This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Online Spatio-Temporal Learning in Deep Neural Networks

Thomas Bohnstingl, *Member, IEEE*, Stanisław Woźniak, *Member, IEEE*, Angeliki Pantazi, *Senior Member, IEEE*, and Evangelos Eleftheriou, *Life Fellow, IEEE*

***Abstract*— Biological neural networks are equipped with an inherent capability to continuously adapt through online learning. This aspect remains in stark contrast to learning with error backpropagation through time (BPTT) that involves offline computation of the gradients due to the need to unroll the network through time. Here, we present an alternative online learning algorithmic framework for deep recurrent neural networks (RNNs) and spiking neural networks (SNNs), called online spatio-temporal learning (OSTL). It is based on insights from biology and proposes the clear separation of spatial and temporal gradient components. For shallow SNNs, OSTL is gradient equivalent to BPTT enabling for the first time online training of SNNs with BPTT-equivalent gradients. In addition, the proposed formulation unveils a class of SNN architectures trainable online at low time complexity. Moreover, we extend OSTL to a generic form, applicable to a wide range of network architectures, including networks comprising long short-term memory (LSTM) and gated recurrent units (GRUs). We demonstrate the operation of our algorithmic framework on various tasks from language modeling to speech recognition and obtain results on par with the BPTT baselines.**

***Index Terms*— Backpropagation, backpropagation through time (BPTT), online learning, real-time recurrent learning (RTRL), spiking neurons.**

I. Introduction

THE brain has the unique capability to adapt to changing environmental conditions with plastic synapses [1] and is able to perform intelligent tasks unattainable yet by computers while consuming very low power [2]. Brain-inspired concepts in machine learning applications have so far primarily focused on incorporating the layered, highly interconnected topology of biological neural networks into so-called artificial neural networks (ANNs), including recurrent neural networks (RNNs). RNNs utilizing more complex long short-term memory (LSTM) or gated recurrent units (GRUs) have demonstrated astounding successes in applications requiring learning from temporal data such as speech recognition or language modeling [3]–[5]. Recent works have transferred the rich dynamics of the biologically-inspired spiking neural networks (SNNs) to RNNs and demonstrated on-par performance in certain applications by utilizing gradient-based learning methods for training [6]–[10]. Typically, the aforementioned models are trained with the ubiquitous error backpropagation through time (BPTT) algorithm [11]. Despite the successes of BPTT-trained networks, this algorithm has severe limitations, especially in scenarios involving online learning, i.e., when the network is required to process and simultaneously learn from a continuous stream of input data [12], [13]. This is because BPTT training has to track all past activities by unrolling the network, which becomes very deep in time as the input-sequence length increases. For example, a two-second-long spoken input sequence with 1 ms time steps results in a 2000-layer-deep unrolled network.

To address these issues, online learning algorithms were developed for calculating the parameters' updates in real time as the input data arrives. In this manner, they are conceptually more similar to the way the brain adapts to changing environmental conditions. Two such online algorithms were initially introduced in [14]. The first one, real-time recurrent learning (RTRL), applies updates to the parameters immediately at each time step, hence sacrificing gradient-equivalence to BPTT, whereas the second one, which we refer to RTRL with deferred updates, applies BPTT-equivalent updates at the end of the input sequence. However, both versions of RTRL have higher time complexity than BPTT and for this reason, remained rarely used in practice.

Recently, online learning algorithms regained popularity and were investigated following two distinct research directions. The first direction focuses on approximations of RTRL with reduced computational complexity. It led to the development of various algorithms; for example, unbiased online recurrent optimization (UORO) [15], Kronecker-factored RTRL (KF-RTRL) [16] or optimal Kronecker-sum approximation of RTRL (OK-RTRL) [17], which all provide approximate gradients. In contrast, the second direction takes more inspiration from biological learning systems and is centered around the debate whether and how they employ error backpropagation [18]. The algorithm, synaptic plasticity dynamics for deep continuous local learning (DECOLLE), avoids using error backpropagation and instead relies on temporally and

Manuscript received May 6, 2021; revised December 4, 2021; accepted February 14, 2022. This work was supported by the ERA-NET CHIST-ERA Programme by the Swiss National Science Foundation (SNSF) under Project 20CH21_186999/1. *(Corresponding author: Thomas Bohnstingl.)*

Thomas Bohnstingl is with IBM Research Zurich, 8803 Rüschlikon, Switzerland, and also with the Institute of Theoretical Computer Science, Graz University of Technology, 8010 Graz, Austria (e-mail: boh@zurich.ibm.com).

Stanisław Woźniak and Angeliki Pantazi are with IBM Research Zurich, 8803 Rüschlikon, Switzerland.

Evangelos Eleftheriou was with IBM Research Zurich, 8803 Rüschlikon, Switzerland. He is now with Axelera AI, 8038 Zürich, Switzerland.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3153985.

Digital Object Identifier 10.1109/TNNLS.2022.3153985

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

spatially local auxiliary error functions computed with layer-wise local readout connections [19]. Learning algorithms, such as SuperSpike [20], random feedback local online (RFLO) [21] and E-prop [13] were proposed to approximate the gradients under biological constraints, see [22] for a summary. The development of these algorithms was inspired by the notion of the so-called eligibility traces that maintain a temporal trace of past neuronal events and the so-called learning signals that transport information spatially from the environment [13], [23]. Note that the above-mentioned approaches require either custom network architectures, utilize approximate gradients, or were primarily derived for a single-large layer of recurrent units. In particular, the latter design choice is quite limiting for modern deep learning architectures, where often multiple layers of recurrent units are stacked to achieve state-of-the-art performance [3], [24], [25].

In general, in the gradient-based training of RNNs, one can distinguish two types of gradient flows. First, gradients that flow between units within the same time step, and second, gradients that flow between units across different time steps. In this work, we revisit the formulation of this gradient-based training and propose a novel online learning methodology that clearly separates these gradients into two components: spatial and temporal. Furthermore, this separation is inspired by the temporal nature of the eligibility traces and the spatial characteristics of the learning signals. Because this gradient separation plays a key role, we refer to our algorithmic framework as online spatio-temporal learning (OSTL).

Contrary to previous approaches, we derive OSTL for a deep recurrent network of spiking neurons and prove that for the special case of a network with a single-recurrent SNN layer the proposed gradient separation maintains equivalence to BPTT. We show that the common deep feed-forward SNN architectures can be trained online with low time complexity and comparable performance to BPTT. Moreover, we extend OSTL for training generic RNNs, such as LSTM networks, with a special focus on deep networks. Our work complements the approximate RTRL-based approaches and the biologically plausible learning methods by proposing a unified approach that introduces biological insights. The key contribution is a novel algorithmic framework for online learning with a clear separation of spatial and temporal gradients, that includes the following.

1) A derivation for deep feed-forward SNNs with time complexity of $O(Kn^2)$, where K is the number of layers and n is the largest layer size.
2) A derivation for shallow feed-forward SNNs with time complexity of $O(n^2)$ while maintaining the gradient equivalence with BPTT.
3) A generalized derivation for deep RNNs, with explicit formulations for LSTMs and GRUs, with time complexity of $O(Kn^4)$.
4) A novel methodology to incorporate approximations separately into spatial and temporal gradients allowing to unify the state-of-the-art online learning algorithms.

The remainder of this article is organized as follows. In Section II, we give a brief overview of SNNs, focusing on a concept that allows for simple integration into deep learning frameworks by representing the SNN with building blocks of RNNs. In Section III, we present the OSTL algorithmic framework and its application to deep and single-layer SNNs. In Section III, we also discuss the generalization of OSTL to generic RNNs, such as LSTMs and GRUs. In Section IV, we compare OSTL to related algorithms from the literature. The detailed performance evaluation of OSTL on various tasks is shown in Section V. Finally, Section VI concludes this article.

II. Spiking Neural Networks

Inspired by insights from neuroscience, SNNs are network architectures with biologically realistic neuronal dynamics and synaptic learning mechanisms. In a simplified view, they consist of neurons, which are interconnected via synapses, receiving input spikes at the dendrites and emitting output spikes through the axons, as shown in Fig. 1(a). Biological neurons maintain a temporal trace of past neuronal events in the eligibility traces that along with presynaptic and postsynaptic activities modulate the synaptic plasticity and therefore provide means for temporal credit assignment [23]. In addition, there exist several learning signals, e.g., dopamine, that transport information spatially from the environment or other brain regions to the individual neurons, as shown in Fig. 1(a).

While the neuronal dynamics have been successfully abstracted to several neuron models, such as the well-known leaky integrate-and-fire (LIF) neuron model, the training was historically performed with variants of the spike-timing-dependent plasticity (STDP) Hebbian rule [23]. Although STDP is a simple biologically inspired online learning mechanism, the accuracy of STDP-based architectures is inferior to that of state-of-the-art deep ANNs trained with BPTT. To address this problem, several research groups have focused their activities on facilitating gradient-based training for SNNs [7]–[10]. In particular, in [10], an alternative viewpoint on the spiking neuron was presented, which incorporates the neural dynamics into a recurrent ANN unit called a spiking neural unit (SNU). Although the SNU concept was developed around the simple LIF neuronal model, the framework is generic and readily extendable to other more involved neuronal constructs and biologically motivated extensions, such as lateral inhibition and adaptive spiking threshold (see [10, Supplementary Note 1]). In essence, the SNU framework bridges the ANN world with the SNN world by recasting the SNN dynamics with ANN-based building blocks. A deep recurrent network composed of SNUs is shown in Fig. 1(b).

The state and output equations for a recurrent SNN layer l composed of n_l SNUs, shown in Fig. 2(a), are

$$\boldsymbol{s}_l^t = \mathbf{g}\left(\mathbf{W}_l \boldsymbol{y}_{l-1}^t + \mathbf{H}_l \boldsymbol{y}_l^{t-1} + d \cdot \boldsymbol{s}_l^{t-1} \odot \left(\mathbb{1} - \boldsymbol{y}_l^{t-1}\right)\right) \quad (1)$$

$$\boldsymbol{y}_l^t = \mathbf{h}\left(\boldsymbol{s}_l^t + \mathbf{b}_l\right) \quad (2)$$

where $\boldsymbol{s}_l^t \in \mathbb{R}^{n_l}$ represents the internal states, i.e., the membrane potentials, $\boldsymbol{y}_l^{t-1} \in \mathbb{R}^{n_l}$ denotes the output of the neurons of the lth layer at time $t-1$, $\boldsymbol{y}_{l-1}^t \in \mathbb{R}^{n_{l-1}}$ denotes the output of the neurons of the $(l-1)$th layer at time t, $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ denotes the input weights from layer $l-1$ to

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

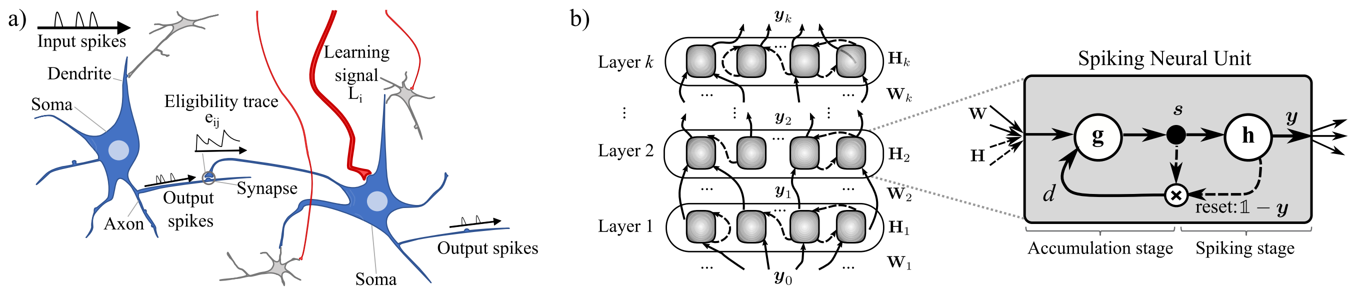


Fig. 1. **Schematic depiction of neural networks.** (a) Illustration of biological neurons connected to each other via synapses. Eligibility traces associated with each synapse maintain a temporal trace of past neuronal events. Learning signals propagated spatially from different brain regions target selected populations of neurons. (b) Illustration of a deep RNN composed of SNUs.

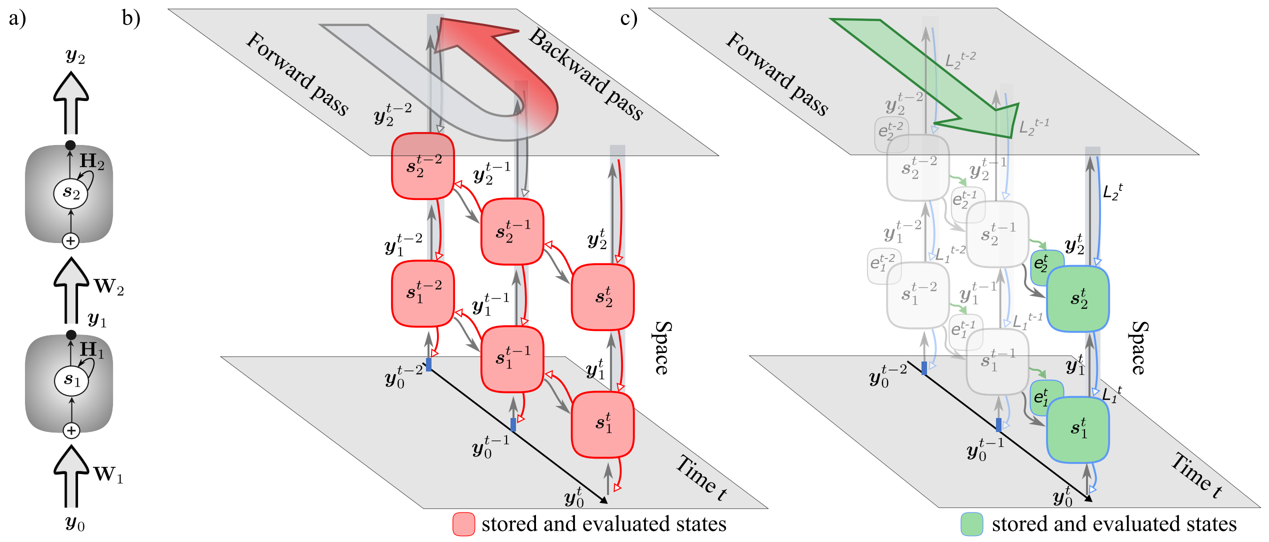


Fig. 2. **Illustration of the computational flow of BPTT and OSTL.** (a) Conceptual illustration of a two-layered RNN. (b) In BPTT, the activities and states of the network are computed forward in time and the gradients are propagated backward in time. During the update at time step t all states are stored and evaluated. (c) In OSTL, the spatial and temporal components are clearly separated. Each layer computes eligibility traces, which account for the temporal gradients. Independently, an individual learning signal per layer passes from the output layer through the network and accounts for the spatial components. The components involved in the gradient computation at time t are marked in green.

layer l, $\mathbf{H}_l \in \mathbb{R}^{n_l \times n_l}$ denotes the recurrent weights of layer l, $\mathbf{b}_l \in \mathbb{R}^{n_l}$ represents the firing thresholds, $d \in \mathbb{R}$ is a constant that represents the decay of the membrane potential, $\odot$ denotes pointwise vector multiplication, and $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are the input and output activation functions, respectively. In order to accurately represent the dynamics of the LIF neuron model, we set $\mathbf{g}(\boldsymbol{x}) = \mathbb{1}(\boldsymbol{x})$ and $\mathbf{h}(\boldsymbol{x}) = 0$ if $\boldsymbol{x} < 0$ and $\mathbf{h}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \geq 0$, i.e., $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\Theta}(\boldsymbol{x})$, with the identity $\mathbb{1}(\boldsymbol{x})$ and the heaviside function $\boldsymbol{\Theta}(\boldsymbol{x})$. Note that the SNU can also be configured to provide a continuous output, forming a so-called soft SNU (sSNU), for example, by using the sigmoid function $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})$, and thus operating similar to other RNN units.

