a post-synaptic neuron during training (see Equation 15). This continuous update occurs over sequences of 25 examples for the binary classification task and 50 examples for the Yin-Yang dataset. We train the model with 2500 samples for binary classification (10000 samples for Spiking Yin-Yang), with $T = 4000$ ($T = 1000$) and learning rate $\eta = 1e10^{-9}$ ($\eta = 5e10^{-9}$). We compute the average metrics (classification loss, target error, classification accuracy, and output firing rates) over these sequences of examples and across 15 different random seeds. We initialize the dataset and the feedforward weights as during offline training.

3) Device mismatch: Device mismatches arise from variations in the electrical properties of identical components due to manufacturing imperfections [42, 43]. In analog circuits, these mismatches typically result in a coefficient of variation (CV) of around 0.2 [44, 45], although this value can vary depending on the circuit design and technology used. To simulate device mismatches, we define a mismatch percentage and apply a random deviation to each model parameter. This deviation is proportional to the ideal parameter value and the specified mismatch percentage. Each output and control neuron parameter is independently affected by random deviations based on the defined mismatch.

ACKNOWLEDGMENT

M.S. was supported by the ETH Postdoctoral fellowship from ETH Zùrich (reference nr. 23-2 FEL-042). C.D.L has received funding from: Bridge Fellowship founded by the Digital Society Initiative at University of Zurich (grant no.G-95017-01-12). G.I. was supported by the HORIZON EUROPE EIC Pathfinder Grant ELEGANCE (Grant No. 101161114) and has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI). B.G. was supported by the Swiss National Science Foundation (CRSII5-173721, 315230 189251) and ETH project funding (24-2 ETH-032).

AUTHORSHIP CONTRIBUTIONS

Conceptualization: MS, CDL, GI, BG. Mathematical analysis: MS. Numerical simulations: MS. Writing: MS, CDL, GI, BG. Supervision: GI, BG.

CODE AVAILABILITY

The code to reproduce our results and all the Figures in the text is freely available at https://github.com/matteosaponati/snn-feedback-control-NICE-2025.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[2] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088 (1986), pp. 533–536.

[3] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for modern deep learning research". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 09. 2020, pp. 13693–13696.

[4] Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. "Backpropagation and the Brain". In: *Nature Reviews Neuroscience* 21.6 (June 2020), pp. 335–346. ISSN: 1471-0048. DOI: 10.1038/s41583-020-0277-3. (Visited on 07/06/2023).

[5] S. M. Plis et al. "Deep learning for neuroimaging: a validation study". In: *Frontiers in Neuroscience* 8 (2014). DOI: 10.3389/fnins.2014.00229.

[6] V. Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518 (7540 2015), pp. 529–533. DOI: 10.1038/nature14236.

[7] Ashish Vaswani et al. "Attention Is All You Need.(Nips), 2017". In: *arXiv preprint arXiv:1706.03762* 10 (2017), S0140525X16001837.

[8] Jiasi Chen and Xukan Ran. "Deep learning with edge computing: A review". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1655–1674.

[9] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. "Towards spike-based machine intelligence with neuromorphic computing". In: *Nature* 575.7784 (2019), pp. 607–617.

[10] Giacomo Indiveri and Shih-Chii Liu. "Memory and information processing in neuromorphic systems". In: *Proceedings of the IEEE* 103.8 (2015), pp. 1379–1397.

[11] Adam H Marblestone, Greg Wayne, and Konrad P Kording. "Toward an integration of deep learning and neuroscience". In: *Frontiers in computational neuroscience* 10 (2016), p. 215943.

[12] Eugene M Izhikevich. "Solving the distal reward problem through linkage of STDP and dopamine signaling". In: *Cerebral cortex* 17.10 (2007), pp. 2443–2452.

[13] Wulfram Gerstner, Richard Kempter, J Leo Van Hemmen, and Hermann Wagner. "A neuronal learning rule for sub-millisecond temporal coding". In: *Nature* 383.6595 (1996), pp. 76–78.

