EVENTPROP: BACKPROPAGATION FOR EXACT GRADIENTS IN SPIKING NEURAL NETWORKS

Timo C. Wunderlich*
Kirchhoff-Institute for Physics
Heidelberg University
69120 Heidelberg, Germany
Current Address:
Berlin Institute of Health
Charité–Universitätsmedizin
10117 Berlin, Germany
timo.wunderlich@charite.de

Christian Pehle*
Kirchhoff-Institute for Physics
Heidelberg University
69120 Heidelberg, Germany
christian.pehle@kip.uni-heidelberg.de

September 18, 2020

ABSTRACT

We derive the backpropagation algorithm for spiking neural networks composed of leaky integrate-and-fire neurons operating in continuous time. This algorithm, EventProp, computes the exact gradient of an arbitrary loss function of spike times and membrane potentials by backpropagating errors in time. For the first time, by leveraging methods from optimal control theory, we are able to backpropagate errors through spike discontinuities and avoid approximations or smoothing operations. EventProp can be applied to spiking networks with arbitrary connectivity, including recurrent, convolutional and deep feed-forward architectures. While we consider the leaky integrate-and-fire neuron model in this work, our methodology to derive the gradient can be applied to other spiking neuron models. As errors are backpropagated in an event-based manner (at spike times), EventProp requires the storage of state variables only at these times, providing favorable memory requirements. We demonstrate learning using gradients computed via EventProp in a deep spiking network using an event-based simulator and a non-linearly separable dataset encoded using spike time latencies. Our work supports the rigorous study of gradient-based methods to train spiking neural networks while providing insights toward the development of learning algorithms in neuromorphic hardware.

*The authors have contributed equally.

1 Introduction

How can we train spiking neural networks to achieve brain-like performance in machine learning tasks? The resounding success and pervasive use of the gradient-based backpropagation algorithm in deep learning suggests an analogous approach.

Spiking neural networks hold the promise for efficient and robust processing of event-based spatio-temporal data as found in biological systems. Although the brain is able to learn impressively well using spike-based communication, spiking models have not seen widespread success in machine learning applications. At the same time, learning in spiking neural networks is an active research subject, with a wide variety of continuously proposed algorithms, and the development of spiking neuromorphic hardware receives increasing attention (Roy et al., 2019). A notorious issue in spiking neurons is the hard, non-differentiable spiking threshold that does not permit a straight-forward application of differential calculus to compute gradients. Although exact gradients have been derived for special cases, this issue is commonly side-stepped by using smoothed or stochastic neuron models or by replacing the hard threshold function using a surrogate function, leading to the computation of surrogate gradients (Neftci et al., 2019).

In contrast, this work provides the exact gradient for an arbitrary loss function defined using the state variables (spike times, membrane potentials) of a general recurrent spiking neural network. Since feed-forward architectures correspond to recurrent neural networks with block-diagonal weight matrices and convolutions can be represented as sparse linear transformations, deep feed-forward networks and convolutional networks are included as special cases. A basic observation is that both the spike times and membrane potential between spikes of a neuron embedded in such a network are differentiable almost everywhere with respect to any synaptic weight, up to the null set that contains the hyperplanes defining the borders in weight space where spikes appear or disappear (the *critical parameters*). Therefore, smooth loss functions defined using these state variables are differentiable as well, up to these critical parameters.

The relevant question is now how to compute these gradients, preferably with the computational efficiency afforded by the backpropagation algorithm and retaining any potential advantages of spike-based coding.

In order to solve this problem, it is helpful to appreciate that the backpropagation algorithm as used in artificial neural networks can be derived using the adjoint method (LeCun et al., 1988). Indeed, there is a direct correspondence between the intermediate variables computed in the backpropagation algorithm and the adjoint variables λ used in the adjoint method. The adjoint variables (Lagrange multipliers) of dynamical systems (Pontryagin, 1962) that operate in continuous time are also functions of time, $\lambda(t)$, and their computation corresponds to the backpropagation of errors in time. The gradient of a given loss function is then a function of the $\lambda(t)$. See Bradley (2019) for a concise derivation in the context of PDE-constrained optimization.

It remains to treat the discontinuities arising in spiking neural networks. The computation of partial derivatives (*parameter sensitivities*) for hybrid systems exhibiting both continuous and discrete dynamics (e.g., a set of ODEs combined with state variable jumps) is an established topic in optimal control theory (Barton and Lee, 2002; Rozenwasser and Yusupov, 2019). For these hybrid systems, the sensitivity $\frac{\partial x}{\partial p}(t)$ of a state variable x with respect to a parameter p generally experiences jumps at the points of discontinuity. The relation between the sensitivities before and after a given discontinuity was first studied in the 1960s (De Backer, 1964; Rozenvasser, 1967). A more general theoretical framework was developed thirty years later by Galán et al. (1999) who provide existence and uniqueness theorems for sensitivity trajectories of hybrid systems. These theorems rely on Gronwall's theorem (Gronwall, 1919) for the existence of sensitivity trajectories between transitions and on the implicit function theorem to relate sensitivities before and after a given transition. The adjoint method can be combined with these insights to derive the adjoint dynamics for hybrid systems (Serban and Recuero, 2019). Our work uses these methods to derive the gradient for a spiking neural network.

Our results show that the gradient can be computed using an adjoint spiking neural network which backpropagates errors at the spike times. This instantly suggests a simple algorithm, *EventProp*, to compute the gradient. Since the adjoint spiking network backpropagates error at the spike times, the algorithm retains the potential advantages of spike-based coding in terms of efficiency and robustness and is amenable to neuromorphic implementation.

1.1 Previous Work

We provide a compact overview of gradient-based approaches to learning in spiking neural networks and refer the reader to the following review articles for a comprehensive survey of this topic: Pfeiffer and Pfeil (2018) and Tavanaei et al. (2019) discuss learning in deep spiking networks, Roy et al. (2019) discusses learning along with the history and future of neuromorphic computing and Neftci et al. (2019) focuses on the surrogate gradient approach.