The SNU formulation enables training of deep SNNs through the application of the BPTT approach, as shown in Fig. 2(b). This type of network architecture takes close inspiration from the organization of the human brain, where several recurrently connected brain regions are hierarchically connected [26], [27].

III. Online Spatio-Temporal Learning

We take inspiration from the existence of the eligibility traces and the learning signals and introduce OSTL, a novel algorithmic framework for online learning that is addressing the limitations of BPTT discussed so far. To that end, we explicitly separate the gradient computation into temporal components corresponding to the eligibility traces and into spatial components corresponding to the learning signals.

A common objective of gradient-based learning in neural networks is to train the parameters $\boldsymbol{\theta}$, which represents a collection of all trainable parameters of the network, so that the error E is minimized. In deep SNNs, the network error $E^t \in \mathbb{R}$ at time t is only a function of the output of the neurons in the last layer K, i.e., $\boldsymbol{y}_K^t$ and the target outputs $\hat{\boldsymbol{y}}^t$, i.e., $E^t = \phi(\boldsymbol{y}_K^t, \hat{\boldsymbol{y}}^t)$. A generic layer of spiking neurons produces outputs $\boldsymbol{y}_l^t = \boldsymbol{\chi}(\boldsymbol{s}_l^t, \boldsymbol{\theta}_l)$ and $\boldsymbol{s}_l^t = \boldsymbol{\psi}(\boldsymbol{s}_l^{t-1}, \boldsymbol{y}_l^{t-1}, \boldsymbol{y}_{l-1}^t, \boldsymbol{\theta}_l)$, where all the trainable parameters, $\mathbf{W}_l, \mathbf{H}_l, \mathbf{b}_l$, are collectively described by the variable $\boldsymbol{\theta}_l$. For example, for the SNU highlighted in Section II, $\boldsymbol{\psi}(\boldsymbol{s}_l^{t-1}, \boldsymbol{y}_l^{t-1}, \boldsymbol{y}_{l-1}^t, \boldsymbol{\theta}_l) = \mathbf{g}(\mathbf{W}_l \boldsymbol{y}_{l-1}^t + \mathbf{H}_l \boldsymbol{y}_l^{t-1} + d \cdot \boldsymbol{s}_l^{t-1} \odot (\mathbb{1} - \boldsymbol{y}_l^{t-1}))$ and $\boldsymbol{\chi}(\boldsymbol{s}_l^t, \boldsymbol{\theta}_l) = \mathbf{h}(\boldsymbol{s}_l^t + \mathbf{b}_l)$.

Using this notation, we compute the updates of the parameters $\Delta\boldsymbol{\theta}_l$ that minimize E based on the principle of gradient descent as

$$\Delta\boldsymbol{\theta}_l = -\eta \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} \tag{3}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

where $\eta \in I\!R$ is the learning rate. We employ the chain rule, i.e., the core ingredient of BPTT, to reformulate the term $(\mathrm{d}E/(\mathrm{d}\boldsymbol{\theta}_l)$ from 3 as

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_{1\le t\le T} \frac{\partial E^t}{\partial \boldsymbol{y}_K^t}\left(\frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\mathrm{d}\boldsymbol{s}_K^t}{\mathrm{d}\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{\theta}_l}\right) \tag{4}$$

where the summation ranges from the first time step $t = 1$ until the last time step $t = T$. We further expand (4) in the following, in particular $((\mathrm{d}\boldsymbol{s}_K^t)/(\mathrm{d}\boldsymbol{\theta}_l))$, and isolate the temporal components involving only layer l. This allows to unravel a time recursion that can be exploited to form an online reformulation of BPTT, see Appendix B. In particular, it can be shown that

$$\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_{1\le\hat{t}\le t}\left(\prod_{t\ge t'>\hat{t}} \frac{\mathrm{d}\boldsymbol{s}_l^{t'}}{\mathrm{d}\boldsymbol{s}_l^{t'-1}}\right)\left(\frac{\partial \boldsymbol{s}_l^{\hat{t}}}{\partial\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_l^{\hat{t}}}{\partial \boldsymbol{y}_l^{\hat{t}-1}}\frac{\partial \boldsymbol{y}_l^{\hat{t}-1}}{\partial \boldsymbol{\theta}_l}\right). \tag{5}$$

Equation (5) can be rewritten in a recursive form introducing a so-called eligibility tensor $\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l}$ as

$$\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} := \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{\theta}_l} = \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\boldsymbol{\theta}_l} + \left(\frac{\partial \boldsymbol{s}_l^t}{\partial\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{y}_l^{t-1}}\frac{\partial \boldsymbol{y}_l^{t-1}}{\partial\boldsymbol{\theta}_l}\right) \tag{6}$$

see Appendix B for a detailed proof. This leads to an expression of the gradient as

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_t\left(\mathbf{L}_l^t\mathbf{e}_l^{t,\boldsymbol{\theta}_l} + \mathbf{R}\right) \tag{7}$$

where

$$\mathbf{e}_l^{t,\boldsymbol{\theta}_l} = \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{s}_l^t}\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{\theta}_l} \tag{8}$$

$$\mathbf{L}_l^t = \frac{\partial E^t}{\partial \boldsymbol{y}_K^t}\left(\prod_{(K-l+1)>m\ge 1}\frac{\partial \boldsymbol{y}_{K-m+1}^t}{\partial \boldsymbol{s}_{K-m+1}^t}\frac{\partial \boldsymbol{s}_{K-m+1}^t}{\partial \boldsymbol{y}_{K-m}^t}\right) \tag{9}$$

$$\boldsymbol{R} = \frac{\mathrm{d}E^t}{\mathrm{d}\boldsymbol{y}_K^t}\frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\left(\sum_{0\le k'}^{K-1}\left(\prod_{0\le m}^{k'}\frac{\partial \boldsymbol{s}_{K-m}^t}{\partial \boldsymbol{y}_{K-m-1}^t}\frac{\partial \boldsymbol{y}_{K-m-1}^t}{\partial \boldsymbol{s}_{K-m-1}^t}\right)\right. \\ \left.\delta_{n',K-k'}\delta_{t',t}\frac{\mathrm{d}\boldsymbol{s}_{n'}^{t'}}{\mathrm{d}\boldsymbol{s}_{n'}^{t'-1}}\boldsymbol{\epsilon}_{K-k',l}^{t'-1,\boldsymbol{\theta}_l}\right). \tag{10}$$

Following the biological inspiration, the eligibility trace $\mathbf{e}_l^{t,\boldsymbol{\theta}_l}$ contains only the temporal component of the gradient computation. It evolves through the recursion (8) that involves only information local to a layer of spiking neurons. The learning signal $\mathbf{L}_l^t \in I\!R^{n_l}$ is applied locally in time and corresponds to the spatial gradient component. Note that the learning signal can also be expressed recursively in space, i.e., from one layer to another, as

$$\mathbf{L}_l^t = \mathbf{L}_{l+1}^t\left(\frac{\partial \boldsymbol{y}_{l+1}^t}{\partial \boldsymbol{s}_{l+1}^t}\frac{\partial \boldsymbol{s}_{l+1}^t}{\partial \boldsymbol{y}_l^t}\right) \quad \text{with} \quad \mathbf{L}_K^t = \frac{\partial E^t}{\partial \boldsymbol{y}_K^t}. \tag{11}$$

The residual term $\mathbf{R}$ defined in (10), represents the combinations of spatial gradients, involving different layers, and temporal gradients, involving different time steps.

The formulation described by (7) is novel and constitutes the basis for deriving our online learning scheme. The main emphasis of OSTL is to maintain the separation between temporal and spatial gradients. Therefore, we explore an approach in which we strictly consider only the temporal and spatial components of (7). This approximation leads to the following formulation of OSTL for deep SNNs:

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} \approx \textstyle\sum_t \mathbf{L}_l^t\mathbf{e}_l^{t,\boldsymbol{\theta}_l}. \tag{12}$$

Thus, $(\mathrm{d}E/(\mathrm{d}\boldsymbol{\theta}_l))$ is in principle calculated as the sum of individual products of the eligibility traces and the learning signal, as schematically shown in Fig. 2(c).

A. OSTL for Deep Feed-Forward SNNs

After the main formalism of OSTL has been introduced in (12), we provide the explicit formulation for the deep feed-forward SNU network described in Section II. In this case, the recurrency matrix $\mathbf{H}_l$ is absent in (1). Using the diagonal function $\mathrm{diag}(\boldsymbol{x})$ that puts the components of the vector $\boldsymbol{x}$ onto the diagonal of a zero matrix, i.e., $\mathrm{diag}(\boldsymbol{x})_{i,j} = x_i\delta_{i,j}$, (8) becomes

$$\mathbf{e}_l^{t,\mathbf{W}_l} = \mathrm{diag}\left(\mathbf{h'}_l^t\right)\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l}, \tag{13}$$

$$\mathbf{e}_l^{t,\mathbf{b}_l} = \mathrm{diag}\left(\mathbf{h'}_l^t\right)\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l} + \mathrm{diag}\left(\mathbf{h'}_l^t\right) \tag{14}$$

where

$$\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l} = \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l} + \mathrm{diag}\left(\mathbf{g'}_l^t\right)\mathrm{diag}\left(\boldsymbol{\Upsilon}_{l-1}^t\right) \tag{15}$$

$$\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l} = \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l} - d\cdot\mathrm{diag}\left(\mathbf{g'}_l^t\right)\mathrm{diag}\left(\boldsymbol{s}_l^{t-1}\odot\mathbf{h'}_l^{t-1}\right) \tag{16}$$

and

$$\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}} = \mathrm{diag}\left(\mathbf{g'}_l^t\right)d\cdot\mathrm{diag}\left(\left(\mathbb{1}-\boldsymbol{y}_l^{t-1}\right)-\boldsymbol{s}_l^{t-1}\odot\mathbf{h'}_l^{t-1}\right). \tag{17}$$

Note that we have used the short-hand notation of $((\mathrm{d}\mathbf{g}(\boldsymbol{\xi}_l^t))/(\mathrm{d}\boldsymbol{\xi}_l^t)) = \mathbf{g'}_l^t$, $((\mathrm{d}\mathbf{h}(\boldsymbol{\xi}_l^t))/(\mathrm{d}\boldsymbol{\xi}_l^t)) = \mathbf{h'}_l^t$. Furthermore, the elements of $\boldsymbol{\Upsilon}_l^t$ are given by $(\Upsilon_l^t)_{\mathrm{opq}} = \delta_{o,p}(y_l^t)_q$.

For a mean squared error loss with readout weights

$$E^t = \frac{1}{2}\sum_{i=1}^{n_K}\left(\left(\mathbf{W}^{\mathrm{out}}\boldsymbol{y}_K^t\right)_i - \left(\hat{\boldsymbol{y}}^t\right)_i\right)^2 \tag{18}$$

where $\mathbf{W}^{\mathrm{out}} \in I\!R^{n_K\times n_K}$ are the readout weights, the learning signal stated in (9) is

$$\mathbf{L}_l^t = \mathbf{L}_{l+1}^t\left(\mathrm{diag}\left(\mathbf{h'}_{l+1}^t\odot\mathbf{g'}_{l+1}^t\right)\mathbf{W}_{l+1}\right) \tag{19}$$

with

$$\mathbf{L}_K^t = \left(\mathbf{W}^{\mathrm{out}}\right)^\top\boldsymbol{y}_K^t - \hat{\boldsymbol{y}}^t. \tag{20}$$

Algorithm 1 shows the steps necessary to compute the parameter updates for a feed-forward SNN architecture. The updates may be applied immediately or deferred, similar to RTRL. Note that in the sequel we focus on the latter case. Clearly, a feed-forward configuration of SNNs does not prevent solving temporal tasks. In such architectures, including also quasi-RNNs [28], the network relies on the internal states of the units rather than on layerwise recurrency matrices $\mathbf{H}_l$. The derivation of an SNN that involves recurrency matrices $\mathbf{H}_l$ can be found in Appendix B.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 1 OSTL for Deep Feed-Forward SNNs (K Layers) With Mean Squared Error Loss

```
for $1 \le t \le T$ do
    for $1 \le l \le K$ do
        Compute $s_l^t$ using (1) with $y_{l-1}^t, s_l^{t-1}, y_l^{t-1}, \theta_l$
        Compute $y_l^t$ using (2) with $s_l^t, \theta_l$
        Compute $\epsilon_l^{t,\theta}$ using (15), (16) with $\epsilon_l^{t-1,\theta}, s_l^t, y_l^t, \theta_l$
        Compute $\mathbf{e}_l^{t,\theta}$ using (13), (14) with $\epsilon_l^{t,\theta}, s_l^t, y_l^t, \theta_l$
    end for
    Compute $E^t$ using (18) with $y_K^t, \hat{y_K}^t$
    for $1 \le l \le k$ do
        Compute $\mathbf{L}_l^t$ using (9)
        Compute $\mathrm{d}E^t = -\eta\left(\mathbf{L}_l^t \mathbf{e}_l^{t,\theta}\right)$
        Update $\theta_l$ with $\mathrm{d}E^t$ [Online updates]
        Accumulate $\Delta\theta_l = \Delta\theta_l + \mathrm{d}E^t$ [Deferred updates]
    end for
end for
for $1 \le l \le K$ do
    Update $\theta_l$ with accumulated $\Delta\theta_l$ [Deferred updates]
end for
```

B. OSTL for Single-Layer SNNs

In case of a single-recurrent layer, there is no residual term $\mathbf{R}$ emerging in (7), that is $\mathbf{R} = 0$, see Appendix B. For this architecture, the additional equations needed besides (13)–(15) are

$$\mathbf{e}^{t,\mathbf{H}} = \mathrm{diag}\left(\mathbf{h}'^t\right)\epsilon^{t,\mathbf{H}}, \tag{21}$$

$$\epsilon^{t,\mathbf{H}} = \frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}}\epsilon^{t-1,\mathbf{H}} + \mathrm{diag}\left(\mathbf{g}'^t\right)\mathrm{diag}\left(\Upsilon^{t-1}\right) \tag{22}$$

$$\epsilon^{t,\mathbf{b}} = \frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}}\epsilon^{t-1,\mathbf{b}} + \mathbf{H}\mathrm{diag}\left(\mathbf{h}'^{t-1}\right) - d\cdot\mathrm{diag}\left(\mathbf{g}'^t\right)\mathrm{diag}\left(s^{t-1}\odot\mathbf{h}'^{t-1}\right) \tag{23}$$

and

$$\frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}} = \mathrm{diag}\left(\mathbf{g}'^t\right)\left(\mathbf{H}\mathrm{diag}\left(\mathbf{h}'^{t-1}\right) + d\cdot\mathrm{diag}\left(\left(\mathbb{1} - y^{t-1}\right) - s^{t-1}\odot\mathbf{h}'^{t-1}\right)\right). \tag{24}$$

Using the loss from (18), the learning signal is

$$\mathbf{L}^t = \frac{\partial E^t}{\partial y^t} = \left(\mathbf{W}^{\mathrm{out}}\right)^\top\left(y^t - \hat{y}^t\right). \tag{25}$$

Note that in this special case of a single-layer network the subscript l, signifying the layer has been removed. Because $\mathbf{R} = 0$, there are no approximations involved and OSTL is gradient-equivalent to BPTT. See also the formal proof in Appendix B. Moreover, the gradient equivalence is maintained even if this recurrent layer is embedded in a deep network architecture comprising an arbitrary number of stateless layers, e.g., a series of convolutions followed by the recurrent layer and a softmax output layer, see Appendix D. This important property has also been demonstrated via simulations.