[14] Jean-Pascal Pfister and Wulfram Gerstner. "Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity". In: *Journal of Neuroscience* 26.38 (2006), pp. 9673–9682.

[15] Claudia Clopath and Wulfram Gerstner. "Voltage and Spike Timing Interact in STDP – A Unified Model". In: *Frontiers in Synaptic Neuroscience* 2 (July 2010). ISSN: 1663-3563. DOI: 10.3389/fnsyn.2010.00025. (Visited on 07/16/2024).

[16] Dmitry Krotov and John J Hopfield. "Unsupervised learning by competing hidden units". In: *Proceedings of the National Academy of Sciences* 116.16 (2019), pp. 7723–7731.

[17] Devdhar Patel and Robert Kozma. "Unsupervised Features Extracted using Winner-Take-All Mechanism Lead to Robust Image Classification". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[18] Timothée Masquelier, Rudy Guyonneau, and Simon J. Thorpe. "Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains". In: *PLOS ONE* 3.1 (Jan. 2008), e1377. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0001377. (Visited on 06/05/2023).

[19] Aboozar Taherkhani et al. "A review of learning in biologically plausible spiking neural networks". In: *Neural Networks* 122 (2020), pp. 253–272.

[20] Michael Pfeiffer and Thomas Pfeil. "Deep Learning With Spiking Neurons: Opportunities and Challenges". In: *Frontiers in Neuroscience* 12 (Oct. 2018). ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00774. (Visited on 11/27/2024).

[21] Benjamin Cramer et al. "Surrogate Gradients for Analog Neuromorphic Computing". In: *Proceedings of the National Academy of Sciences* 119.4 (Jan. 2022), e2109194119. DOI: 10.1073/pnas.2109194119. (Visited on 08/22/2023).

[22] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. "Dendritic Cortical Microcircuits Approximate the Backpropagation Algorithm". In: *Advances in neural information processing systems* 31 (2018).

[23] Alexander Meulemans et al. "Credit Assignment in Neural Networks through Deep Feedback Control". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 4674–4687. (Visited on 07/10/2023).

[24] Alexander Meulemans, Matilde Tristany Farinha, Maria R. Cervera, João Sacramento, and Benjamin F. Grewe. "Minimizing Control for Credit Assignment with Strong Feedback". In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, June 2022, pp. 15458–15483. (Visited on 07/10/2023).

[25] Alexander Meulemans, Nicolas Zucchet, Seijin Kobayashi, Johannes von Oswald, and João Sacramento. *The Least-Control Principle for Local Learning at Equilibrium*. Oct. 2022. DOI: 10.48550/arXiv.2207.01332. (Visited on 11/10/2024).

[26] Pau Vilimelis Aceituno, Matilde Tristany Farinha, Reinhard Loidl, and Benjamin F. Grewe. "Learning Cortical Hierarchies with Temporal Hebbian Updates". In: *Frontiers in Computational Neuroscience* 17 (2023). ISSN: 1662-5188. (Visited on 07/10/2023).

[27] Laura Kriener, Julian Göltz, and Mihai A. Petrovici. *The Yin-Yang Dataset*. Jan. 2022. DOI: 10.48550/arXiv.2102.08211. (Visited on 06/10/2024).

[28] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)". In: *IEEE Transactions on Biomedical Circuits and Systems* 12.1 (Feb. 2018), pp. 106–122. ISSN: 1940-9990. DOI: 10.1109/TBCAS.2017.2759700. (Visited on 11/16/2023).

[29] Ole Richter et al. *DYNAP-SE2: A Scalable Multi-Core Dynamic Neuromorphic Asynchronous Spiking Neural Network Processor*. Nov. 2023. DOI: 10.48550/arXiv.2310.00564. (Visited on 11/16/2023).

[30] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. "Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs". In: *Science* 275.5297 (Jan. 1997), pp. 213–215. DOI: 10.1126/science.275.5297.213. (Visited on 05/04/2023).