Surrogate gradients replace non-differentiable spiking threshold functions with smooth functions for the purposes of backpropagation and have been used to train spiking networks in a variety of settings (e.g., Esser et al., 2016; Bellec et al., 2018; Zenke and Ganguli, 2018; Shrestha and Orchard, 2018). Apart from surrogate gradients, several publications provide exact gradients for first-spike-time based loss functions and LIF neurons: Bohte et al. (2000) provides the gradient for at most one spike per layer and this result was subsequently generalized to an arbitrary number of spikes as well as recurrent connectivity (Booij and tat Nguyen, 2005; Xu et al., 2013). While these publications provide recursive relations for the gradient that can be implicitly computed using backpropagation, we explicitly provide the dynamical system that implements backpropagation and show that it represents an adjoint spiking network which transmits errors at spike times. In addition, we also consider voltage-dependent loss functions and our methodology can be applied to neuron models without analytic expressions for the Post-Synaptic Potential (PSP) kernels. The chronotron (Florian, 2012) uses a gradient-based learning rule based on the Victor-Purpura metric which enables single LIF neurons to learn a target spike train. Our work, as well as the works mentioned above which derive exact gradients, applies the implicit function theorem to the relation $v(t, w) - \vartheta = 0$ to differentiate spike times with respect to synaptic weights. A different approach is to consider ratios of the neuronal time constants τ_{mem}, τ_{syn} where analytic expressions for first spike times can be given and to derive the corresponding gradients, as done in Göltz et al. (2019); Comsa et al. (2020); Mostafa (2017). Our work encompasses these results as special cases.

The seminal Tempotron model uses gradient descent to adjust the sub-threshold voltage maximum in a single neuron (Gütig and Sompolinsky, 2006) and has recently been generalized to the spike threshold surface formalism Gütig (2016) that uses the exact gradient of the critical thresholds ϑ_k^* at which an LIF neuron transitions from emitting k to $k-1$ spikes; computing this gradient is not considered in this work. The adjoint method was recently used to optimize neural ordinary differential equations (Chen et al., 2018) and to derive the gradient for a smoothed spiking neuron model without reset (Huh and Sejnowski, 2018).

2 Gradient of a Recurrent Spiking Neural Network

2.1 Leaky Integrate-and-Fire Neural Network Model

We define a network of N standard LIF neurons with arbitrary (up to self-connections) recurrent connectivity (table 1). We set the leak potential to zero and choose parameter-independent initial conditions. Note that the Spike-Response Model (SRM) (Gerstner and Kistler, 2002) with double-exponential or α-shaped PSPs is generally an integral expression of the model given in table 1 with corresponding time constants.

Free Dynamics	**Transition Condition**	**Jumps at Transition**
$\tau_{\text{mem}} \frac{\mathrm{d}}{\mathrm{d}t} V = -V + I$	$(V)_n - \vartheta = 0$	$(V^+)_n = (V^-)_n - \vartheta$
	$(\dot{V})_n \neq 0$	$I^+ = I^- + W e_n$
$\tau_{\text{syn}} \frac{\mathrm{d}}{\mathrm{d}t} I = -I$	for any n	

Table 1: The LIF spiking neural network model. Inbetween spikes, the vector of membrane potentials V and synaptic currents I evolve according to the free dynamics. When some neuron n crosses the threshold ϑ, the transition condition is fulfilled, causing a spike. This leads to a reset of the membrane potential as well as post-synaptic current jumps. $W \in \mathbb{R}^{N \times N}$ is the weight matrix with zero diagonal and $e_n \in \mathbb{R}^N$ is the unit vector with a 1 at index n and 0 at all other indices. We use $+$ and $-$ to denote quantities before and after a given spike.

2.2 Main Result

Consider smooth loss functions $l_V(V, t)$, $l_{\text{p}}(t^{\text{post}})$ that depend on the membrane potentials V, time t and the set of post-synaptic spike times t^{post}. The total loss is given by

$$\mathcal{L} = l_{\text{p}}(t^{\text{post}}) + \int_0^T l_V(V(t), t) \mathrm{d}t. \tag{1}$$

As will be derived in the following section, the gradient of the total loss with respect to a specific weight $w_{ij} = (W)_{ij}$ that connects pre-synaptic neuron i to post-synaptic neuron j is given by a sum over the spikes caused by i,

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = -\tau_{\text{syn}} \sum_{\text{spikes from } i} (\lambda_I)_j \tag{2}$$

where λ_I is the adjoint variable (Lagrange multiplier) corresponding to I. Equation (2) therefore samples the post-synaptic neuron's adjoint variable $(\lambda_I)_j$ at the spike times caused by neuron i.

After the neuron dynamics given by table 1 have been computed from $t = 0$ to $t = T$, λ_I is computed in reverse time (i.e., from $t = T$ to $t = 0$) as the solution of the system defined in table 2. The dynamical system defined by table 2 is the adjoint spiking network to table 1 which backpropagates error signals at the spike times t^{post}.

Free Dynamics	**Transition Condition**	**Jump at Transition**
$\tau_{\text{mem}}\lambda_V' = -\lambda_V - \frac{\partial l_V}{\partial V}$ $\tau_{\text{syn}}\lambda_I' = -\lambda_I + \lambda_V$	$t - t_k^{\text{post}} = 0$ for any k	$(\lambda_V^-)_{n(k)} = (\lambda_V^+)_{n(k)} + \frac{1}{\tau_{\text{mem}}(\dot{V}^-)_{n(k)}}\Big[\vartheta(\lambda_V^+)_{n(k)}$ $+ W^T(\lambda_V^+ - \lambda_I) + \frac{\partial l_{\text{p}}}{\partial t_k^{\text{post}}} + l_V^- - l_V^+\Big]$

Table 2: The adjoint spiking network to table 1 that computes the adjoint variable λ_I needed for the gradient (eq. (2)). The adjoint variables are computed in reverse time (i.e., from $t = T$ to $t = 0$) with $' = -\frac{\mathrm{d}}{\mathrm{d}t}$ denoting the reverse time derivative. $(\lambda_V^-)_{n(k)}$ experiences jumps at the spikes times t_k^{post}, where $n(k)$ is the index of the neuron that caused the kth spike. Computing this system amounts to the backpropagation of errors in time. The initial conditions are $\lambda_V(T) = \lambda_I(T) = 0$ and we provide λ_V^- in terms of λ_V^+ because the computation happens in reverse time.

Equation (2) and table 2 suggest a simple algorithm, EventProp, to compute the gradient (algorithm 1). Notably, if the loss is voltage-independent (i.e., $l_V = 0$), the backward pass of the algorithm requires only the spike times t^{post} and the synaptic current of the firing neuron's at their respective firing times. The memory requirement of the algorithm therefore scales as $\mathcal{O}(rTN)$, where r is the firing rate and T is the total trial duration. In case of a voltage-dependent loss $l_V \neq 0$, the algorithm has to store the non-zero components of $\frac{\partial l_V}{\partial V}$ along the forward trajectory. Note that in many practical scenarios as found in deep learning, the loss l_V depends only on the state of a constant number of neurons, irrespective of network size. If l_V depends on the voltage of non-firing readout neurons, we have $l_V^+ = l_V^-$ and the corresponding term in the jump given in table 2 vanishes.