C. Biologically Inspired Approximations of OSTL

Although the eligibility traces are biologically inspired and involve only information local to a layer, the learning signal relies on the mathematical concept of error backpropagation applied locally in time. It is possible to further study and introduce approximations into both components independently. Here, we outline a few examples of such approaches, which also highlight the benefits of the clear separation of the gradient components.

For example, approximate eligibility traces may be introduced. Their exact formulation for recurrent network architectures has been given in (21)–(24) and involves computations with the recurrent matrix $\mathbf{H}$, which requires information that is local to one layer, but nonlocal to an individual neuron. In addition, these computations dominate to a large extent the time complexity of OSTL, as discussed in detail in Appendix C. To remove the need to transmit this information, one could neglect the influence of the recurrent matrix $\mathbf{H}$ in the eligibility traces, leading to an approximation of OSTL, denoted OSTL *w/o* $\boldsymbol{H}$, similar to [21]. Although this scheme may lead to performance degradation in certain tasks, it substantially reduces the time complexity of OSTL from $O(Kn^4)$ down to $O(Kn^2)$, see Appendix C.

Further biological constraints may be introduced into the spatial gradient components, i.e., the learning signals. Specifically, the learning signal $\mathbf{L}_l^t$ of OSTL uses the same weight matrix W_{l+1}, see (19), to transport information, as in the forward pass. A common concern in computational neuroscience is that signals in biological substrates should not use the same synaptic connection in both directions. To address this concern, OSTL can utilize schemes, such as feedback alignment [29], direct feedback alignment, or indirect feedback alignment [30]. In feedback alignment, the learning signals are delivered via separate connections from each layer to the previous one, using random matrices. Considering the mean squared error loss with readout weights given in (18), the learning signals become

$$\mathbf{L}_l^t = \mathbf{L}_{l+1}^t\left(\mathrm{diag}\left(\mathbf{h}'^t_{l+1}\odot\mathbf{g}'^t_{l+1}\right)\mathbf{B}_{l+1}\right) \tag{26}$$

with

$$\mathbf{L}_K^t = \left(\mathbf{W}^{\mathrm{out}}\right)^\top y_K^t - \hat{y}^t \tag{27}$$

where $\mathbf{B}_{l+1}$ is a random matrix connecting the layer $l+1$ to layer l. Because of the random nature of this approximation, we refer to this algorithm as OSTL *rnd*.

The aforementioned examples highlight that OSTL allows to easily incorporate more biological constraints, and thereby, even bridge different approaches described in the literature.

D. Generalization of OSTL to Generic RNNs

The derivation of OSTL for SNUs implicitly demonstrates the applicability of OSTL to generic RNNs, as SNUs incorporate the LIF dynamics of SNNs into RNNs. However, in the sequel, we explicitly extend the OSTL framework to a wider variety of deep RNNs, for example, networks consisting of LSTM units. We note that such units obey more complex state and output equations compared with SNNs. In particular, the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

output $\boldsymbol{y}_l^t$ is a recursive function and depends on the outputs of the previous layer and the trainable parameters, that is,

$$\boldsymbol{y}_l^t = \boldsymbol{\chi}\left(\boldsymbol{s}_l^t, \boldsymbol{y}_l^{t-1}, \boldsymbol{y}_{l-1}^t, \boldsymbol{\theta}_l\right) \quad (28)$$

$$\boldsymbol{s}_l^t = \boldsymbol{\psi}\left(\boldsymbol{s}_l^{t-1}, \boldsymbol{y}_l^{t-1}, \boldsymbol{y}_{l-1}^t, \boldsymbol{\theta}_l\right). \quad (29)$$

To the best of our knowledge, this generalized form covers all standard RNN units currently used in the literature, including sophisticated SNU variants [31].

Performing similar steps as in Sections III-A–III-C, we derive the eligibility trace for layer l as

$$\mathbf{e}_l^{t,\boldsymbol{\theta}_l} = \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{s}_l^t}\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{y}_l^{t-1}}\mathbf{e}_l^{t-1,\boldsymbol{\theta}_l}. \quad (30)$$

Note that compared with the case of SNNs, the eligibility trace contains an additional recurrent term, which is the eligibility trace of the previous time step $t-1$, i.e., $\mathbf{e}_l^{t-1,\boldsymbol{\theta}_l}$. Hence, these formulations expand the concept of the eligibility traces of a particular neuron to depend also on the activities of other neurons within the same layer. Nevertheless, the core function of the eligibility traces, in capturing the temporal gradient contributions, is maintained.

The generalized learning signal is

$$\mathbf{L}_l^t = \mathbf{L}_{l+1}^t\left(\frac{\partial \boldsymbol{y}_{l+1}^t}{\partial \boldsymbol{s}_{l+1}^t}\frac{\partial \boldsymbol{s}_{l+1}^t}{\partial \boldsymbol{y}_l^t} + \frac{\partial \boldsymbol{y}_{l+1}^t}{\partial \boldsymbol{y}_l^t}\right) \quad \text{with} \quad \mathbf{L}_K^t = \frac{\partial E^t}{\partial \boldsymbol{y}_K^t}. \quad (31)$$

Finally, similar to the case of deep SNNs, the parameter update $(\mathrm{d}E/(\mathrm{d}\boldsymbol{\theta}_l))$ is computed using (12). Therefore, the OSTL extension described in (30) and (31) enables online training for a broad variety of RNN units, including LSTMs and GRUs, for which the explicit formulas can be found in Appendixes E and F, respectively.

IV. Comparison With Related Algorithms

Table I summarizes the properties of various online learning algorithms, including OSTL. The horizontal lines delimit three groups: the first group includes the fundamental algorithms, i.e., BPTT and RTRL, the second one the recently introduced approximate algorithms, and the third one the various OSTL versions. OSTL and BPTT provide a clear way to be applied to deep networks, whereas RFLO or E-prop require additional workarounds, such as layerwise direct connections to the output layer. For algorithms extendable to training K-layered networks, the memory and time complexity typically scale linearly with the number of layers K.

Two algorithms from the prior art, namely, RTRL and E-prop, share important concepts with OSTL, and therefore, deserve further discussion. OSTL and RTRL are derived by optimizing the same objective function using the gradient descent principles, but the explicit treatment of the gradient separation in OSTL leads to a different factorization of the operations that rely on layer local, rather than global information. In other words, the eligibility traces and learning signals of OSTL are defined on a per-layer basis, rather than on a complete network basis. Thus, for deep networks, OSTL scales with the size of the largest layer, whereas RTRL with the total number of neurons of a global pool of neurons.

Although the generic form of (12) appears to be similar to E-prop, there are several critical differences. First, the eligibility traces and learning signals of OSTL described in (8) and (9) have been derived without approximations and are different than the corresponding ones in E-prop ([13, eqs. (4) and (13)]). Specifically, OSTL uses the total derivative $((\mathrm{d}\boldsymbol{s}_l^t)/(\mathrm{d}\boldsymbol{s}_l^{t-1}))$, while E-prop uses the partial derivative $((\partial \boldsymbol{s}_l^t)/(\partial \boldsymbol{s}_l^{t-1}))$.

This subtle, yet crucial difference leads to different learning results. For example, in a recurrently connected layer of SNUs described by (1) and (2), the differences between the total derivative in (24) and the partial derivative given below (taken from [13])

$$\frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{s}_l^{t-1}} = \mathrm{diag}\left(\mathbf{g}'^t_l\right)\left(d \cdot \mathrm{diag}\left(\mathbb{1} - \boldsymbol{y}_l^{t-1}\right)\right) \quad (32)$$

become apparent, as the terms involving $((\mathrm{d}\boldsymbol{y}_l^t(\boldsymbol{s}_l^t))/(\mathrm{d}\boldsymbol{s}_l^t))$ are missing.

Moreover, the definition of the learning signal in OSTL is also different from the one in E-prop. OSTL builds on a clear separation of spatial and temporal gradients without any approximations. Therefore, the learning signal $\mathbf{L}_l^t$ of OSTL, stated in (25), is defined as the exact gradient signal from the output layer w.r.t. all layers of the network. In contrast, E-prop mostly focuses on single-layer networks and defines the learning signal following the general form given in [13, eq. (4)]

$$\mathbf{L}_{\text{E-prop}}^t = \mathbf{B}\left(\boldsymbol{y}^t - \boldsymbol{y}^{*,t}\right) \quad (33)$$

where $\mathbf{B}$, depending on the variant of E-prop, is either the matrix of output weights or a random weight matrix. The difference of the learning signal is especially predominant in the case of a deep network, where in E-prop every neuron in each layer is required to have a direct connection to the output layer, while the learning signal of OSTL is defined for every neuron, even in layers without a direct connection to the output layer.

V. Results

We evaluated the performance of OSTL on four tasks: music prediction, handwritten digit classification, language modeling, and speech recognition. For each task, we compared the accuracy of OSTL with that of BPTT-based training and to E-prop, where possible. To ensure repeatability of all results, we performed simulations with five random network initializations and report their mean and standard deviation. In all our runs, the fixed random feedback matrix in OSTL *rnd* and in E-prop was initialized with random numbers according to an univariate Gaussian distribution with mean $\mu = 0$ and variance var $= 1$. We also evaluated a symmetric variant of E-prop (E-prop *symm*), which uses the same output matrices for the forward and the backward pass [13]. In all our experiments involving SNUs, i.e., $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\Theta}(\boldsymbol{x})$, we use the pseudoderivative $\mathrm{d}\boldsymbol{\Theta}(\boldsymbol{x})/\mathrm{d}\boldsymbol{x} = 1 - \mathbf{tanh}^2(\boldsymbol{x})$.

A. Music Prediction

For the music prediction task, we used the JSB dataset [32] with the standard training/testing data split, i.e., 229 music

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I

COMPARISON OF OSTL WITH ALGORITHMS FROM THE LITERATURE. THE COMPLEXITIES CORRESPOND TO A SINGLE PARAMETER UPDATE IN SINGLE-LAYER NEURAL NETWORKS. FOR ALGORITHMS EXTENDABLE TO K-LAYERED NETWORKS, THE MEMORY AND TIME COMPLEXITY SCALE LINEARLY WITH K

ALGORITHM	MEMORY COMPLEXITY	TIME COMPLEXITY	EXACT GRADIENTS (VS. BPTT)	DERIVED FOR
BPTT (UNROLLED FOR T TIME STEPS)	Tn	Tn^2	✓	RNNS
RTRL WITH DEFERRED UPDATES	n^3	n^4	✓	RNNS
RTRL	n^3	n^4	×	RNNS
UORO	n^2	n^2	×	RNNS
KF-RTRL	n^2	n^3	×	RNNS
OK-RTRL (FOR r SUMMATION TERMS)	rn^2	rn^3	×	RNNS
RFLO	n^2	n^2	×	SNNS & RNNS
SUPERSPIKE (FOR FINITE TEMPORAL INTERVAL t [20])	n^2	tn^2	×	SNNS
E-PROP	n^2	n^2	×	SNNS & RNNS
OSTL: FEED-FORWARD SNNS	n^2	n^2	✓	SNNS
OSTL: RECURRENT SNNS (*w/o* $\boldsymbol{H}$)	n^2	n^2	×	SNNS
OSTL: GENERIC RNNS	n^3	n^4	✓	SNNS & RNNS

pieces for training and 77 for testing. As shown in Fig. 3(a), we employed a network architecture comprising a single layer with 150 units and a stateless output layer of 88 sigmoid units. In particular, we analyzed a feed-forward SNU layer configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\Theta}(\boldsymbol{x})$, $\mathbf{g}(\boldsymbol{x}) = \mathbb{1}(\boldsymbol{x})$ and $d = 0.4$, a feed-forward sSNU layer configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})$, $\mathbf{g}(\boldsymbol{x}) = \mathbf{max}(0, \boldsymbol{x}) = \mathbf{ReL}(\boldsymbol{x})$ and $d = 0.8$, and a standard LSTM layer. The assessment was performed based on the common approach for JSB to report the negative log-likelihood, whose lower values correspond to a higher quality of predictions. We used the standard stochastic gradient descent optimizer and trained with BPTT, OSTL, OSTL *w/o* $\boldsymbol{H}$, E-prop *symm*, OSTL *rnd*, and E-prop for at least 100 epochs, or until the test performance did not improve for more than 15 epochs. For OSTL and its variants, we investigated learning rates between 10^{-5} and 10^{-2}. Our final results were obtained using learning rates of 0.0005, 0.0001, and 0.0003 for the SNU, sSNU, and LSTM network, respectively.

Fig. 3(b) summarizes the results obtained with the various training algorithms applied to the three considered network architectures. OSTL achieved results on par with those obtained with BPTT, thereby empirically validating the BPTT-gradient equivalence. Importantly, for the JSB task, OSTL provides a low-complexity $O(n^2)$ formulation for the SNU architecture.

Although the formulations of OSTL and E-prop *symm* differ, and the latter does not yield equivalent gradients, as explained in Section IV, the performance on this task was comparable. However, for approaches using a random feedback matrix, i.e., OSTL *rnd* and E-prop, the negative log-likelihood metric deteriorated consistently across LSTMs, sSNUs, and SNUs.

