# SuperSpike: Supervised learning in multi-layer spiking neural networks

**F. Zenke**
Department of Applied Physics
Stanford University
Stanford, CA 94305
fzenke@stanford.edu

**S. Ganguli**
Department of Applied Physics
Stanford University
Stanford, CA 94305
sganguli@stanford.edu

## Abstract

A vast majority of computation in the brain is performed by spiking neural networks. Despite the ubiquity of such spiking, we currently lack an understanding of how biological spiking neural circuits learn and compute *in-vivo*, as well as how we can instantiate such capabilities in artificial spiking circuits *in-silico*. Here we revisit the problem of supervised learning in temporally coding multi-layer spiking neural networks. First, by using a surrogate gradient approach, we derive SuperSpike, a nonlinear voltage-based three factor learning rule capable of training multi-layer networks of deterministic integrate-and-fire neurons to perform nonlinear computations on spatiotemporal spike patterns. Second, inspired by recent results on feedback alignment, we compare the performance of our learning rule under different credit assignment strategies for propagating output errors to hidden units. Specifically, we test uniform, symmetric and random feedback, finding that simpler tasks can be solved with any type of feedback, while more complex tasks require symmetric feedback. In summary, our results open the door to obtaining a better scientific understanding of learning and computation in spiking neural networks by advancing our ability to train them to solve nonlinear problems involving transformations between different spatiotemporal spike-time patterns.

## 1 Introduction

Neurons in biological circuits form intricate networks in which the primary mode of communication occurs through spikes. The theoretical basis for how such networks are sculpted by experience to give rise to emergent computations remains poorly understood. Consequently, building meaningful spiking models of brain-like neural networks in-silico is a largely unsolved problem. In contrast, the field of deep learning has made remarkable progress in building non-spiking convolutional networks which often achieve human-level performance at solving difficult tasks [1, 2]. Even though the details of how these artificial rate-based networks are trained may arguably be different from how the brain learns, several studies have begun to draw interesting parallels between the internal representations formed by deep neural networks and the recorded activity from different brain regions [3–6]. A major impediment to deriving a similar comparison at the spiking level is that we currently lack efficient ways of training spiking neural networks (SNNs), thereby limiting their applications to mostly small toy problems that do not fundamentally involve spatiotemporal spike time computations. For instance, only recently have some groups begun to train SNNs on datasets such as MNIST [7–10], whereas most previous studies have used smaller artificial datasets.

The difficulty in simulating and training SNNs originates from multiple factors. First, time is an indispensable component of the functional form of a SNN, as even individual stimuli and their associated outputs are spatiotemporal spike patterns, rather than simple spatial activation vectors.

Preprint currently pending peer review.

This fundamental difference necessitates the use of different cost functions from the ones commonly encountered in deep learning. Second, most spiking neuron models are inherently non-differentiable at spike time and the derivative of their output with respect to synaptic weights is zero at all other times. Third, the intrinsic self-memory of most spiking neurons introduced by the spike reset is difficult to treat analytically. Finally, credit assignment in hidden layers is problematic for two reasons: (i) it is technically challenging because efficient auto-differentiation tools are not available for most event-based spiking neural network frameworks, and (ii) the method of weight updates implemented by the standard backpropagation algorithm is thought to be biologically implausible [11, 12].

Several studies of multi-layer networks which build on the notion of "feedback alignment" [13] have recently illustrated that the strict requirements imposed by backpropagation on the feedback of error signals can be loosened substantially without a large loss of performance on standard benchmarks like MNIST [8, 9, 13–15]. While some of these studies have been performed using spiking networks, they still use effectively a rate-based approach in which a given input activity vector is interpreted as the firing rate of a set of input neurons [7–9, 16, 17]. While this approach is appealing because it can often be related directly to equivalent rate-based models with stationary neuronal transfer functions, it also largely ignores the idea that individual spike timing may carry additional information which could be crucial for efficient coding [18–21] and fast computation [22, 23].

In this paper we develop a novel learning rule to train multi-layer SNNs of deterministic leaky integrate-an-fire (LIF) neurons on tasks which fundamentally involve spatiotemporal spike pattern transformations. In doing so we go beyond the purely spatial rate-based activation vectors prevalent in deep learning. We further study how biologically more plausible strategies for deep credit assignment across multiple layers generalize to the enhanced context of more complex spatiotemporal spike-pattern transformations.