For voltage-independent losses (i.e., $l_V = 0$), all variables only need to be computed at spike times. In that case, EventProp can be computed in a purely event-based manner. Figure 1 illustrates how EventProp computes gradients for two LIF neurons where one neuron receives Poisson spike trains via 100 synapses and is connected to the other neuron via a single feed-forward weight w.

Algorithm 1 EventProp: Algorithm to compute eq. (2).

Require: Input spikes, losses l_{p}, l_V, parameters W, τ_{mem}, τ_{syn}, initial conditions $V(0)$, $I(0)$
  $\text{grad} \leftarrow 0$
  Compute neuron state trajectory (table 1) from $t = 0$ to $t = T$: ▷ Forward pass
    for all spikes k, store spike time t_k^{post} and the firing neuron's component of $I(t_k^{\text{post}})$
    if $l_V \neq 0$, also store $\frac{\partial l_V}{\partial V}$
  Compute adjoint state trajectory (table 2) from $t = T$ to $t = 0$: ▷ Backward pass
    accumulate
      $\text{grad}_{ij} \leftarrow \text{grad}_{ij} - \tau_{\text{syn}}(\lambda_I)_j$
    for each spike from neuron i to j
  return grad

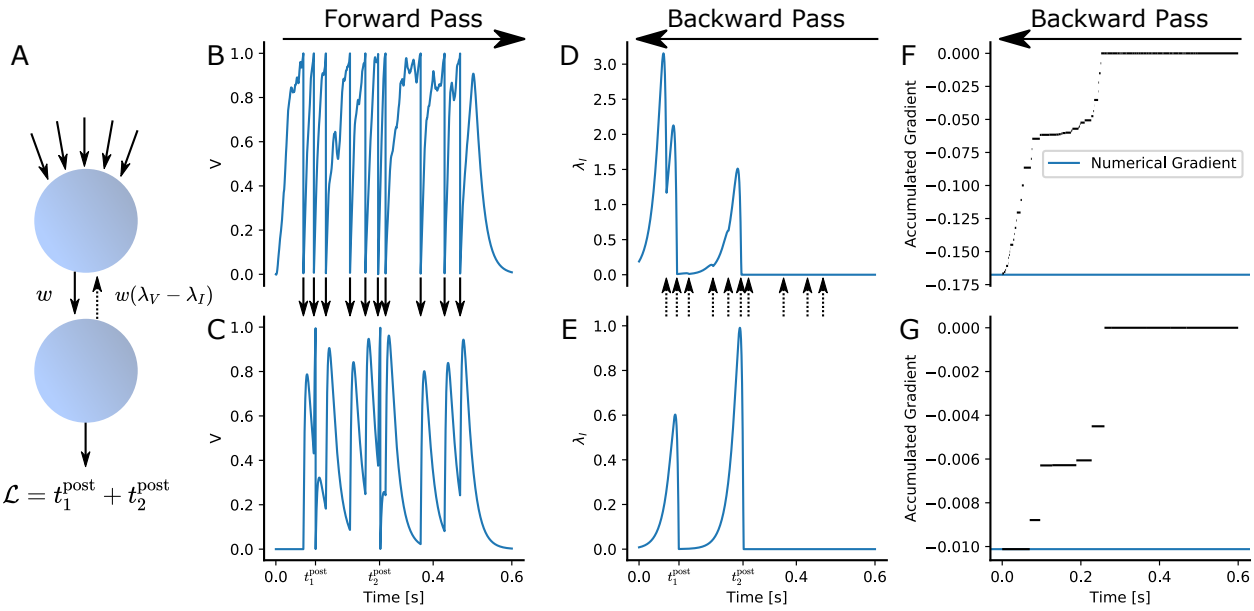

Figure 1: Illustration of EventProp-based gradient calculation in two LIF neurons connected with weight w and a spike-time dependent loss $\mathcal{L}$. The forward pass (B, C) computes the spike times for both neurons and the backward pass (D-G) backpropagates errors at spike times, yielding the gradient as given in eq. (2). **A**: The upper neuron receives 100 independent Poisson spike trains with frequency $200\,\text{Hz}$ across randomly initialized weights and is connected to the lower neuron via a single weight w. The loss $\mathcal{L}$ is a sum of the spike times of the lower neuron. **B, C**: Membrane potential of upper and lower neuron. Spike times of upper neuron are indicated using arrows. **D, E**: Adjoint variable λ_I of upper and lower neuron. The lower neuron backpropagates its error signal $\lambda_V - \lambda_I$ at the upper neuron's spike times (indicated by arrows). **F, G**: Accumulated gradient for one of the 100 input weights of the upper neuron and the weight w connecting the upper and lower neuron. EventProp computes the adjoint variables from $t = T$ to $t = 0$ and accumulates the gradients by sampling $-\tau_{\text{syn}}\lambda_I$ when spikes are transmitted across the respective weight. The numerical gradients were calculated via central differences and match the final accumulated gradients up to a relative deviation of less than 10^{-7}.

2.3 Derivation

Our derivation is based on the observation that the system defined in table 1 has an equivalent formulation in the language of control theory, where the state variables (membrane potentials) cause discrete transitions, with continuous dynamics between the transitions. The following derivation is specific the model given in table 1. A fully general derivation of sensitivity analysis in hybrid systems can be found in Galán et al. (1999) or Serban and Recuero (2019).

The differential equations defining the free dynamics in implicit form are

$$f_V \equiv \tau_{\text{mem}}\dot{V} + V - I = 0, \tag{3a}$$

$$f_I \equiv \tau_{\text{syn}}\dot{I} + I = 0, \tag{3b}$$

where f_V, f_I are again vectors of size N. We now split up the integral in eq. (1) at the spike times t^{post} and add vectors of Lagrange multipliers λ_V, λ_I that fix the system dynamics f_V, f_I between transitions.