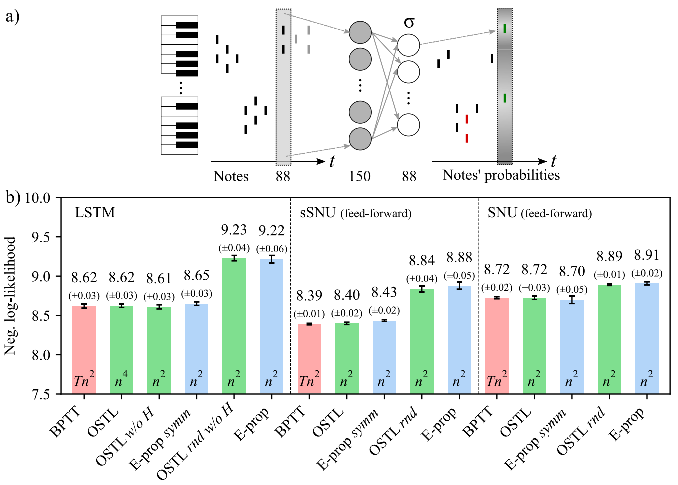


Fig. 3. **Music prediction based on the JSB dataset**. (a) Network architecture comprising a single layer of LSTMs, sSNUs, or SNUs with 150 units and a stateless output layer with 88 sigmoid units. (b) Negative log-likelihood, the lower the better, of the different network configurations trained with BPTT, OSTL, OSTL *w/o* $\boldsymbol{H}$, E-prop *symm*, and E-prop. Note that the $O(\cdot)$ time complexity is given at the base of the bars.

B. Handwritten Digit Classification

The handwritten digit classification task involved the MNIST dataset [33] with the standard training/testing split, i.e., 60 000 digits for training and 10 000 digits for testing. The gray values of the digits' pixels were transformed into spike rates using the same procedure as presented in [10] with $N_s = 20$. Fig. 4(a) shows the employed deep feed-forward network architecture comprising three layers with 256, 256, and ten units, respectively. In particular, we used SNUs configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\Theta}(\boldsymbol{x})$, $\mathbf{g}(\boldsymbol{x}) = \alpha\boldsymbol{x}$ if $\boldsymbol{x} < 0$ and $\mathbf{g}(\boldsymbol{x}) = \boldsymbol{x}$ if $\boldsymbol{x} \geq 0$, i.e., $\mathbf{LReL}_{\alpha(\boldsymbol{x})}$ with $\alpha = 0.01$ and $d = 0.9$ and sSNUs configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})$, $\mathbf{g}(\boldsymbol{x}) = \mathbf{LReL}_{\alpha(\boldsymbol{x})}$ (leaky rectified linear function) with $\alpha = 0.01$, and $d = 0.9$. During training, the correct digit label was provided at the last time step only. Thus, this setup is particularly challenging for training algorithms, because the loss function is zero in all time steps, except for the last one. We used the standard stochastic gradient descent optimizer and trained for 100 epochs with BPTT, OSTL, E-prop *symm*, and E-prop. For OSTL, we investigated learning rates between 10^{-3} and 0.5. Our final results were obtained using learning rates of 0.15 and 0.2 for the sSNU and the SNU configuration, respectively. The test classification accuracy, whose higher values correspond to a better classification, was calculated based on a commonly used approach, in which the average spiking rate for all time steps is considered, and the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

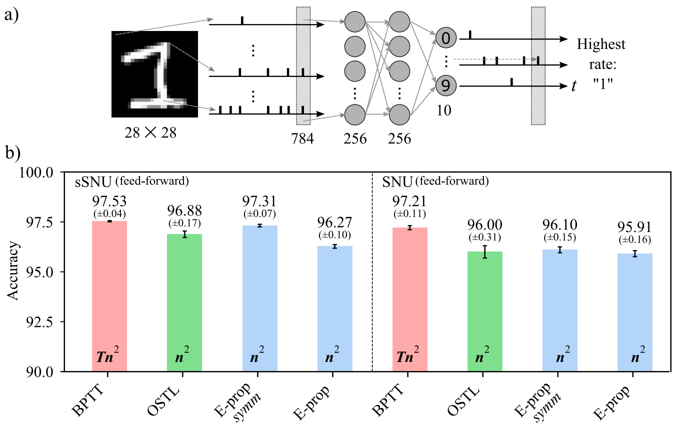


Fig. 4. **Handwritten digit classification based on the MNIST dataset**. (a) Network architecture comprising three layers of sSNUs or SNUs with 256, 256, and Ten units, respectively. (b) Classification accuracy, the higher the better, of the networks trained with BPTT, OSTL, E-prop *symm*, and E-prop. Note that the $O(\cdot)$ time complexity is given at the base of the bars.

digits from the test set are transformed into spike rates for an extended testing period of $N_s = 300$ in our case.

The results in Fig. 4(b) show that OSTL is able to cope with the time delay of the nonzero learning signal. Moreover, this task illustrates the low-complexity formulation of OSTL for deep feed-forward SNNs, i.e., $O(Kn^2)$. Although this OSTL formulation omits the residual term $\mathbf{R}$, OSTL achieved competitive classification accuracy to the BPTT baseline.

Despite extensive hyperparameter search, in many cases, E-prop had stability issues when applied to the SNUs, leading to increased variance or deteriorated accuracy.

C. Language Modeling

As the third task, we investigated language modeling based on the Penn Tree Bank (PTB) dataset [34]. This dataset comprises sequences of words corresponding to English sentences from the Wall Street Journal, restricted to the most frequent 10k tokens and split into training, validation, and test subsequences with around 930k, 74k, and 82k tokens, respectively. As shown in Fig. 5(a), the employed architecture comprises a fully connected layer of 10 000 units for embedding, three layers of 1300 sSNUs, configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})$, $\mathbf{g}(\boldsymbol{x}) = \mathbb{1}(\boldsymbol{x})$ and $d = 0.4$, and a softmax output layer of 10 000 units. The weights of the softmax were tied to the input embeddings. We trained a feed-forward architecture of sSNUs and recurrently connected sSNUs. In order to avoid overfitting, we used gradient clipping with a ratio of 3.5. We used the standard stochastic gradient descent optimizer, where the initial learning rates were multiplied by a factor of 0.5 at each epoch after the fourth epoch. For OSTL, we investigated learning rates in the range of 10^{-2} and 1.0. Our final results were obtained with an initial learning rate of 1.0 for BPTT and 0.08 for OSTL. For BPTT, we used a truncation horizon of $T = 20$ words. The assessment was performed based on the perplexity metric, whose lower values correspond to a higher quality of predictions.

In addition, in this difficult task, OSTL achieved competitive performance compared with BPTT, see Fig. 5(b). Extending the architecture with recurrent connections led to an overall

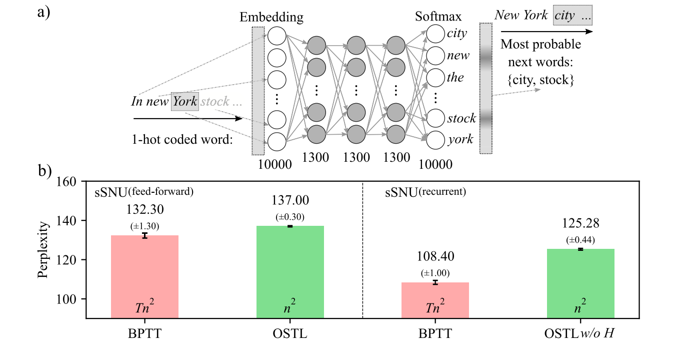


Fig. 5. **Language modeling based on the Penn Tree Bank dataset**. (a) Network architecture comprising a fully connected layer of 10 000 units for embedding, three layers with 1300 each of sSNUs and SNUs, and a softmax output layer of 10 000 units. (b) Test perplexities, the lower the better, for the two network architectures trained with BPTT and OSTL. Note that the $O(\cdot)$ time complexity is given at the base of the bars.

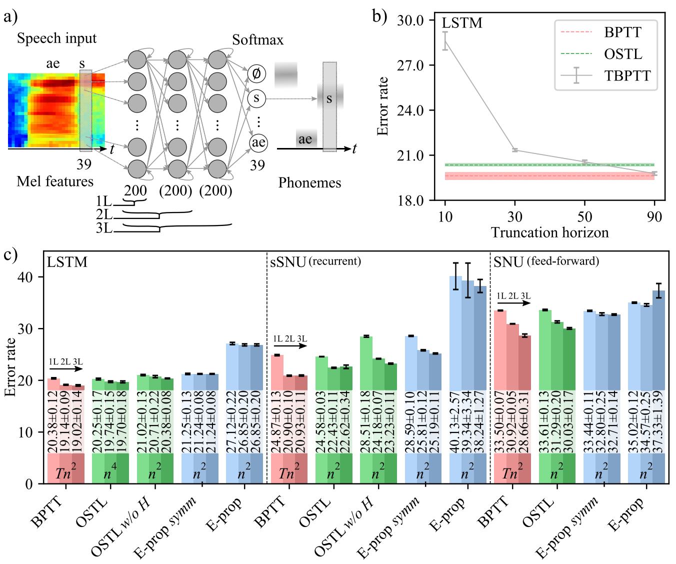


Fig. 6. **Speech recognition based on the TIMIT dataset**. (a) Network architecture comprising one, two or three layers with 200 units each of LSTMs, sSNUs, or feed-forward SNUs, and a softmax output layer of 39 units. (b) Error rate, the lower the better, of OSTL, BPTT, and truncated BPTT with truncation horizons T. (c) Error rate of the various network architectures and training algorithms. Each three consecutive bars in a group represent one-, two-, and three-hidden-layer architectures, respectively. Note that the $O(\cdot)$ time complexity is given at the base of the bars.

improvement of the test perplexity for BPTT and OSTL. However, the improvement was larger for BPTT than for OSTL *w/o* $\boldsymbol{H}$ which uses approximate recurrent gradients. This suggests that this task involves a complex temporal structure that benefits from using accurate gradients for the recurrent connections.

It is important to note that this deep architecture cannot be trained with E-prop because weight tying cannot be used when every hidden layer is connected to the output layer.

D. Speech Recognition

Finally, we considered an application to speech recognition based on the TIMIT dataset [35]. To this end, we used the same data preprocessing, as described in [36], but instead of 61 output classes, we used the reduced set of 39 output classes. We employed network architectures with 39 input neurons, and

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

either one, two, or three hidden layers of 200 sSNUs, feed-forward SNUs or LSTM units, and a softmax output layer of 39 units, as shown in Fig. 6(a). The sSNUs were configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})$, $\mathbf{g}(\boldsymbol{x}) = \mathbb{1}(\boldsymbol{x})$, and $d = 0.4$ and the feed-forward SNUs configured to $\mathbf{h}(\boldsymbol{x}) = \boldsymbol{\Theta}(\boldsymbol{x})$, $\mathbf{g}(\mathbf{x}) = \mathbb{1}(x)$, and $d = 0.4$. The assessment was done based on the phoneme error rate, whose lower values correspond to a higher quality of speech recognition. For OSTL, we investigated learning rates between 10^{-5} and 10^{-2}. Our final results were obtained using learning rates of 0.00057 for LSTMs, 0.00083 for sSNUs and 0.008 for the SNUs, respectively. We used the RMSProp optimizer with a decay of 0.95, an epsilon of 10^{-8} and trained for 24 epochs with BPTT, truncated BPTT (TBPTT), OSTL, and E-prop.

We first note that this task requires capturing long temporal dependencies between the inputs and the outputs, which is especially challenging for TBPTT. TBPTT reduces the amount of information that needs to be stored by splitting the input stream into shorter segments and by updating the parameters of the network after each segment [37]. Therefore, TBPTT encounters learning deficiencies in tasks, where there are long time lags between the network activity and the feedback from the environment. We analyzed this by varying the truncation horizon $T = \{10, 30, 50, 90\}$. The performance of TBPTT was then compared with OSTL and BPTT with complete unrolling, see Fig. 6(b). The decreasing performance of TBPTT with shorter truncation horizons T indicates that the substantial overhead of unrolling the network far into the past cannot be avoided for this temporal dataset. OSTL, in contrast, despite the lack of full BPTT-equivalence for this network architecture, demonstrates competitive performance without unrolling.

We extensively investigated BPTT, OSTL, E-prop, and their variants. In particular, we investigated whether the training algorithms could cope with an increasing number of hidden layers, whether OSTL can provide competitive performance while omitting the recurrent matrix $\mathbf{H}$, thus significantly reducing the complexity and what effects are caused by the nonexact nature of E-prop. The detailed results of these studies are presented in Fig. 6(c). First, we found that the gradient equivalence of OSTL and BPTT is again empirically validated by comparing the results of the 1L network architectures. Second, OSTL is capable of dealing with multiple layers and the performance improves similar to the BPTT baseline. Third, despite omitting the recurrent matrix $\mathbf{H}$ for the eligibility traces, OSTL *w/o* $\boldsymbol{H}$ does not suffer significant performance degradation, while the time complexity is reduced substantially to $O(Kn^2)$. These results demonstrate the importance of an explicit derivation of online learning algorithms for deep network architectures, as provided by OSTL.

VI. Conclusion

OSTL is a novel online learning algorithmic framework, whose basic form provides gradients equivalent to those of BPTT and RTRL with deferred updates. However, in contrast to these algorithms, it specifically focuses on the biologically inspired separation of spatial and temporal gradient components, which facilitates studies of credit assignment schemes and further development of approximate online learning algorithms. In particular, OSTL lends itself to efficient online training of K-layer feed-forward SNU-based networks with time complexity of $O(Kn^2)$, where the update of the synaptic weights is decomposed into a learning signal and an eligibility trace. Even more so, it enables online training of shallow feed-forward SNNs with BPTT-equivalent gradients and time complexity of $O(n^2)$. Moreover, in the case of recurrent single-layer SNNs by omitting the influence of the $\mathbf{H}$ matrix in the eligibility traces the complexity remains at $O(n^2)$, without substantial loss in performance. Finally, OSTL has been further generalized to deep RNNs comprising spiking neurons or more complex units, such as LSTMs and GRUs, demonstrating competitive accuracy in comparison with BPTT. The proposed algorithmic framework allows efficient online training for temporal data and opens a new avenue for the adoption of trainable recurrent networks in the low-power IoT and edge AI devices.

Appendix A

Table II contains a glossary of symbols and notations.