[31] Per Jesper Sjöström, Gina G Turrigiano, and Sacha B Nelson. "Rate, Timing, and Cooperativity Jointly Determine Cortical Synaptic Plasticity". In: *Neuron* 32.6 (Dec. 2001), pp. 1149–1164. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(01)00542-6. (Visited on 04/20/2023).

[32] Katie C Bittner, Aaron D Milstein, Christine Grienberger, Sandro Romani, and Jeffrey C Magee. "Behavioral Time Scale Synaptic Plasticity Underlies CA1 Place Fields". In: *Science* 357.6355 (2017), pp. 1033–1036.

[33] Giacomo Indiveri. "Neuromorphic Bisable VLSI Synapses with Spike-Timing-Dependent Plasticity". In: *Advances in Neural Information Processing Systems*. Vol. 15. MIT Press, 2002. (Visited on 11/15/2024).

[34] E. Chicca et al. "A VLSI Recurrent Network of Integrate-and-Fire Neurons Connected by Plastic Synapses with Long-Term Memory". In: *IEEE Transactions on Neural Networks* 14.5 (Sept. 2003), pp. 1297–1307. ISSN: 1941-0093. DOI: 10.1109/TNN.2003.816367. (Visited on 11/15/2024).

[35] S Mitra, S Fusi, and G Indiveri. "Real-Time Classification of Complex Patterns Using Spike-Based Learning in Neuromorphic VLSI". In: *IEEE transactions on biomedical circuits and systems* 3.1 (Feb. 2009), pp. 32–42. ISSN: 1940-9990. DOI: 10.1109/tbcas.2008.2005781. (Visited on 11/15/2024).

[36] Shubha Ramakrishnan, Paul E. Hasler, and Christal Gordon. "Floating Gate Synapses With Spike-Time-Dependent Plasticity". In: *IEEE Transactions on Biomedical Circuits and Systems* 5.3 (June 2011), pp. 244–252. ISSN: 1940-9990. DOI: 10.1109/TBCAS.2011.2109000. (Visited on 11/15/2024).

[37] Arianna Rubino, Matteo Cartiglia, Melika Payvand, and Giacomo Indiveri. "Neuromorphic Analog Circuits for Robust On-Chip Always-on Learning in Spiking Neural Networks". In: *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2023, pp. 1–5.

[38] Paul J Werbos. "Backpropagation through Time: What It Does and How to Do It". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[39] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks". In: *IEEE Signal Processing Magazine* 36.6 (Nov. 2019), pp. 51–63. ISSN: 1558-0792. DOI: 10.1109/MSP.2019.2931595.

[40] Friedemann Zenke and Tim P. Vogels. "The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks". In: *Neural Computation* 33.4 (Mar. 2021), pp. 899–925. ISSN: 0899-7667. DOI: 10.1162/neco_a_01367. (Visited on 02/22/2024).

[41] Christian Pehle et al. "The BrainScaleS-2 Accelerated Neuromorphic System With Hybrid Plasticity". In: *Frontiers in Neuroscience* 16 (2022). ISSN: 1662-453X. (Visited on 11/16/2023).

[42] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers. "Matching properties of MOS transistors". In: *IEEE Journal of Solid-State Circuits* 24.5 (Oct. 1989), pp. 1433–1440. DOI: 10.1109/JSSC.1989.572629.

[43] Wen Sun et al. "Understanding memristive switching via in situ characterization and device modeling". In: *Nature communications* 10.1 (2019), pp. 1–13. DOI: 10.1038/s41467-019-11411-6.

[44] Dmitrii Zendrikov, Sergio Solinas, and Giacomo Indiveri. "Brain-inspired methods for achieving robust computation in heterogeneous mixed-signal neuromorphic processing systems". In: *Neuromorphic Computing and Engineering* 3.3 (July 2023), p. 034002. ISSN: 2634-4386. DOI: 10.1088/2634-4386/ace64c.

[45] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri. "Neuromorphic electronic circuits for building autonomous cognitive systems". In: *Proceedings of the IEEE* 102.9 (Sept. 2014), pp. 1367–1388. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2313954.