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = \frac{\mathrm{d}}{\mathrm{d}w_{ij}}\left[l_{\text{p}}(t^{\text{post}}) + \sum_{k=0}^{N_{\text{post}}} \int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \left[l_V(V,t) + \lambda_V \cdot f_V + \lambda_I \cdot f_I \right] \mathrm{d}t \right], \tag{4}$$

where we set $t_0^{\text{post}} = 0$ and $t_{N_{\text{post}}+1}^{\text{post}} = T$ and $x \cdot y$ is the dot product of two vectors x, y. Using eq. (3) and the short-hand notation $s_X \equiv \frac{\partial X}{\partial w_{ij}}$ (the *parameter sensitivities* in the language of control theory), we have, as per Gronwall's theorem (Gronwall, 1919),

$$\frac{\partial f_V}{\partial w_{ij}} = \tau_{\text{mem}}\dot{s}_V + s_V - s_I, \tag{5a}$$

$$\frac{\partial f_I}{\partial w_{ij}} = \tau_{\text{syn}}\dot{s}_I + s_I, \tag{5b}$$

where we have commuted the derivatives $\frac{\partial}{\partial w_{ij}}\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial w_{ij}}$. The gradient then becomes, by application of the Leibniz integral rule,

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = \sum_{k=0}^{N_{\text{post}}} \Bigg[&\int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \left[\frac{\partial l_V}{\partial V} \cdot s_V + \lambda_V \cdot (\tau_{\text{mem}}\dot{s}_V + s_V - s_I) + \lambda_I \cdot (\tau_{\text{syn}}\dot{s}_I + s_I) \right] \mathrm{d}t \\ &+ \frac{\partial l_{\text{p}}}{\partial t_k^{\text{post}}}\frac{\mathrm{d}t_k^{\text{post}}}{\mathrm{d}w_{ij}} + l_{V,k+1}^{-}\frac{\mathrm{d}t_{k+1}^{\text{post}}}{\mathrm{d}w_{ij}} - l_{V,k}^{+}\frac{\mathrm{d}t_k^{\text{post}}}{\mathrm{d}w_{ij}} \Bigg],\end{aligned} \tag{6}$$

where $l_{V,k}^{\pm}$ is the voltage-dependent loss evaluated before $(-)$ or after $(+)$ the transition and we have used that $f_V = f_I = 0$ along all considered trajectories. Using partial integration, we have

$$\int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \lambda_V \dot{s}_V \mathrm{d}t = -\int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \dot{\lambda}_V \cdot s_V \mathrm{d}t + \left[\lambda_V \cdot s_V\right]_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}}, \tag{7}$$

$$\int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \lambda_I \dot{s}_I \mathrm{d}t = -\int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \dot{\lambda}_I \cdot s_I \mathrm{d}t + \left[\lambda_I \cdot s_I\right]_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}}. \tag{8}$$

Using the short-hand notation $\frac{\mathrm{d}t_k^{\text{post}}}{\mathrm{d}w_{ij}} \equiv \tau_k^{\text{post}}$ and collecting terms in s_V, s_I, we have

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = \sum_{k=0}^{N_{\text{post}}} \Bigg[&\int_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} \left[\left(\frac{\partial l_V}{\partial V} - \tau_{\text{mem}}\dot{\lambda}_V + \lambda_V \right) \cdot s_V + \left(-\tau_{\text{syn}}\dot{\lambda}_I + \lambda_I - \lambda_V \right) \cdot s_I \right] \mathrm{d}t \\ &+ \frac{\partial l_{\text{p}}}{\partial t_k^{\text{post}}}\tau_k^{\text{post}} + \tau_{\text{mem}}\left[\lambda_V \cdot s_V\right]_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} + \tau_{\text{syn}}\left[\lambda_I \cdot s_I\right]_{t_k^{\text{post}}}^{t_{k+1}^{\text{post}}} + l_{V,k+1}^{-}\tau_{k+1}^{\text{post}} - l_{V,k}^{+}\tau_k^{\text{post}} \Bigg].\end{aligned} \tag{9}$$

This form allows us to set the dynamics of the adjoint variables between transitions. Since the integration of the adjoint variables is done from $t = T$ to $t = 0$ in practice (i.e., reverse in time), it is practical to transform the time derivative as $\frac{\mathrm{d}}{\mathrm{d}t} \to -\frac{\mathrm{d}}{\mathrm{d}t}$. Denoting the new time derivative by $'$, we have

$$\tau_{\text{mem}}\lambda_V' = -\lambda_V - \frac{\partial l_V}{\partial V} \tag{10a}$$

$$\tau_{\text{syn}}\lambda_I' = -\lambda_I + \lambda_V. \tag{10b}$$

The integrand therefore vanishes along the trajectory and we are left with a sum over the transitions. Since the initial conditions of V and I are assumed to be parameter independent, we have $s_V = s_I = 0$ at $t = 0$. We set the initial condition for the adjoint variables to be $\lambda_V = \lambda_I = 0$ at $t = T$. We are therefore left with a sum over transitions ξ_k evaluated at the transition times t_k^{post},

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = \sum_{k=1}^{N_{\text{post}}} \xi_k \tag{11}$$

with the definition

$$\begin{aligned}\xi_k \equiv \frac{\partial l_{\text{p}}}{\partial t_k^{\text{post}}} \tau_k^{\text{post}} + l_{V,k}^- \tau_k^{\text{post}} - l_{V,k}^+ \tau_k^{\text{post}} \\ + \left[\tau_{\text{mem}}(\lambda_V^- \cdot s_V^- - \lambda_V^+ \cdot s_V^+) + \tau_{\text{syn}}(\lambda_I^- \cdot s_V^- - \lambda_I^+ \cdot s_I^+)\right]\Big|_{t_k^{\text{post}}}.\end{aligned} \tag{12}$$

We proceed by deriving the relationship between the adjoint variables before and after each transition. Since the computation of the adjoint variables happens in reverse time in practice, we provide λ^- in terms of λ^+.

Consider a spike caused by the nth neuron, with all other neurons $m \neq n$ remaining silent. We start by first deriving the relationships between s_V^+, s_V^- and s_I^+, s_I^-.

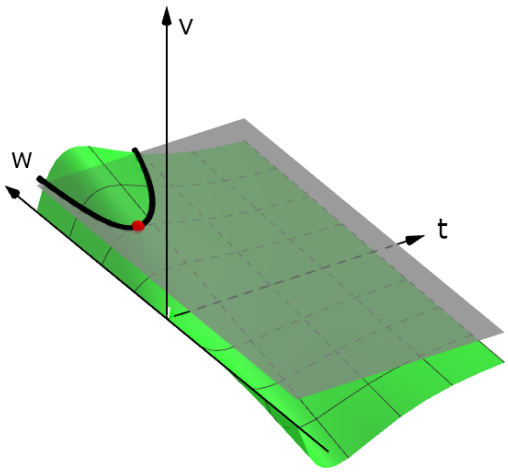

Figure 2: In this sketch, the relation $v(t, w) - \vartheta = 0$ defines an implicit function (black line along which $\mathrm{d}v = 0$). The critical point where the gradient diverges is shown in red.