Appendix B
Detailed Derivations of OSTL for SNNs

A. BPTT Gradient Equivalence for Single-Layer Networks

The state and the output equations (1) and (2) of a single-layer spiking neural network (SNN) can be generalized to

$$\begin{aligned} \boldsymbol{s}^t &= \boldsymbol{\psi}\left(\boldsymbol{y}_0^t, \boldsymbol{s}^{t-1}, \boldsymbol{y}^{t-1}, \boldsymbol{\theta}\right) \\ \boldsymbol{y}^t &= \boldsymbol{\chi}^t\left(\boldsymbol{s}^t, \boldsymbol{\theta}\right). \end{aligned}$$

The starting point for the derivation is the calculation of the derivative of the loss function E with respect to the parameters $\boldsymbol{\theta}$. Based on the chain rule, we express $\mathrm{d}E/\mathrm{d}\boldsymbol{\theta}$ as

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} = \sum_{1\le t\le T} \frac{\partial E^t}{\partial \boldsymbol{y}^t}\left(\frac{\partial \boldsymbol{y}^t}{\partial \boldsymbol{s}^t}\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{y}^t}{\partial \boldsymbol{\theta}}\right). \tag{34}$$

We will prove that

$$\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{\theta}} = \sum_{1\le \hat{t}\le t}\left(\prod_{t\ge t'>\hat{t}} \frac{\mathrm{d}\boldsymbol{s}^{t'}}{\mathrm{d}\boldsymbol{s}^{t'-1}}\right)\left(\frac{\partial \boldsymbol{s}^{\hat{t}}}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{s}^{\hat{t}}}{\partial \boldsymbol{y}^{\hat{t}-1}}\frac{\partial \boldsymbol{y}^{\hat{t}-1}}{\partial \boldsymbol{\theta}}\right). \tag{35}$$

To validate the equality in (35), we resort to a proof by induction. The induction basis for the first step of $t = 1$ for the left-hand side becomes

$$\left.\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{\theta}}\right|_{t=1} = \frac{\mathrm{d}\boldsymbol{s}^1}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \boldsymbol{s}^1}{\partial \boldsymbol{\theta}} \tag{36}$$

and for the right-hand side

$$\begin{aligned} &\left.\sum_{1\le \hat{t}\le t}\left(\prod_{t\ge t'>\hat{t}} \frac{\mathrm{d}\boldsymbol{s}^{t'}}{\mathrm{d}\boldsymbol{s}^{t'-1}}\right)\left(\frac{\partial \boldsymbol{s}^{\hat{t}}}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{s}^{\hat{t}}}{\partial \boldsymbol{y}^{\hat{t}-1}}\frac{\partial \boldsymbol{y}^{\hat{t}-1}}{\partial \boldsymbol{\theta}}\right)\right|_{t=1} \\ &\qquad = \sum_{1\le \hat{t}\le 1}\left(\frac{\partial \boldsymbol{s}^{\hat{t}}}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{s}^{\hat{t}}}{\partial \boldsymbol{y}^{\hat{t}-1}}\frac{\partial \boldsymbol{y}^{\hat{t}-1}}{\partial \boldsymbol{\theta}}\right) = \frac{\partial \boldsymbol{s}^1}{\partial \boldsymbol{\theta}}. \end{aligned} \tag{37}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
GLOSSARY OF COMMONLY USED SYMBOLS AND NOTATIONS

SYMBOL OR NOTATION	DESCRIPTION
$x \in I\!R$	Scalar variable x
$\boldsymbol{x} \in I\!R^n$	Vector variables are indicated with bold symbols
$\boldsymbol{M} \in I\!R^{n \times m}$	Matrix variables are indicated with capital bold symbols
$\boldsymbol{s}_l^t, \boldsymbol{y}_l^t$	Hidden state and output of the RNN units in layer l at time t
$\boldsymbol{\theta}_l$	Variable containing all trainable parameters of the layer l
$\Theta(\cdot), \sigma(\cdot)$	Heaviside and sigmoid function
$\psi(\cdot), \chi(\cdot), \phi(\cdot)$	Generic functions to compute the state and the output of the neurons as well as the error of the network
$\frac{\mathrm{d}\boldsymbol{y}_l^t}{\mathrm{d}\boldsymbol{s}_l^t}$	Total derivative of $\boldsymbol{y}_l^t$ with respect to $\boldsymbol{s}_l^t$
$\frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{s}_l^t}$	Partial derivative of $\boldsymbol{y}_l^t$ with respect to $\boldsymbol{s}_l^t$
$\delta_{i,j}$	Kronecker-Delta: 1 if $i = j$ and 0 if $i \neq j$
$\mathrm{diag}(\mathbf{x})$	Diagonal function creating a diagonal matrix from the vector $\mathbf{x}$, i.e., $\mathrm{diag}(\mathbf{x})_{i,j} = x_i \delta_{i,j}$
$O(1), O(n), O(n^2)$	O-Notation to indicate constant, linear and squared complexity
$\boldsymbol{x} \odot \boldsymbol{y}$	Element-wise multiplication of the vector $\boldsymbol{x}$ with the vector $\boldsymbol{y}$
$\boldsymbol{xM}$	Matrix multiplication of the vector $\boldsymbol{x}$ with the matrix $\boldsymbol{M}$

Having verified the induction basis, the induction step needs to be investigated

$$\frac{\mathrm{d}\boldsymbol{s}^{t+1}}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}\boldsymbol{s}^{t+1}}{\mathrm{d}\boldsymbol{s}^{t}}\frac{\mathrm{d}\boldsymbol{s}^{t}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{t+1}}{\partial\boldsymbol{y}^{t}}\frac{\partial\boldsymbol{y}^{t}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{t+1}}{\partial\boldsymbol{\theta}} \tag{38}$$

$$\begin{aligned} &= \frac{\mathrm{d}\boldsymbol{s}^{t+1}}{\mathrm{d}\boldsymbol{s}^{t}}\left(\sum_{1\le\hat{t}\le t}\left(\prod_{t\ge t'>\hat{t}}\frac{\mathrm{d}\boldsymbol{s}^{t'}}{\mathrm{d}\boldsymbol{s}^{t'-1}}\right)\left(\frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{y}^{\hat{t}-1}}\frac{\partial\boldsymbol{y}^{\hat{t}-1}}{\partial\boldsymbol{\theta}}\right)\right) \\ &\quad + \frac{\partial\boldsymbol{s}^{t+1}}{\partial\boldsymbol{y}^{t}}\frac{\partial\boldsymbol{y}^{t}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{t+1}}{\partial\boldsymbol{\theta}} \\ &= \left(\sum_{1\le\hat{t}\le t}\left(\prod_{(t+1)\ge t'>\hat{t}}\frac{\mathrm{d}\boldsymbol{s}^{t'}}{\mathrm{d}\boldsymbol{s}^{t'-1}}\right)\left(\frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{y}^{\hat{t}-1}}\frac{\partial\boldsymbol{y}^{\hat{t}-1}}{\partial\boldsymbol{\theta}}\right)\right) \\ &\quad + \frac{\partial\boldsymbol{s}^{t+1}}{\partial\boldsymbol{y}^{t}}\frac{\partial\boldsymbol{y}^{t}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{t+1}}{\partial\boldsymbol{\theta}} \\ &= \sum_{1\le\hat{t}\le(t+1)}\left(\prod_{(t+1)\ge t'>\hat{t}}\frac{\mathrm{d}\boldsymbol{s}^{t'}}{\mathrm{d}\boldsymbol{s}^{t'-1}}\right)\left(\frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{y}^{\hat{t}-1}}\frac{\partial\boldsymbol{y}^{\hat{t}-1}}{\partial\boldsymbol{\theta}}\right). \end{aligned} \tag{39}$$

By comparing (39) with the hypothesis in (35) for $t+1$, it is evident that this induction step is also fulfilled, which completes the proof.

Therefore, we can expand (4) to

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} = \sum_{1\le t\le T}\frac{\partial E^t}{\partial\boldsymbol{y}^t}\left(\frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{s}^t}\sum_{1\le\hat{t}\le t}\left(\prod_{t\ge t'>\hat{t}}\frac{\mathrm{d}\boldsymbol{s}^{t'}}{\mathrm{d}\boldsymbol{s}^{t'-1}}\right)\left(\frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^{\hat{t}}}{\partial\boldsymbol{y}^{\hat{t}-1}}\frac{\partial\boldsymbol{y}^{\hat{t}-1}}{\partial\boldsymbol{\theta}}\right) + \frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{\theta}}\right). \tag{40}$$

We now analyze the term $((\mathrm{d}\boldsymbol{s}^t)/(\mathrm{d}\boldsymbol{\theta}))$ in (35). At time $t+1$, the term $((\mathrm{d}\boldsymbol{s}^{t+1})/(\mathrm{d}\boldsymbol{\theta}))$ can be obtained by multiplying $((\mathrm{d}\boldsymbol{s}^t)/(\mathrm{d}\boldsymbol{\theta}))$ by $((\mathrm{d}\boldsymbol{s}^{t+1})/(\mathrm{d}\boldsymbol{s}^t))$ and adding the term $(((\partial\boldsymbol{s}^{t+1})/(\partial\boldsymbol{y}^t))((\partial\boldsymbol{y}^t)/(\partial\boldsymbol{\theta})) + ((\partial\boldsymbol{s}^{t+1})/(\partial\boldsymbol{\theta})))$, and thus, we obtain (38). This leads to the following recursive reformulation of (35)

$$\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{\theta}} = \left(\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{s}^{t-1}}\frac{\mathrm{d}\boldsymbol{s}^{t-1}}{\mathrm{d}\boldsymbol{\theta}} + \left(\frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{y}^{t-1}}\frac{\partial\boldsymbol{y}^{t-1}}{\partial\boldsymbol{\theta}}\right)\right). \tag{41}$$

We define the eligibility tensor $\boldsymbol{\epsilon}^{t,\boldsymbol{\theta}}$ to be

$$\boldsymbol{\epsilon}^{t,\boldsymbol{\theta}} := \frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{\theta}} \tag{42}$$

which can also be recursively expressed as

$$\boldsymbol{\epsilon}^{t,\boldsymbol{\theta}} = \left(\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{s}^{t-1}}\boldsymbol{\epsilon}^{t-1,\boldsymbol{\theta}} + \left(\frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{y}^{t-1}}\frac{\partial\boldsymbol{y}^{t-1}}{\partial\boldsymbol{\theta}}\right)\right). \tag{43}$$

Note that this equation corresponds to (6) of the main text for a single layer. In the case of $t = 1$, the eligibility tensor has the special form

$$\boldsymbol{\epsilon}^{1,\boldsymbol{\theta}} = \frac{\mathrm{d}\boldsymbol{s}^1}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial\boldsymbol{s}^1}{\partial\boldsymbol{\theta}}. \tag{44}$$

Next, we rewrite (40) using (43) as

$$\begin{aligned} \frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}} &= \sum_{1\le t\le T}\frac{\partial E^t}{\partial\boldsymbol{y}^t}\left(\frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{s}^t}\left(\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{s}^{t-1}}\boldsymbol{\epsilon}^{t-1,\boldsymbol{\theta}} + \left(\frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{y}^{t-1}}\frac{\partial\boldsymbol{y}^{t-1}}{\partial\boldsymbol{\theta}}\right)\right) + \frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{\theta}}\right) \\ &= \sum_{1\le t\le T}\mathbf{L}^t\left(\frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{s}^t}\left(\frac{\mathrm{d}\boldsymbol{s}^t}{\mathrm{d}\boldsymbol{s}^{t-1}}\boldsymbol{\epsilon}^{t-1,\boldsymbol{\theta}} + \left(\frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{s}^t}{\partial\boldsymbol{y}^{t-1}}\frac{\partial\boldsymbol{y}^{t-1}}{\partial\boldsymbol{\theta}}\right)\right) + \frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{\theta}}\right) \\ &= \sum_{1\le t\le T}\mathbf{L}^t\left(\frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{s}^t}\boldsymbol{\epsilon}^{t,\boldsymbol{\theta}} + \frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{\theta}}\right) \\ &= \sum_{1\le t\le T}\mathbf{L}^t\mathbf{e}^{t,\boldsymbol{\theta}} \end{aligned} \tag{45}$$

where we define the eligibility trace $\mathbf{e}^{t,\boldsymbol{\theta}}$ and the learning signal $\mathbf{L}^t$ as

$$\mathbf{e}^{t,\boldsymbol{\theta}} := \frac{\mathrm{d}\boldsymbol{y}^t}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{s}^t}\boldsymbol{\epsilon}^{t,\boldsymbol{\theta}} + \frac{\partial\boldsymbol{y}^t}{\partial\boldsymbol{\theta}} \tag{46}$$

$$\mathbf{L}^t := \frac{\partial E^t}{\partial\boldsymbol{y}^t}. \tag{47}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

We have shown that $\mathrm{d}E/\mathrm{d}\boldsymbol{\theta}$ can be calculated as the sum of products of a learning signal (47), and an eligibility trace (46) without loss of generality. Thus, this proves that for a single layer of recurrently connected SNNs, online spatio-temporal learning (OSTL) provides gradient equivalence to backpropagation through time (BPTT). Furthermore, this equivalence is preserved even if this recurrent layer is embedded in a deep network architecture comprising an arbitrary number of stateless layers, see Appendix D.

B. Derivation of OSTL for Deep Recurrent Networks

We now turn to the more general case of a deep network, where the state and output equations are as follows:

$$\boldsymbol{s}_l^t = \boldsymbol{\psi}\left(\boldsymbol{s}_l^{t-1}, \boldsymbol{y}_l^{t-1}, \boldsymbol{y}_{l-1}^t, \boldsymbol{\theta}_l\right) \tag{48}$$

$$\boldsymbol{y}_l^t = \boldsymbol{\chi}\left(\boldsymbol{s}_l^t, \boldsymbol{\theta}_l\right) \tag{49}$$

see Section III of the main text.

Using this reformulation, (4) is generalized as follows:

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_{1\le t\le T}\frac{\mathrm{d}E^t}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_{1\le t\le T}\frac{\partial E^t}{\partial \boldsymbol{y}_K^t}\left(\frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\mathrm{d}\boldsymbol{s}_K^t}{\mathrm{d}\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{\theta}_l}\right). \tag{50}$$

For the last layer of a deep network, where $l = k$, (50) corresponds to (4) for a single layer. However, for the hidden layers, i.e., $l \neq k$, the term $((\mathrm{d}\boldsymbol{s}_K^t)/(\mathrm{d}\boldsymbol{\theta}_l))$ is expanded as follows:

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{s}_K^t}{\mathrm{d}\boldsymbol{\theta}_l} &= \frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{s}_K^{t-1}}\frac{\mathrm{d}\boldsymbol{s}_K^{t-1}}{\mathrm{d}\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_K^{t-1}}\frac{\mathrm{d}\boldsymbol{y}_K^{t-1}}{\mathrm{d}\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_{K-1}^t}\frac{\mathrm{d}\boldsymbol{y}_{K-1}^t}{\mathrm{d}\boldsymbol{\theta}_l}\\
\frac{\mathrm{d}\boldsymbol{s}_K^t}{\mathrm{d}\boldsymbol{\theta}_l} &= \underbrace{\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{s}_K^{t-1}}}_{\text{Temporal}}\left(\frac{\partial \boldsymbol{s}_K^{t-1}}{\partial \boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_K^{t-1}}{\partial \boldsymbol{s}_K^{t-2}}\cdots + \frac{\partial \boldsymbol{s}_K^{t-1}}{\partial \boldsymbol{y}_K^{t-1}}\right.\\
&\quad\left.\cdots + \underbrace{\frac{\partial \boldsymbol{s}_K^{t-1}}{\partial \boldsymbol{y}_{K-1}^{t-1}}}_{\text{Spatial}}\cdots\right) + \frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_K^{t-1}}\left(\frac{\partial \boldsymbol{y}_K^{t-1}}{\partial \boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_K^{t-1}}{\partial \boldsymbol{s}_K^{t-1}}\cdots\right)\\
&\quad + \underbrace{\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_{K-1}^t}}_{\text{Spatial}}\left(\frac{\partial \boldsymbol{y}_{K-1}^t}{\partial \boldsymbol{\theta}_l} + \underbrace{\frac{\partial \boldsymbol{y}_{K-1}^t}{\partial \boldsymbol{s}_{K-1}^{t-1}}}_{\text{Temporal}}\cdots\right).
\end{aligned} \tag{51}$$

As one can readily see, there are cross-layer dependencies involved, for example, through the terms $((\partial \boldsymbol{s}_K^{t-1})/(\partial \boldsymbol{y}_{K-1}^t))$ and $((\partial \boldsymbol{s}_K^t)/(\partial \boldsymbol{y}_{K-1}^t))$.

Similar to the case of a single-layer network we derived earlier, we also separate the gradient into an eligibility trace, which contains only temporal components, and into a learning signal, which contains only spatial components. Therefore, we define the recursive term $\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l}$ as

$$\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l} := \frac{\mathrm{d}\boldsymbol{s}_m^t}{\mathrm{d}\boldsymbol{\theta}_l} \tag{52}$$

$$\begin{aligned}
\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l} &= \frac{\partial \boldsymbol{s}_m^t}{\partial \boldsymbol{s}_m^{t-1}}\boldsymbol{\epsilon}_{m,l}^{t-1,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_m^t}{\partial \boldsymbol{y}_m^{t-1}}\left(\frac{\partial \boldsymbol{y}_m^{t-1}}{\partial \boldsymbol{s}_m^{t-1}}\boldsymbol{\epsilon}_{m,l}^{t-1,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_m^{t-1}}{\partial \boldsymbol{\theta}_l}\right)\\
&\quad + \frac{\partial \boldsymbol{s}_m^t}{\partial \boldsymbol{y}_{m-1}^t}\left(\frac{\partial \boldsymbol{y}_{m-1}^t}{\partial \boldsymbol{s}_{m-1}^t}\boldsymbol{\epsilon}_{m-1,l}^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_{m-1}^t}{\partial \boldsymbol{\theta}_l}\right) + \frac{\partial \boldsymbol{s}_m^t}{\partial \boldsymbol{\theta}_l}
\end{aligned} \tag{53}$$

with the following properties:

$$\begin{aligned}
\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} &:= \boldsymbol{\epsilon}_{l,l}^{t,\boldsymbol{\theta}_l} && (54)\\
&= \frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\boldsymbol{\theta}_l}\\
&\quad + \frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{y}_l^{t-1}}\left(\frac{\partial \boldsymbol{y}_l^{t-1}}{\partial \boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^{t-1}}{\partial \boldsymbol{\theta}_l}\right) + \frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{\theta}_l}\\
&= \left(\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\boldsymbol{\theta}_l} + \left(\frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{s}_l^t}{\partial \boldsymbol{y}_l^{t-1}}\frac{\partial \boldsymbol{y}_l^{t-1}}{\partial \boldsymbol{\theta}_l}\right)\right) && (55)\\
\mathbf{e}_l^{t,\boldsymbol{\theta}_l} &:= \frac{\mathrm{d}\boldsymbol{y}_l^t}{\mathrm{d}\boldsymbol{\theta}_l} = \left(\frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{s}_l^t}\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{\theta}_l}\right)\\
\boldsymbol{\epsilon}_{m,l}^{t<1,\boldsymbol{\theta}_l} &= \mathbf{0},\ \ \boldsymbol{\epsilon}_{l,l+1}^{t,\boldsymbol{\theta}_{l+1}} = \mathbf{0},\ \ \boldsymbol{\epsilon}_{m,l<1}^{t,\boldsymbol{\theta}_{l<1}} = \mathbf{0},\ \ \boldsymbol{\epsilon}_{m<1,l}^{t,\boldsymbol{\theta}_l} = \mathbf{0}. && (56)
\end{aligned}$$

Because the definition in (54) coincides with the eligibility trace in (6), we refer to $\boldsymbol{\epsilon}_{l,m}^{t,\boldsymbol{\theta}_l}$ as the generalized eligibility tensor. The term $\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l}$ for $m \neq l$ and $l \neq k$ contains a recursion in time, i.e., it depends on $\boldsymbol{\epsilon}_{m,l}^{t-1,\boldsymbol{\theta}_l}$, similarly as (41), but additionally it contains a recursion in space, i.e., it depends on other layers through $\boldsymbol{\epsilon}_{m-1,l}^{t,\boldsymbol{\theta}_l}$.

If we insert the term $\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l}$ from (53) into (50)

$$\begin{aligned}
&\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l}\\
&= \sum_{1\le t\le T}\frac{\partial E^t}{\partial \boldsymbol{y}_K^t}\left(\frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\boldsymbol{\epsilon}_{K,l}^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{\theta}_l}\right) && (57)\\
&\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l}\\
&= \sum_{1\le t\le T}\left(\frac{\mathrm{d}E^t}{\mathrm{d}\boldsymbol{y}_K^t}\left(\frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{s}_K^{t-1}}\boldsymbol{\epsilon}_{K,l}^{t-1,\boldsymbol{\theta}_l}\right.\right.\\
&\qquad + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_K^{t-1}}\left(\frac{\partial \boldsymbol{y}_K^{t-1}}{\partial \boldsymbol{s}_K^{t-1}}\boldsymbol{\epsilon}_{K,l}^{t-1,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_K^{t-1}}{\partial \boldsymbol{\theta}_l}\right)\\
&\qquad + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_{K-1}^t}\left(\frac{\partial \boldsymbol{y}_{K-1}^t}{\partial \boldsymbol{s}_{K-1}^t}\boldsymbol{\epsilon}_{K-1,l}^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_{K-1}^t}{\partial \boldsymbol{\theta}_l}\right)\\
&\qquad \left.\left. + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{\theta}_l}\right) + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{\theta}_l}\right) && (58)
\end{aligned}$$

the two recurrencies—$\boldsymbol{\epsilon}_{K-1,l}^{t,\boldsymbol{\theta}_l}$ in space, and $\boldsymbol{\epsilon}_{K,l}^{t-1,\boldsymbol{\theta}_l}$ in time—become apparent. Therefore, it can be seen that when expanding $\boldsymbol{\epsilon}_{K-1,l}^{t,\boldsymbol{\theta}_l}$ far enough in space, eventually terms involving $\boldsymbol{\epsilon}_{l,l}^{t,\boldsymbol{\theta}} = \boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l}$ are reached. We isolate the components involving only layer l, which allows us to rewrite (58) as

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} &= \sum_{1\le t\le T}\left(\frac{\mathrm{d}E^t}{\mathrm{d}\boldsymbol{y}_K^t}\left(\prod_{(K-l+1)>m\ge 1}\frac{\partial \boldsymbol{y}_{K-m+1}^t}{\partial \boldsymbol{s}_{K-m+1}^t}\frac{\partial \boldsymbol{s}_{K-m+1}^t}{\partial \boldsymbol{y}_{K-m}^t}\right)\right.\\
&\qquad\left.\left(\frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{s}_l^t}\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{\theta}_l}\right) + \mathbf{R}\right).
\end{aligned} \tag{59}$$

Note that all remaining terms were collected into a residual term $\mathbf{R}$ explicitly expressed in (10) of the main text.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

In addition, we define a generalized learning signal $\mathbf{L}_l^t$ and a generalized eligibility trace $\mathbf{e}_l^{t,\boldsymbol{\theta}_l}$ as

$$\mathbf{L}_l^t = \frac{\partial E^t}{\partial \boldsymbol{y}_K^t}\left(\prod_{(K-l+1)\geq m\geq 1}\frac{\partial \boldsymbol{y}_{K-m+1}^t}{\partial \boldsymbol{s}_{K-m+1}^t}\frac{\partial \boldsymbol{s}_{K-m+1}^t}{\partial \boldsymbol{y}_{K-m}^t}\right) \tag{60}$$

$$\mathbf{e}_l^{t,\boldsymbol{\theta}_l} = \left(\frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{s}_l^t}\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_l^t}{\partial \boldsymbol{\theta}_l}\right) \tag{61}$$

see (8) and (9) of the main text. This allows to express the parameter update as

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_{1\leq t\leq T}\left[\mathbf{L}_l^t\mathbf{e}_l^{t,\boldsymbol{\theta}_l} + \mathbf{R}\right] \tag{62}$$

see (7) of the main text.

Finally, by omitting the residual term $\mathbf{R}$, we arrive at the simplified approximate expression of OSTL for deep recurrent networks, given in (12) of the main text.

Appendix C
Complexity Analysis

In this section, we investigate the computational complexity of OSTL for a single-SNN layer, where the state and output equations are given by

$$\boldsymbol{s}^t = \mathbf{g}\left(\mathbf{W}\boldsymbol{y}_0^t + \mathbf{H}\boldsymbol{y}^{t-1} + d\cdot\boldsymbol{s}^{t-1}\odot\left(\mathbb{1}-\boldsymbol{y}^{t-1}\right)\right) \tag{63}$$

$$\boldsymbol{y}^t = \mathbf{h}\left(\boldsymbol{s}^t + \mathbf{b}\right) \tag{64}$$

see (1) and (2) of the main text. The majority of the computations arise from calculating the eligibility tensors as given in (43). Specifically, the term $((\mathrm{d}\boldsymbol{s}^t)/(\mathrm{d}\boldsymbol{s}^{t-1}))\boldsymbol{\epsilon}^{t-1,\boldsymbol{\theta}}$ is computationally the most intensive one, as it requires matrix multiplications. We use the index notation to explicitly investigate the complexity in terms of the number of multiplications involved. Note that the indices do not indicate a specific layer, but indicate elements of a matrix. The eligibility tensors are computed as

$$\epsilon_{opq}^{t,\boldsymbol{\theta}} = \left(\frac{\mathrm{d}s_o^t}{\mathrm{d}s_r^{t-1}}\epsilon_{rpq}^{t-1,\boldsymbol{\theta}} + \left(\frac{\partial s_o^t}{\partial\theta_{pq}} + \frac{\partial s_o^t}{\partial y_r^{t-1}}\frac{\partial y_r^{t-1}}{\partial\theta_{pq}}\right)\right) \tag{65}$$

$$\epsilon_{opq}^{t,\boldsymbol{\theta}} \sim \frac{\mathrm{d}s_o^t}{\mathrm{d}s_r^{t-1}}\epsilon_{rpq}^{t-1,\boldsymbol{\theta}}.$$

Thus, we observe

$$\epsilon_{opq}^{t,\boldsymbol{\theta}} \sim O(n) \quad\text{and}\quad \boldsymbol{\epsilon}^{t,\boldsymbol{\theta}} \sim O\left(n^4\right).$$

In particular, computing a single value of the eligibility tensor $\epsilon_{\text{opq}}^{t,\boldsymbol{\theta}}$ requires $O(n)$ multiplications, where n denotes the number of units in the layer. Therefore, the total computational cost of OSTL for shallow SNNs is $O(n^4)$ as stated in Table I of the main text. In case of a deep architecture, this complexity scales with the number of layers, i.e., $O(Kn^4)$, where K denotes the number of layers and $n = \max_{1\leq l\leq K} n_l$ denotes the maximum layer size in the network. However, if we analyze a feed-forward SNN with the state and output equations

$$\boldsymbol{s}^t = \mathbf{g}\left(\mathbf{W}\boldsymbol{y}_0^t + d\cdot\boldsymbol{s}^{t-1}\odot\left(\mathbb{1}-\boldsymbol{y}^{t-1}\right)\right) \tag{66}$$

$$\boldsymbol{y}^t = \mathbf{h}\left(\boldsymbol{s}^t + \mathbf{b}\right) \tag{67}$$

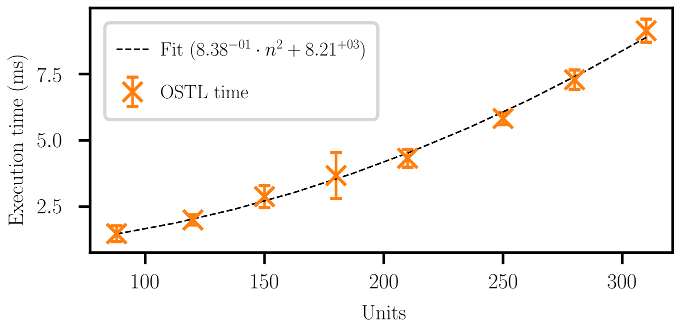


Fig. 7. **Execution time of OSTL for various layer sizes**.

the eligibility tensor $\epsilon_{\text{opq}}^{t,\boldsymbol{\theta}}$ reduces to a rank-two tensor, because the Jacobian $((\mathrm{d}s_o^t)/(\mathrm{d}s_r^{t-1}))$ reduces to a diagonal matrix as

$$\epsilon_{opq}^{t,\boldsymbol{\theta}} = \frac{\mathrm{d}s_o^t}{\mathrm{d}s_r^{t-1}}\delta_{o,r}\epsilon_{rpq}^{t-1,\boldsymbol{\theta}} + \left(\frac{\partial s_o^t}{\partial\theta_{pq}}\delta_{o,q} + \frac{\partial s_o^t}{\partial y_r^{t-1}}\delta_{o,r}\frac{\partial y_r^{t-1}}{\partial\theta_{pq}}\delta_{r,q}\right) \tag{68}$$

$$\epsilon_{op}^{t,\boldsymbol{\theta}} \sim \frac{\mathrm{d}s_o^t}{\mathrm{d}s_r^{t-1}}\delta_{o,r}\epsilon_{rp}^{t-1,\boldsymbol{\theta}} \sim \frac{\mathrm{d}s_o^t}{\mathrm{d}s_o^{t-1}}\epsilon_{op}^{t-1,\boldsymbol{\theta}}. \tag{69}$$

Thus, we observe that the number of multiplications is

$$\epsilon_{op}^{t,\boldsymbol{\theta}} \sim O(1) \quad\text{and}\quad \boldsymbol{\epsilon}^{t,\boldsymbol{\theta}} \sim O\left(n^2\right).$$

This implies that OSTL enables training of feed-forward SNNs with reduced time complexity of $O(n^2)$ for shallow and of $O(Kn^2)$ for K-layered networks. We empirically verified the complexity for shallow networks and show the results in Fig. 7. Note that this analysis holds also for single or multiple layers of generic recurrent neural networks (RNNs).