Membrane Potential Transition By considering the relations between V^+, V^- and $\dot{V}^+$, $\dot{V}^-$, we can derive the relation between s_V^+ and s_V^- at each spike. Each spike at t^{post} is triggered by a neuron's membrane potential crossing the threshold. We therefore have, at t^{post},

$$v_n^- - \vartheta = 0, \tag{13}$$

where $v_n^- = (V^-)_n$ is the membrane potential of the spiking neuron n before the transition. This relation defines a differentiable function of w_{ij} via the implicit function theorem (illustrated in fig. 2, see also Yang et al., 2014). Differentiation of this relation yields

$$(s_V^-)_n + \dot{v}_n^- \tau^{\text{post}} = 0. \tag{14}$$

Note that corresponding relations were previously used to derive gradient-based learning rules for spiking neuron models (Bell and Parra, 2005; Bohte et al., 2000; Booij and tat Nguyen, 2005; Xu et al., 2013; Florian, 2012); in contrast to the suggestion in Bohte et al. (2000), eq. (14) is not an approximation but rather an exact relation at all non-critical parameters and invalid at all critical parameters.

Since we only allow transitions for $\dot{v}_n \neq 0$, we have

$$\tau^{\text{post}} = -\frac{1}{\dot{v}_n^-}(s_V^-)_n. \tag{15}$$

Because the spiking neuron's membrane potential is reset to zero, we have

$$v_n^+ = v_n^- - \vartheta. \tag{16}$$

This implies, again via the implicit function theorem, the relationship between the parameter sensitivities for the membrane potential before and after the spike,

$$(s_V^+)_n + \dot{v}_n^+ \tau^{\text{post}} = (s_V^-)_n + \dot{v}_n^- \tau^{\text{post}} \tag{17}$$

$$\Rightarrow (s_V^+)_n = \frac{\dot{v}_n^+}{\dot{v}_n^-}(s_V^-)_n. \tag{18}$$

Since we have $v_m^+ = v_m^-$ for all other, non-spiking neurons $m \neq n$, it holds that

$$(s_V^+)_m + \dot{v}_m^+ \tau^{\text{post}} = (s_V^-)_m + \dot{v}_m^- \tau^{\text{post}}. \tag{19}$$

Because the spiking neuron n causes the synaptic current of all neurons $m \neq n$ to jump by w_{nm}, we have

$$\tau_{\text{mem}} \dot{v}_m^+ = \tau_{\text{mem}} \dot{v}_m^- + w_{nm} \tag{20}$$

and therefore get with eq. (14)

$$(s_V^+)_m = (s_V^-)_m - \tau_{\text{mem}}^{-1} w_{nm} \tau^{\text{post}} \tag{21}$$

$$= (s_V^-)_m + \frac{1}{\tau_{\text{mem}} \dot{v}_n^-} w_{nm} (s_V^-)_n. \tag{22}$$

Synaptic Current Transition The spiking neuron n causes the synaptic current of all neurons $m \neq n$ to jump by the corresponding weight w_{nm}. We therefore have

$$(I^+)_m = (I^-)_m + w_{\text{nm}}. \tag{23}$$

By differentiation, this relation implies the consistency equations for the sensitivities s_I with respect to the considered weight w_{ij},

$$(s_I^+)_m + (\dot{I}^+)_m \tau^{\text{post}} = (s_I^-)_m + (\dot{I}^-)_m \tau^{\text{post}} + \delta_{in} \delta_{jm}, \tag{24}$$

where δ_{ij} is the Kronecker delta. Because

$$\tau_{\text{syn}} (\dot{I}^+)_m = \tau_{\text{syn}} (\dot{I}^-)_m - w_{nm}, \tag{25}$$

we get with eq. (14)

$$(s_I^+)_m = (s_I^-)_m + \tau_{\text{syn}}^{-1} w_{nm} \tau^{\text{post}} + \delta_{in} \delta_{jm} \tag{26}$$

$$= (s_I^-)_m - \frac{1}{\tau_{\text{syn}} \dot{v}_n^-} w_{nm} (s_V^-)_n + \delta_{in} \delta_{jm}. \tag{27}$$

With $(I^+)_n = (I^-)_n$ and $(\dot{I}^+)_n = (\dot{I}^-)_n$, we have

$$(s_I^+)_n = (s_I^-)_n. \tag{28}$$

Using the parameter sensitivity relations from eqs. (15), (18), (22), (27) and (28) in the transition equation eq. (12), we now derive relations between the adjoint variables. Collecting terms in the sensitivities and writing the index of the spiking neuron for the kth spike as $n(k)$, we have

$$\begin{aligned}
\xi_k = \Bigg[& \sum_{m \neq n(k)} \left[\tau_{\text{mem}} (\lambda_V^- - \lambda_V^+)_m (s_V^-)_m + \tau_{\text{syn}} (\lambda_I^- - \lambda_I^+)_m (s_I^-)_m - \tau_{\text{syn}} \delta_{in(k)} \delta_{jm} (\lambda_I^+)_m \right] \\
& + (s_V^-)_{n(k)} \left[\tau_{\text{mem}} \left(\lambda_V^- - \frac{\dot{v}_{n(k)}^+}{\dot{v}_{n(k)}^-} \lambda_V^+ \right)_{n(k)} + \frac{1}{\dot{v}_{n(k)}^-} \left(\sum_{m \neq n(k)} w_{n(k)m} (\lambda_I^+ - \lambda_V^+)_m - \frac{\partial l_{\text{p}}}{\partial t_k^{\text{post}}} + l_V^+ - l_V^- \right) \right] \\
& + \tau_{\text{syn}} (\lambda_I^- - \lambda_I^+)(s_I^-)_{n(k)} \Bigg] \Bigg|_{t_k^{\text{post}}}.
\end{aligned} \tag{29}$$

This form dictates the jumps of the adjoint variables for the spiking neuron n and all other, silent neurons m,

$$(\lambda_V^-)_n = \frac{\dot{v}_n^+}{\dot{v}_n^-} (\lambda_V^+)_n + \frac{1}{\tau_{\text{mem}} \dot{v}_n^-} \left[\sum_{m \neq n} w_{nm} (\lambda_V^+ - \lambda_I^+)_m + \frac{\partial l_{\text{p}}}{\partial t_k^{\text{post}}} + l_V^- - l_V^+ \right], \tag{30a}$$

$$(\lambda_V^-)_m = (\lambda_V^+)_m, \tag{30b}$$

$$\lambda_I^- = \lambda_I^+. \tag{30c}$$

With these jumps, the gradient reduces to

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = -\tau_{\text{syn}} \sum_{k=1}^{N_{\text{post}}} \delta_{in(k)} (\lambda_I)_j \tag{31}$$

$$= -\tau_{\text{syn}} \sum_{\text{spikes from } i} (\lambda_I)_j. \tag{32}$$

Summary The free adjoint dynamics between spikes are given by eq. (10) while spikes cause jumps given by eq. (30). The gradient for a given weight samples the post-synaptic neuron's λ_I when spikes are transmitted across the corresponding synapse (eq. (31)). Since we can identify

$$\frac{\dot{v}_n^+}{\dot{v}_n^-} = \frac{\dot{v}_n^+ - \dot{v}_n^-}{\dot{v}_n^-} + 1 = \frac{\vartheta}{\tau_{\text{mem}} \dot{v}_n^-} + 1 \tag{33}$$

the derived solution is equivalent to eq. (2) and table 2.