Appendix D
Recurrent Layers Embedded in Stateless Networks

In many network architectures, a recurrent layer is embedded along with an arbitrary number of nonrecurrent, stateless layers, for example, a softmax output layer. From the perspective of OSTL, a stateless layer does not introduce any residual term $\mathbf{R}$, and therefore, OSTL maintains gradient equivalence with BPTT even for deep architectures. To demonstrate this, we consider a deep network consisting of stateless layers, with the following state and output equations:

$$\boldsymbol{s}_l^t = \boldsymbol{\psi}\left(\boldsymbol{y}_{l-1}^t, \boldsymbol{\theta}_l\right) \tag{70}$$

$$\boldsymbol{y}_l^t = \boldsymbol{\chi}\left(\boldsymbol{s}_l^t, \boldsymbol{\theta}_l\right). \tag{71}$$

These equations simplify the term $\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l}$ from (53) to

$$\boldsymbol{\epsilon}_{m,l}^{t,\boldsymbol{\theta}_l} = \frac{\partial \boldsymbol{s}_m^t}{\partial \boldsymbol{y}_{m-1}^t}\left(\frac{\partial \boldsymbol{y}_{m-1}^t}{\partial \boldsymbol{s}_{m-1}^t}\boldsymbol{\epsilon}_{m-1,l}^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_{m-1}^t}{\partial \boldsymbol{\theta}_l}\right) + \frac{\partial \boldsymbol{s}_m^t}{\partial \boldsymbol{\theta}_l}. \tag{72}$$

If we insert this term into (57), we obtain

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l} = \sum_{1\leq t\leq T}\left(\frac{\mathrm{d}E^t}{\mathrm{d}\boldsymbol{y}_K^t}\left(\frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{y}_{K-1}^t}\left(\frac{\partial \boldsymbol{y}_{K-1}^t}{\partial \boldsymbol{s}_{K-1}^t}\boldsymbol{\epsilon}_{K-1,l}^{t,\boldsymbol{\theta}_l} + \frac{\partial \boldsymbol{y}_{K-1}^t}{\partial \boldsymbol{\theta}_l}\right) + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{s}_K^t}\frac{\partial \boldsymbol{s}_K^t}{\partial \boldsymbol{\theta}_l}\right) + \frac{\partial \boldsymbol{y}_K^t}{\partial \boldsymbol{\theta}_l}\right). \tag{73}$$

By recursively inserting (72) into (73) until $K-r=l$, most terms include partial derivatives of the form $((\partial \boldsymbol{y}_{K-r}^t)/(\partial \boldsymbol{\theta}_l))$ or $((\partial \boldsymbol{s}_{K-r}^t)/(\partial \boldsymbol{\theta}_l))$. These terms vanish because the layer

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

$K-r$ does not depend on parameters $\boldsymbol{\theta}_l$, except when $K-r=l$. Therefore, (73) can be written as

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l}=\sum_{1\le t\le T}\left(\frac{\mathrm{d}E^t}{\mathrm{d}\boldsymbol{y}_K^t}\left(\prod_{(K-l+1)>m\ge 1}\frac{\partial \boldsymbol{y}^t_{K-m+1}}{\partial \boldsymbol{s}^t_{K-m+1}}\frac{\partial \boldsymbol{s}^t_{K-m+1}}{\partial \boldsymbol{y}^t_{K-m}}\right)\times\left(\frac{\partial \boldsymbol{y}^t_l}{\partial \boldsymbol{s}^t_l}\boldsymbol{\epsilon}_l^{t,\boldsymbol{\theta}_l}+\frac{\partial \boldsymbol{y}^t_l}{\partial \boldsymbol{\theta}_l}\right)\right) \tag{74}$$

which is the combination of the generalized learning signal L_l^t and eligibility trace $e_l^{t,\boldsymbol{\theta}_l}$ without any residual term $\mathbf{R}$, as stated in (62)

$$\frac{\mathrm{d}E}{\mathrm{d}\boldsymbol{\theta}_l}=\sum_{1\le t\le T}\mathbf{L}_l^t\mathbf{e}_l^{t,\boldsymbol{\theta}_l}. \tag{75}$$

Appendix E
OSTL for Long Short-Term Memory

Because long short-term memory (LSTM) units are probably the most commonly used type of RNN units in contemporary state-of-the-art machine learning applications, we include an explicit derivation of OSTL for them. An LSTM unit contains three gates and an internal state. A single layer l, composed of LSTM units, is governed by the following:

$$\mathbf{i}_l^t=\mathbf{g}\left(\mathbf{W}_l^i\boldsymbol{y}_{l-1}^t+\mathbf{H}_l^i\boldsymbol{y}_l^{t-1}+\mathbf{b}_l^i\right) \tag{76}$$
$$\mathbf{c}_l^t=\mathbf{g}\left(\mathbf{W}_l^c\boldsymbol{y}_{l-1}^t+\mathbf{H}_l^c\boldsymbol{y}_l^{t-1}+\mathbf{b}_l^c\right) \tag{77}$$
$$\mathbf{f}_l^t=\mathbf{g}\left(\mathbf{W}_l^f\boldsymbol{y}_{l-1}^t+\mathbf{H}_l^f\boldsymbol{y}_l^{t-1}+\mathbf{b}_l^f\right) \tag{78}$$
$$\boldsymbol{s}_l^t=\mathbf{f}_l^t\odot\boldsymbol{s}_l^{t-1}+\mathbf{i}_l^t\odot\mathbf{h}\left(\mathbf{W}_l^s\boldsymbol{y}_{l-1}^t+\mathbf{H}_l^s\boldsymbol{y}_l^{t-1}+\mathbf{b}_l^s\right) \tag{79}$$
$$\boldsymbol{y}_l^t=\mathbf{c}_l^t\odot\mathbf{h}\left(\boldsymbol{s}_l^t\right) \tag{80}$$

where typically $\mathbf{g}=\boldsymbol{\sigma}$ and $\mathbf{h}=\mathbf{tanh}$. Note that we reformulated (79) and (80) such that one can immediately see the resemblance to the generalized form of (48) and (49) of the main text. By using (8), the eligibility traces for a layer of LSTM units take the following form:

$$\begin{aligned}
\mathbf{e}_l^{t,\mathbf{W}_l^i}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{W}_l^i}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{W}_l^i}\\
\mathbf{e}_l^{t,\mathbf{H}_l^i}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{H}_l^i}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{H}_l^i}\\
\mathbf{e}_l^{t,\mathbf{b}_l^i}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{b}_l^i}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{b}_l^i}\\
\mathbf{e}_l^{t,\mathbf{W}_l^f}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{W}_l^f}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{W}_l^f}\\
\mathbf{e}_l^{t,\mathbf{H}_l^f}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{H}_l^f}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{H}_l^f}\\
\mathbf{e}_l^{t,\mathbf{b}_l^f}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{b}_l^f}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{b}_l^f}\\
\mathbf{e}_l^{t,\mathbf{W}_l^s}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{W}_l^s}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{W}_l^s}\\
\mathbf{e}_l^{t,\mathbf{H}_l^s}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{H}_l^s}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{H}_l^s}\\
\mathbf{e}_l^{t,\mathbf{b}_l^s}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{b}_l^s}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{b}_l^s}\\
\mathbf{e}_l^{t,\mathbf{W}_l^c}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{W}_l^c}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t\\
&\quad+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{W}_l^c}\\
\mathbf{e}_l^{t,\mathbf{H}_l^c}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{H}_l^c}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1}\\
&\quad+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{H}_l^c}\\
\mathbf{e}_l^{t,\mathbf{b}_l^c}&=\mathrm{diag}\left(\mathbf{c}_l^t\odot\mathbf{h}'^{y,t}_l\right)\boldsymbol{\epsilon}^{t,\mathbf{b}_l^c}+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\\
&\quad+\mathrm{diag}\left(\mathbf{h}_l^{y,t}\odot\mathbf{g}'^{c,t}_l\right)\mathbf{H}_l^c\mathbf{e}_l^{t-1,\mathbf{b}_l^c}
\end{aligned}$$

using (43) the eligibility tensors in the above-mentioned equations become

$$\begin{aligned}
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^i}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^i}+\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t\\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^i}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^i}+\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1}\\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^i}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^i}+\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\\
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^f}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^f}+\mathrm{diag}\left(\boldsymbol{s}_l^{t-1}\odot\mathbf{g}'^{f,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t\\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^f}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^f}+\mathrm{diag}\left(\boldsymbol{s}_l^{t-1}\odot\mathbf{g}'^{f,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1}\\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^f}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^f}+\mathrm{diag}\left(\boldsymbol{s}_l^{t-1}\odot\mathbf{g}'^{f,t}_l\right)\\
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^s}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^s}+\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t\\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^s}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^s}+\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1}\\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^s}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^s}+\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\\
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^c}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^c}\\
&\quad+\left(\mathrm{diag}\left(\mathbf{g}'^{f,t}_l\odot\boldsymbol{s}_l^{t-1}\right)\mathbf{H}_l^f\mathrm{diag}\left(\mathbf{g}'^{c,t-1}_l\odot\mathbf{h}_l^{y,t-1}\right)\boldsymbol{\Upsilon}_{l-1}^{t-1}\right)\\
&\quad+\left(\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\mathbf{H}_l^i\mathrm{diag}\left(\mathbf{g}'^{c,t-1}_l\odot\mathbf{h}_l^{y,t-1}\right)\boldsymbol{\Upsilon}_{l-1}^{t-1}\right)\\
&\quad+\left(\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\mathbf{H}_l^s\mathrm{diag}\left(\mathbf{g}'^{c,t-1}_l\odot\mathbf{h}_l^{y,t-1}\right)\boldsymbol{\Upsilon}_{l-1}^{t-1}\right)\\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^c}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^c}+\mathrm{diag}\left(\mathbf{g}'^{f,t}_l\odot\boldsymbol{s}_l^{t-1}\right)\mathbf{H}_l^f\\
&\quad\mathrm{diag}\left(\mathbf{g}_l'^{c,t-1}\odot\mathbf{h}_l^{y,t-1}\right)\boldsymbol{\Upsilon}_l^{t-2}+\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\mathbf{H}_l^i\\
&\quad\mathrm{diag}\left(\mathbf{g}_l'^{c,t-1}\odot\mathbf{h}_l^{y,t-1}\right)\boldsymbol{\Upsilon}_l^{t-2}+\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\mathbf{H}_l^s\\
&\quad\mathrm{diag}\left(\mathbf{g}_l'^{c,t-1}\odot\mathbf{h}_l^{y,t-1}\right)\boldsymbol{\Upsilon}_l^{t-2}\\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^c}&=\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^c}\\
&\quad+\left(\mathrm{diag}\left(\mathbf{g}'^{f,t}_l\odot\boldsymbol{s}_l^{t-1}\right)\mathbf{H}_l^f\mathrm{diag}\left(\mathbf{g}'^{c,t-1}_l\odot\mathbf{h}_l^{y,t-1}\right)\right.\\
&\qquad+\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\mathbf{H}_l^i\mathrm{diag}\left(\mathbf{g}'^{c,t-1}_l\odot\mathbf{h}_l^{y,t-1}\right)\\
&\qquad\left.+\,\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\mathbf{H}_l^s\mathrm{diag}\left(\mathbf{g}'^{c,t-1}_l\odot\mathbf{h}_l^{y,t-1}\right)\right)
\end{aligned}$$

with

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}&=\mathrm{diag}\left(\mathbf{f}^t\right)+\mathrm{diag}\left(\boldsymbol{s}_l^{t-1}\odot\mathbf{g}'^{f,t}_l\right)\mathbf{H}_l^f\mathrm{diag}\left(\mathbf{c}_l^{t-1}\odot\mathbf{h}'^{y,t-1}_l\right)\\
&\quad+\mathrm{diag}\left(\mathbf{h}_l^{s,t}\odot\mathbf{g}'^{i,t}_l\right)\mathbf{H}_l^i\mathrm{diag}\left(\mathbf{c}_l^{t-1}\odot\mathbf{h}'^{y,t-1}_l\right)\\
&\quad+\mathrm{diag}\left(\mathbf{i}_l^t\odot\mathbf{h}'^{s,t}_l\right)\mathbf{H}_l^s\mathrm{diag}\left(\mathbf{c}_l^{t-1}\odot\mathbf{h}'^{y,t-1}_l\right).
\end{aligned}$$

We have used the short-hand notation of $((\mathrm{d}\boldsymbol{\psi}_l^t)/(\mathrm{d}\boldsymbol{\omega}^t))=\mathbf{g}'|_{\boldsymbol{\psi}_l^t}=\mathbf{g}'^{\boldsymbol{\psi},t}_l$ with $\boldsymbol{\psi}=\{\mathbf{i},\mathbf{c},\mathbf{f},\mathbf{s}\}$ and $\boldsymbol{\omega}=\{\boldsymbol{y}_l^{t-1},\boldsymbol{\theta}_l\}$, $h(\boldsymbol{s}_l^t)=\mathbf{h}_l^{y,t}$, $((dh(\boldsymbol{s}_l^t))/(\mathrm{d}\boldsymbol{s}_l^t))=\mathbf{h}'^{y,t}_l$, $\mathbf{h}(\mathbf{W}_l^s\boldsymbol{y}_{l-1}^t+\mathbf{H}_l^s\boldsymbol{y}_l^{t-1}+\mathbf{b}_l^s)=\mathbf{h}_l^{s,t}$,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

and $((d\mathbf{h}(\mathbf{W}_l^s \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^s \boldsymbol{y}_l^{t-1} + \mathbf{b}_l^s))/(\mathrm{d}\boldsymbol{\omega})) = \mathbf{h}'|_{\mathbf{W}_l^s \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^s \boldsymbol{y}_l^{t-1} + \mathbf{b}_l^s} = \mathbf{h}'^{y,t}_l$. Furthermore, the elements of $\boldsymbol{\Upsilon}_l^t$ are given by $(\Upsilon_l^t)_{\mathrm{opq}} = \delta_{o,p}(y_l^t)_q$.

Considering the mean squared error loss with readout weights given in (18) of the main text

$$E^t = \frac{1}{2}\sum_{i=1}^{n_K}\left(\left(\mathbf{W}^{\mathrm{out}}\boldsymbol{y}_K^t\right)_i - \left(\hat{\boldsymbol{y}}^t\right)_i\right)^2 \tag{81}$$

the learning signals become

$$\begin{aligned}
&\mathbf{L}_l^t \\
&= \mathbf{L}_{l+1}^t\Big(\mathrm{diag}\big(\mathbf{c}_{l+1}^t \odot \mathbf{h}'^{y,t}_{l+1}\big) \\
&\qquad \Big(\mathrm{diag}\Big(\boldsymbol{s}_{l+1}^{t-1} \odot \mathbf{g}'^{f,t}_{l+1}\Big)\mathbf{W}_{l+1}^f + \mathrm{diag}\Big(\mathbf{h}_{l+1}^{s,t} \odot \mathbf{g}'^{i,t}_{l+1}\Big)\mathbf{W}_{l+1}^i \\
&\qquad + \mathrm{diag}\Big(\mathbf{g}_{l+1}^{i,t} \odot \mathbf{h}'^{s,t}_{l+1}\Big)\mathbf{W}_{l+1}^s\Big) + \mathrm{diag}\big(\mathbf{h}_{l+1}^{s,t} \odot \mathbf{g}'^{c,t}_{l+1}\big)\mathbf{H}_{l+1}^c\Big)
\end{aligned}$$

with

$$\mathbf{L}_K^t = \left(\mathbf{W}^{\mathrm{out}}\right)^\top \boldsymbol{y}_K^t - \hat{\boldsymbol{y}}^t. \tag{82}$$

Appendix F
OSTL for Gated Recurrent Unit

OSTL can also be applied to GRUs. To illustrate this, we start from the state equations

$$\mathbf{u}_l^t = \mathbf{g}\left(\mathbf{W}_l^u \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^u \boldsymbol{s}_l^{t-1} + \mathbf{b}_l^u\right) \tag{83}$$

$$\mathbf{v}_l^t = \mathbf{g}\left(\mathbf{W}_l^v \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^v \boldsymbol{s}_l^{t-1} + \mathbf{b}_l^v\right) \tag{84}$$

$$\begin{aligned}\boldsymbol{s}_l^t &= \mathbf{u}_l^t \odot \boldsymbol{s}_l^{t-1} \\ &\quad + \left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}\left(\mathbf{W}_l^c \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^c\left(\mathbf{v}_l^t \odot \boldsymbol{s}_l^{t-1}\right) + \mathbf{b}_l^c\right)\end{aligned} \tag{85}$$

where typically $\mathbf{g} = \boldsymbol{\sigma}$ and $\mathbf{h} = \mathbf{tanh}$. Note that we reformulated (85) such that one can immediately see the resemblance to the generalized form of (48) and (49) of the main text.

As one can see, a gated recurrent unit (GRU) does not have a separate output equation. Therefore, in order to remain consistent with the previous notation and to apply OSTL, we introduce a simple output equation

$$\boldsymbol{y}_l^t = \boldsymbol{s}_l^t \tag{86}$$

which does not change the behavior of the GRU. The eligibility traces can then be calculated according to (8) as

$$\begin{aligned}
\mathbf{e}_l^{t,\mathbf{W}_l^u} &= \boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^u}, \quad \mathbf{e}_l^{t,\mathbf{H}_l^u} = \boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^u}, \quad \mathbf{e}_l^{t,\mathbf{b}_l^u} = \boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^u} \\
\mathbf{e}_l^{t,\mathbf{W}_l^v} &= \boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^v}, \quad \mathbf{e}_l^{t,\mathbf{H}_l^v} = \boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^v}, \quad \mathbf{e}_l^{t,\mathbf{b}_l^v} = \boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^v} \\
\mathbf{e}_l^{t,\mathbf{W}_l^c} &= \boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^c}, \quad \mathbf{e}_l^{t,\mathbf{H}_l^c} = \boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^c}, \quad \mathbf{e}_l^{t,\mathbf{b}_l^c} = \boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^c}
\end{aligned}$$

using (43) the eligibility tensors in the equations earlier become

$$\begin{aligned}
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^u} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^u} + \mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{u,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t \\
&\quad - \mathrm{diag}\left(\mathbf{h}_l^{s,t} \odot \mathbf{g}'^{u,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t \\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^u} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^u} + \mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{u,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1} \\
&\quad - \mathrm{diag}\left(\mathbf{h}_l^{s,t} \odot \mathbf{g}'^{u,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1} \\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^u} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^u} \\
&\quad + \mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{u,t}_l\right) - \mathrm{diag}\left(\mathbf{h}_l^{s,t} \odot \mathbf{g}'^{u,t}_l\right) \\
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^v} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^v} + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right) \\
&\quad \times \mathbf{H}_l^c\,\mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{v,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t \\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^v} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^v} + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right) \\
&\quad \times \mathbf{H}_l^c\,\mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{v,t}_l\right)\boldsymbol{\Upsilon}_l^{t-1} \\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^v} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^v} \\
&\quad + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right)\mathbf{H}_l^c\,\mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{v,t}_l\right) \\
\boldsymbol{\epsilon}_l^{t,\mathbf{W}_l^c} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{W}_l^c} + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right)\boldsymbol{\Upsilon}_{l-1}^t \\
\boldsymbol{\epsilon}_l^{t,\mathbf{H}_l^c} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{H}_l^c} + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right)\boldsymbol{P}_l^{t-1} \\
\boldsymbol{\epsilon}_l^{t,\mathbf{b}_l^c} &= \frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}}\boldsymbol{\epsilon}_l^{t-1,\mathbf{b}_l^c} + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right)
\end{aligned} \tag{87}$$

with

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{s}_l^t}{\mathrm{d}\boldsymbol{s}_l^{t-1}} &= \mathrm{diag}\left(\mathbf{u}_l^t\right) + \mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{u,t}_l\right)\mathbf{H}_l^u \\
&\quad - \mathrm{diag}\left(\mathbf{h}_l^{s,t} \odot g'^{u,t}_l\right)\mathbf{H}_l^u + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right)\mathbf{H}_l^c \\
&\quad \times \mathrm{diag}\left(\mathbf{v}_l^t\right) + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right)\mathrm{diag}\left(\boldsymbol{s}_l^{t-1}\right)\mathbf{H}_l^v \\
&\quad \times \mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{v,t}_l\right)\mathbf{H}_l^c.
\end{aligned}$$

We used the short-hand notation of $((\mathrm{d}\boldsymbol{\psi}_l^t)/(\mathrm{d}\boldsymbol{\omega}^t)) = \mathbf{g}'|_{\boldsymbol{\psi}_l^t} = \mathbf{g}'^{\boldsymbol{\psi},t}_l$ with $\boldsymbol{\psi} = \{\mathbf{i}, \mathbf{v}, \mathbf{f}, \mathbf{s}\}$ and $\boldsymbol{\omega} = \{\boldsymbol{y}_l^{t-1}, \boldsymbol{\theta}_l\}$, $\mathbf{h}(\mathbf{W}_l^c \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^c(\mathbf{g}(\mathbf{W}_l^v \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^v \boldsymbol{s}_l^{t-1} + \mathbf{b}_l^v) \odot \boldsymbol{s}_l^{t-1}) + \mathbf{b}_l^c) = \mathbf{h}_l^{s,t}$ and $((d\mathbf{h}_l^{s,t})/(\mathrm{d}\boldsymbol{\omega})) = \mathbf{h}'|_{\mathbf{W}_l^c \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^c(\mathbf{g}(\mathbf{W}_l^v \boldsymbol{y}_{l-1}^t + \mathbf{H}_l^v \boldsymbol{s}_l^{t-1} + \mathbf{b}_l^v) \odot \boldsymbol{s}_l^{t-1}) + \mathbf{b}_l^c} = \mathbf{h}'^{s,t}_l$. Furthermore, the elements of $\boldsymbol{\Upsilon}_l^t$ are given by $(\Upsilon_l^t)_{\mathrm{opq}} = \delta_{o,p}(y_l^t)_q$ and the elements of $\boldsymbol{P}_l^t$ are given by $(P_l^t)_{\mathrm{opq}} = \delta_{o,p}(r_l^{t+1} s_l^t)_q$.

Considering again the mean squared error loss from (18), the learning signals become

$$\begin{aligned}
\mathbf{L}_l^t &= \mathbf{L}_{l+1}^t\big(\mathrm{diag}\left(\boldsymbol{s}_l^{t-1} \odot \mathbf{g}'^{u,t}_l\right)\mathbf{W}_{l+1}^u \\
&\quad - \mathrm{diag}\left(\mathbf{h}_l^{s,t} \odot \mathbf{g}'^{u,t}_l\right)\mathbf{W}_{l+1}^u + \mathrm{diag}\left(\left(\mathbb{1} - \mathbf{u}_l^t\right) \odot \mathbf{h}'^{s,t}_l\right) \\
&\quad \times \left(\mathbf{W}_{l+1}^c + \mathbf{H}_{l+1}^c\,\mathrm{diag}\left(\mathbf{g}'^{v,t}_l\right)\mathbf{W}_{l+1}^v\right)\big)
\end{aligned} \tag{88}$$

with

$$\mathbf{L}_K^t = \left(\mathbf{W}^{\mathrm{out}}\right)^\top \boldsymbol{y}_K^t - \hat{\boldsymbol{y}}^t. \tag{89}$$

Acknowledgment

The authors would like to thank Robert Legenstein, Friedemann Zenke, Emre Neftci, and especially Wolfgang Maass for helpful comments and fruitful discussions on earlier versions of this article.

References

[1] B. Kolb and I. Whishaw, "Brain plasticity and behavior," *Annu. Rev. Psychol.*, vol. 49, pp. 43–64, Feb. 1998.

[2] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[3] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[4] Y. He *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6381–6385.

[5] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. INTERSPEECH*, Sep. 2012, pp. 194–197.

[6] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Proc. NIPS*, 2018, pp. 1412–1421.

[7] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. NIPS*, 2018, pp. 787–797.

[8] S. Woźniak, A. Pantazi, T. Bohnstingl, and E. Eleftheriou, "Deep learning incorporating biologically-inspired neural dynamics," 2018, *arXiv:1812.07040*.

[9] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[10] S. Woźniak, A. Pantazi, T. Bohnstingl, and E. Eleftheriou, "Deep learning incorporating biologically inspired neural dynamics and in-memory computing," *Nature Mach. Intell.*, vol. 2, no. 6, pp. 325–336, Jun. 2020.

[11] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Berlin, Germany: Springer-Verlag, 2006.

[13] G. Bellec *et al.*, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Commun.*, vol. 11, no. 1, pp. 1–15, Jul. 2020.

[14] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, Jun. 1989.

[15] C. Tallec and Y. Ollivier, "Unbiased online recurrent optimization," in *Proc. ICLR*, 2018.

[16] A. Mujika, F. Meier, and A. Steger, "Approximating real-time recurrent learning with random Kronecker factors," in *Proc. NIPS*, 2018, pp. 6594–6603.

[17] F. Benzing, M. M. Gauy, A. Mujika, A. Martinsson, and A. Steger, "Optimal Kronecker-sum approximation of real time recurrent learning," in *Proc. ICML*, 2019, pp. 604–613.

[18] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain," *Current Opinion Neurobiol.*, vol. 55, pp. 82–89, Apr. 2019.

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

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

[21] J. M. Murray, "Local online learning in recurrent networks with random feedback," *eLife*, vol. 8, p. e43299, May 2019.

[22] O. Marschall, K. Cho, and C. Savin, "A unified framework of online learning algorithms for training recurrent neural networks," Jul. 2019, *arXiv:1907.02649*.

[23] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea, "Eligibility traces and plasticity on behavioral time scales: Experimental support of NeoHebbian three-factor learning rules," *Frontiers Neural Circuits*, vol. 12, p. 53, Jul. 2018.

[24] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *Proc. NIPS*, 2016, pp. 5067–5075.

[25] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," 2019, *arXiv:1909.12415*.

[26] O. Sporns, D. R. Chialvo, M. Kaiser, and C. C. Hilgetag, "Organization, development and function of complex brain networks," *Trends Cogn. Sci.*, vol. 8, no. 9, pp. 418–425, Sep. 2004.

[27] O. Sporns, G. Tononi, and R. Kötter, "The human connectome: A structural description of the human brain," *PLoS Comput. Biol.*, vol. 1, no. 4, p. e42, 2005.

[28] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," in *Proc. ICLR*, 2017.

[29] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Commun.*, vol. 7, no. 1, pp. 1–10, Nov. 2016.

[30] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Proc. NIPS*, 2016, pp. 1037–1045.

[31] T. Bohnstingl, A. Garg, S. Woźniak, G. Saon, E. Eleftheriou, and A. Pantazi, "Towards efficient end-to-end speech recognition with biologically-inspired neural networks," Oct. 2021, *arXiv:2110.02743*.

[32] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. ICML*, 2012, pp. 1881–1888.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[34] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: The penn treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, Jun. 1993.

[35] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NTIS Order PB91-505065, 1993.

[36] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2016.

[37] R. J. Williams and D. Zipser, *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Jan. 1995.

Thomas Bohnstingl (Member, IEEE) received the M.Sc. degree in computer science and the M.Sc. degree in technical physics from Graz University of Technology, Graz, Austria. He is currently pursuing the Ph.D. degree with the Neuromorphic Computing and I/O Links Group, IBM Research Zurich, Rüschlikon, Switzerland, under the supervision of Prof. Wolfgang Maass.

His research interests include enhancing the capabilities of neural networks with biological inspiration, learning theories in machine learning, and neuromorphic computing.

Stanisław Woźniak (Member, IEEE) received the Ph.D. degree in neuromorphic computing from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2017.

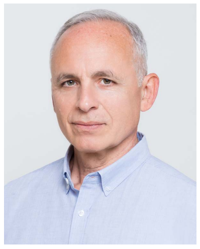
Since 2018, he has been a Researcher with the Neuromorphic Computing and I/O Links Group, IBM Research Zurich, Rüschlikon, Switzerland. His research interest includes closing the gap between the theoretically appealing properties of SNNs and their applications.

Dr. Woźniak received the Fritz Kutter Award of ETH Zürich in 2018 and the IBM Research Division Award for contributions in neuromorphic computing.

Angeliki Pantazi (Senior Member, IEEE) received the Diploma and Ph.D. degrees in electrical engineering and computer technology from the University of Patras, Patras, Greece, in 1996 and 2005, respectively.

She is currently a Principal Research Staff Member with IBM Research Zurich, Rüschlikon, Switzerland, where she manages the research on neuromorphic computing. She was named IBM Master Inventor in 2014. She has contributed to several control-related projects in data storage systems and in particular, magnetic tape storage.

Dr. Pantazi was named a fellow of the International Federation of Automatic Control (IFAC) in 2019.



Evangelos Eleftheriou (Life Fellow, IEEE) received the Ph.D. degree in electrical engineering from Carleton University, Ottawa, ON, Canada, in 1985.

In 1986, he joined IBM Research Zurich, Rüschlikon, Switzerland, as a Research Staff Member, where he has held various management positions over the years. He is currently the CTO of Axelera AI, Zürich, Switzerland.

Dr. Eleftheriou was appointed an IBM Fellow in 2005. In 2009, he received the IEEE Control Systems Technology Award. In 2018, he was inducted into the U.S. National Academy of Engineering.