Fixed Input Spikes If a given neuron i is subjected to a fixed pre-synaptic spike train across a synapse with weight w_{input}, the transition times are fixed and the adjoint variables do not experience jumps. The gradient simply samples the neuron's λ_I at the times of spike arrival,

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{\text{input}}} = -\tau_{\text{syn}} \sum_{\text{input spikes}} (\lambda_I)_i. \tag{34}$$

Coincident Spikes The derivation above assumes that only a single neuron of the recurrent network spikes at a given t_k^{post}. In general, coincident spikes may occur. If neurons a and b spike at the same time and the times of their respective threshold crossing vary independently as function of w_{ij}, the derivation above still holds, with both neuron's λ_V experiencing a jump as in eq. (30a).

3 Simulation Results

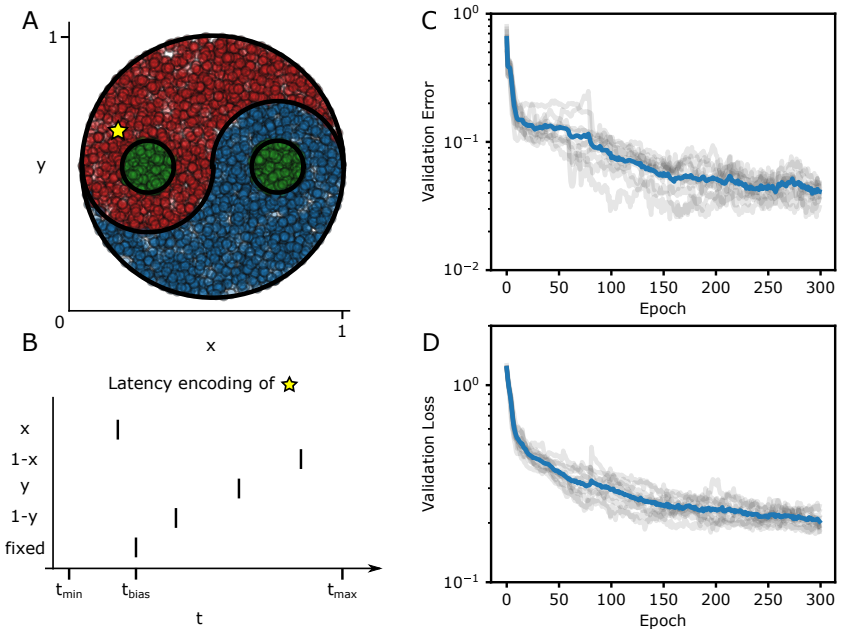

Figure 3: We used EventProp and a time-to-first-spike loss function to train a two-layer LIF network on the Yin-Yang dataset. **A**: Illustration of the two-dimensional dataset (adapted from Göltz et al. (2019)). The three different classes are shown in red, green and blue. **B**: Illustration of spike-time latency encoding using the datapoint marked by a star in A. **C, D**: Training results in terms of validation error and loss averaged over 10 different random seeds (individual traces shown as grey lines).

We demonstrate learning using EventProp using an event-based simulator. The simulator uses an event queue and root-bracketing to compute post-synaptic spike times in the forward pass and back-propagates errors by attaching error signals to events in the backward pass. The gradients computed in this way are used for stochastic gradient descent via the Adam optimizer (Kingma and Ba, 2014). We use a two-layer neural network and encode the Yin-Yang dataset (Kriener, 2020) using input spike latencies. This dataset is non-linearly separable, with a shallow classifier achieving around 64% accuracy, and it therefore requires a hidden layer and backpropagation of errors for reasonable classification

accuracy. Consider that in contrast, the widely used MNIST dataset can be classified using a linear classifier with at least 88% accuracy (Lecun et al., 1998).

In analogy to Göltz et al. (2019), we use a cross-entropy loss defined using the first output spike times per neuron,

$$\mathcal{L}(t^{\text{post}}, l) = -\log\left[\frac{\exp\left(-t_l^{\text{post}}/(\xi\tau_{\text{syn}})\right)}{\sum_k \exp\left(-t_k^{\text{post}}/(\xi\tau_{\text{syn}})\right)}\right] + \alpha\left[\exp\left(\frac{t_l^{\text{post}}}{\beta\tau_{\text{syn}}}\right) - 1\right], \tag{35}$$

where t_k^{post} is the first spike time of neuron k, l is the index of the correct label and the second term is a regularization term that encourages early spiking of the label neuron. Each two-dimensional datapoint of the dataset (x, y) is transformed into $(x, 1-x, y, 1-y)$ and encoded using spike latencies in the interval $[t_{\text{min}}, t_{\text{max}}]$ (see fig. 3 B). We added a fixed bias spike at time t_{bias} for a total of 5 input spikes per datapoint. If the activity of one of the two layers fell below a certain threshold for a given minibatch, we incremented all weights by a constant amount Δw_{bump}.

Training results are shown in fig. 3. All parameters used are given in fig. 4. After training, the validation accuracy was $(95.9 \pm 0.5)\,\%$ (mean and standard deviation over 10 different random seeds). This is comparable to the results shown in Göltz et al. (2019), who report $(95.9 \pm 0.7)\,\%$ accuracy with a similar architecture. Note that the results shown here can likely be improved using systematic hyperparameter tuning.

4 Discussion

We have derived, and provided an algorithm (EventProp) to compute, the gradient of an arbitrary loss function of a spiking neural network composed of leaky integrate-and-fire neurons. An obvious issue with gradient-descent based learning in the context of spiking networks is that the gradient diverges at the critical points in parameter space (note the $\dot{v}^{-1}$ term in the jump term given in table 2; this term diverges as the membrane potential becomes tangent to the threshold and we have $\dot{v} \to 0$). Indeed, this is a known issue in the broader context of optimal control of dynamical systems with parameter-dependent state transitions (Barton and Lee, 2002; Galán et al., 1999). While this divergence can be mitigated using gradient clipping in practice, commonly considered loss functions lead to learning dynamics that are ignorant with respect to these critical points and are therefore unable to selectively recruit additional spikes or dismiss existing spikes. It is therefore plausible that surrogate gradient methods which continuously transmit errors across spiking neurons represent a form of implicit regularization. Neftci et al. (2019) report that the surrogate gradient approximates the true gradient in a minimalistic binary classification task while at the same time remaining finite and continuous along an interpolation path in weight space. Hybrid algorithms that combine the exact gradient with explicit regularization techniques could be a direction for future research and provide more principled learning algorithms as compared to ad-hoc replacements of threshold functions.

This work is based on the widely used LIF neuron model. Adjoint equations for models with fixed refractory periods, adaptive thresholds, synaptic plasticity or of multi-compartment neuron models can be derived in an analogous way (Pehle, 2020). While the absence of explicit solutions to the resulting differential equations can prevent the use of typical event-based simulation approaches as the one used in this work, such extensions can significantly enhance the computational capabilities of spiking networks. For example, Bellec et al. (2018) uses adaptive thresholds to implement LSTM-like memory cells in a recurrent spiking neural network.

Neuromorphic hardware is an increasingly active research subject (e.g., Aamir et al., 2018; Davies et al., 2018; Furber et al., 2014; Neckar et al., 2019; Moradi et al., 2018; Merolla et al., 2014; Pei et al., 2019; Billaudelle et al., 2019; Feldmann et al., 2019; Boybat et al., 2017; Wunderlich et al., 2019) and implementing EventProp on such hardware is a natural consideration. The adjoint dynamics as given in table 2 represent a spiking neural network which, instead of spiking dynamically, transmits errors at fixed times t^{post} that are scaled with factors $\dot{v}^{-1}$ retained from the forward pass. Therefore, a neuromorphic implementation could store spike times and scaling factors locally at each neuron, where they could be combined with the dynamic error signal ($\lambda_V - \lambda_I$ in table 2) in the backward pass. This requires a possibility to read out neuronal state variables both in the forward and backward pass (membrane potential, synaptic current). Transmitting error signals across the network in a neuromorphic system requires similar considerations to distributed event-based simulators. The error signals associated with each spike could be propagated using message passing with gather/scatter primitives. As mentioned above, EventProp can be extended to multi-compartment neuron models as used in a recent neuromorphic architecture (Schemmel et al., 2017).

We have demonstrated learning using EventProp using a two-layer feed-forward architecture and a simple non-linearly separable dataset. The algorithm can, however, compute the gradient for arbitrary recurrent or convolutional architectures (Pehle, 2020). Its computational and spatial complexity scales linearly with network size, analogous to backpropagation in non-spiking artificial neural networks. The performance in more complex tasks therefore hinges on the general efficacy of gradient descent in spiking networks. As mentioned above, naive gradient descent using loss functions

defined in terms of spike times or membrane potentials ignores the presence of critical parameters where spikes appear or disappear. We suggest that studying regularization techniques which deal with this fundamental issue in a targeted manner could enable powerful learning algorithms for spiking networks. By providing a theoretical foundation for backpropagation in spiking networks, we have laid the groundwork for future research that combines such regularization techniques with the computation of exact gradients.

Contributions

CP had the initial idea to apply the adjoint method to spiking neuron models and conceived the approach. CP and TW derived the adjoint equations for LIF neurons and the resulting EventProp algorithm. TW implemented the event based simulation code. TW conceived and implemented the example experiment and analysed its results. CP and TW wrote and edited the manuscript.

Acknowledgements

We thank Korbinian Schreiber, Laura Kriener, Julian Göltz and Mihai Petrovici for helpful discussions.

Funding

The research has received funding from the EC Horizon 2020 Framework Programme under Grant Agreements 785907 and 945539 (HBP).

References

S. A. Aamir, Y. Stradmann, P. Müller, C. Pehle, A. Hartel, A. Grübl, J. Schemmel, and K. Meier. An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12):4299–4312, 2018.

P. I. Barton and C. K. Lee. Modeling, simulation, sensitivity analysis, and optimization of hybrid systems. *ACM Trans. Model. Comput. Simul.*, 12(4):256–289, Oct. 2002. ISSN 1049-3301. doi: 10.1145/643120.643122. URL `https://doi.org/10.1145/643120.643122`.

A. J. Bell and L. C. Parra. Maximising sensitivity in a spiking network. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 121–128. MIT Press, 2005. URL `http://papers.nips.cc/paper/2674-maximising-sensitivity-in-a-spiking-network.pdf`.

G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, pages 787–797, 2018.

S. Billaudelle, Y. Stradmann, K. Schreiber, B. Cramer, A. Baumbach, D. Dold, J. Göltz, A. F. Kungl, T. C. Wunderlich, A. Hartel, E. Müller, O. Breitwieser, C. Mauch, M. Kleider, A. Grübl, D. Stöckel, C. Pehle, A. Heimbrecht, P. Spilger, G. Kiene, V. Karasenko, W. Senn, M. A. Petrovici, J. Schemmel, and K. Meier. Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, 2019.

S. M. Bohte, J. N. Kok, and J. A. La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, volume 48, pages 17–37, 2000.

O. Booij and H. tat Nguyen. A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, 95(6):552 – 558, 2005. ISSN 0020-0190. doi: https://doi.org/10.1016/j.ipl.2005.05.023. URL `http://www.sciencedirect.com/science/article/pii/S0020019005001560`. Applications of Spiking Neural Networks.

I. Boybat, M. Gallo, N. S.R., T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou. Neuromorphic computing with multi-memristive synapses. *Nature Communications*, 9, 11 2017. doi: 10.1038/s41467-018-04933-y.

A. M. Bradley. Pde-constrained optimization and the adjoint method, 2019. URL `https://cs.stanford.edu/~ambrad/adjoint_tutorial.pdf`.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8529–8533, 2020.

M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

W. De Backer. Jump conditions for sensitivity coefficients. *IFAC Proceedings Volumes*, 1(3):168 – 175, 1964. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)69603-4. URL http://www.sciencedirect.com/science/article/pii/S1474667017696034. International Symposium on Sensitivity Methods in Control Theory, Dubrovnik, Yugoslavia, August 31-September 5, 1964.

S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1604850113. URL https://www.pnas.org/content/113/41/11441.

J. Feldmann, N. Youngblood, C. Wright, H. Bhaskaran, and W. Pernice. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature*, 569:208–214, 05 2019. doi: 10.1038/s41586-019-1157-8.

R. V. Florian. The chronotron: A neuron that learns to fire temporally precise spike patterns. *PLOS ONE*, 7(8):1–27, 08 2012. doi: 10.1371/journal.pone.0040233. URL https://doi.org/10.1371/journal.pone.0040233.

S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

S. Galán, W. F. Feehery, and P. I. Barton. Parametric sensitivity functions for hybrid discrete/continuous systems. *Applied Numerical Mathematics*, 31(1):17 – 47, 1999. ISSN 0168-9274. doi: https://doi.org/10.1016/S0168-9274(98)00125-1. URL http://www.sciencedirect.com/science/article/pii/S0168927498001251.

W. Gerstner and W. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, 2002. ISBN 9780521890793.

T. H. Gronwall. Note on the derivatives with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, 20(4):292–296, 1919. ISSN 0003486X. URL http://www.jstor.org/stable/1967124.

R. Gütig. Spiking neurons can discover predictive features by aggregate-label learning. *Science*, 351(6277), 2016. ISSN 0036-8075. doi: 10.1126/science.aab4113. URL https://science.sciencemag.org/content/351/6277/aab4113.

R. Gütig and H. Sompolinsky. The tempotron: a neuron that learns spike timing–based decisions. *Nature Neuroscience*, 9(3):420–428, Mar 2006. ISSN 1546-1726. doi: 10.1038/nn1643. URL https://doi.org/10.1038/nn1643.

J. Göltz, L. Kriener, A. Baumbach, S. Billaudelle, O. Breitwieser, B. Cramer, D. Dold, A. F. Kungl, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici. Fast and deep: energy-efficient neuromorphic learning with first-spike times, 2019.

D. Huh and T. J. Sejnowski. Gradient descent for spiking neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1433–1443. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7417-gradient-descent-for-spiking-neural-networks.pdf.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

L. Kriener. Yin-yang dataset. https://github.com/lkriener/yin_yang_data_set, 2020.

Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Transactions on Biomedical Circuits and Systems*, 12(1):106–122, 2018.

H. Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–9, 2017. ISSN 2162-2388. doi: 10.1109/tnnls.2017.2726060. URL http://dx.doi.org/10.1109/TNNLS.2017.2726060.

A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen. Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proceedings of the IEEE*, 107(1):144–164, 2019.

E. O. Neftci, H. Mostafa, and F. Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

C. Pehle. Adjoint equations of spiking neural networks. In preparation, 2020.

J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, and L. Shi. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572:106, 08 2019. doi: 10.1038/s41586-019-1424-8.

M. Pfeiffer and T. Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774, 2018. ISSN 1662-453X. doi: 10.3389/fnins.2018.00774. URL https://www.frontiersin.org/article/10.3389/fnins.2018.00774.

L. S. Pontryagin. *Mathematical theory of optimal processes*. Routledge, 1962.

K. Roy, A. Jaiswal, and P. Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575:607–617, 11 2019. doi: 10.1038/s41586-019-1677-2.

E. Rozenvasser. General sensitivity equations of discontinuous systems. *Automatika i telemekhanika*, 3:52–56, 1967.

E. Rozenwasser and R. Yusupov. *Sensitivity of Automatic Control Systems*. Control Series. CRC Press, 2019. ISBN 9781420049749. doi: 10.1201/9781420049749. URL https://doi.org/10.1201/9781420049749.

J. Schemmel, L. Kriener, P. Müller, and K. Meier. An accelerated analog neuromorphic hardware system emulating nmda-and calcium-based non-linear dendrites. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2217–2226. IEEE, 2017.

R. Serban and A. Recuero. Sensitivity analysis for hybrid systems and systems with memory. *Journal of Computational and Nonlinear Dynamics*, 14(9), Jul 2019. ISSN 1555-1423. doi: 10.1115/1.4044028. URL http://dx.doi.org/10.1115/1.4044028.

S. B. Shrestha and G. Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, pages 1412–1421, 2018.

A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47 – 63, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2018.12.002. URL http://www.sciencedirect.com/science/article/pii/S0893608018303332.

T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel, and et al. Demonstrating advantages of neuromorphic computation: A pilot study. *Frontiers in Neuroscience*, 13, Mar 2019. ISSN 1662-453X. doi: 10.3389/fnins.2019.00260. URL http://dx.doi.org/10.3389/fnins.2019.00260.

Y. Xu, X. Zeng, L. Han, and J. Yang. A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural networks : the official journal of the International Neural Network Society*, 43:99—113, July 2013. ISSN 0893-6080. doi: 10.1016/j.neunet.2013.02.003. URL https://doi.org/10.1016/j.neunet.2013.02.003.

W. Yang, D. Yang, and Y. Fan. A proof of a key formula in the error-backpropagation learning algorithm for multiple spiking neural networks. In Z. Zeng, Y. Li, and I. King, editors, *Advances in Neural Networks – ISNN 2014*, pages 19–26, Cham, 2014. Springer International Publishing. ISBN 978-3-319-12436-0.

F. Zenke and S. Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

A Simulation Parameters

Symbol	Description	Value
τ_{mem}	Membrane Time Constant	$20\,\text{ms}$
τ_{syn}	Synaptic Time Constant	$5\,\text{ms}$
ϑ	Threshold	1
	Input Size	5
	Hidden Size	200
	Output Size	3
t_{bias}	Bias Time	$20\,\text{ms}$
$t_{\min}$	Minimum Time	$10\,\text{ms}$
$t_{\max}$	Maximum Time	$40\,\text{ms}$
	Hidden Weights Mean	2
	Hidden Weights Standard Deviation	1
	Output Weights Mean	0.4
	Output Weights Standard Deviation	0.4
	Minibatch Size	200
	Optimizer	Adam
β_1	Adam Parameter	0.9
β_2	Adam Parameter	0.999
ϵ	Adam Parameter	1×10^{-8}
η	Learning Rate	1×10^{-3}
	Allowed Ratio of Missing Spikes in Hidden Layer	0.15
	Allowed Ratio of Missing Spikes in Output Layer	0
Δw_{bump}	Weight Bump Value	1×10^{-4}
α	Regularization Factor	1×10^{-2}
ξ	First Time Constant Factor	0.4
β	Second Time Constant Factor	2

Figure 4: Simulation parameters used for the results shown in section 3.