### 1.1 Prior work

Supervised learning of precisely timed spikes in single neurons and networks without hidden units has been studied extensively. Pfister et al. [24] have used a probabilistic escape rate model to deal with the hard non-linearity of the spike. Similar probabilistic approaches have also been used to derive spike timing dependent plasticity (STDP) from information maximizing principles [25, 26]. In contrast to that, ReSuMe [27] and SPAN [28] are deterministic approaches which can be seen as generalizations of the Widrow-Hoff rule to spiking neurons. In a similar vein, the Chronotron [29] learns precisely timed output spikes by minimizing the Victor-Pupura distance [30] to a given target output spike train. Similarly, Gardner and Grüning [31] and Albers et al. [32] have studied the convergence properties of rules that reduce the van Rossum distance by gradient descent. Moreover, Memmesheimer et al. [33] proposed a learning algorithm which achieves high capacity in learning long precisely timed spike trains in single units and recurrent networks. The problem of sequence learning in recurrent neural networks has also been studied as a variational learning problem [34, 35] and by combining adaptive control theory with heterogeneous neurons [36].

Supervised learning in SNNs without hidden units has also been studied for classification problems. For instance, Maass et al. [37] have used the p-delta rule [38] to train the readout layer of a liquid state machine. Moreover, the Tempotron [39, 40], which can be derived as a gradient-based approach [41], classifies large numbers of temporally coded spike patterns without explicitly specifying a target firing time.

Only a few works have embarked upon the problem of training SNNs with hidden units to process precisely timed input and output spike trains by porting backprop to the spiking domain. The main analytical difficulty in these approaches arises from partial derivatives of the form ${}^{\partial S_i(t)}/_{\partial w_{ij}}$ where $S_i(t) = \sum_k \delta(t - t_i^k)$ is the spike train of the hidden neuron $i$ and $w_{ij}$ is a hidden weight. SpikeProp [42] circumvents this problem by defining a differentiable expression on the firing time instead, on which standard gradient descent can be performed. However, such a method cannot learn starting from a quiescent state of no spiking, as the spike time is then ill-defined. While the original approach was limited to a single spike per neuron, multiple extensions of the algorithm exist, some of which also improve its convergence properties [43–46]. One extension of ReSuMe to multiple layers was proposed to overcome the requirement of single spikes [47]. More recently, the same group proposed another generalization of backprop to SNNs in Gardner et al. [48]. This stochastic approach can be seen as an extension of Pfister et al. [24] to multiple layers. In a similar flavour as Fremaux et al.

[49], this work approximates the partial derivative of the hidden spike trains by invoking the outer product of pre- and postsynaptic spike trains. Like in the case of SpikeProp, this limits the algorithm's applicability to cases in which hidden units are not quiescent.

In contrast to these previous works, our method permits to train multi-layer networks of *deterministic* LIF neurons to solve tasks involving spatiotemporal spike pattern transformations without the need for injecting noise even when hidden units are initially completely silent. To achieve this, we approximate the partial derivative of the hidden unit outputs as the product of the filtered presynaptic spike train and a nonlinear function of the postsynaptic *voltage* instead of the postsynaptic spike train. In the following section we explain the details of our approach.

## 2 Derivation of the SuperSpike plasticity model

To begin, we consider a single LIF neuron which we would like to emit a given target spike train $\hat{S}_i$ for a given stimulus. Formally, we can frame this problem as an optimization problem in which we want to minimize the van Rossum distance [31, 50] between $\hat{S}_i$ and the actual output spike train $S_i$,

$$L = \frac{1}{2} \int_{-\infty}^{t} ds \left[ \left( \alpha * \hat{S}_i - \alpha * S_i \right) (s) \right]^2 \tag{1}$$

where $\alpha$ is a smooth temporal convolution kernel. We use double exponential causal kernels throughout because they can be easily computed online and could be implemented as electrical or chemical traces in neurobiology. When computing the gradient of Eq. 1 with respect to the synaptic weights $w_{ij}$ we get

$$\frac{\partial L}{\partial w_{ij}} = -\int_{-\infty}^{t} ds \left[ \left( \alpha * \hat{S}_i - \alpha * S_i \right) (s) \right] \left( \alpha * \frac{\partial S_i}{\partial w_{ij}} \right) (s) \tag{2}$$

in which the derivative of a spike train $\partial S_i / \partial w_{ij}$ appears. This derivative is problematic because for most neuron models it is zero except at spike threshold at which it is not defined. Most existing training algorithms circumvent this problem by either performing optimization directly on the membrane potential $U_i$ or by introducing noise which renders the likelihood of the spike train $\langle S_i(t) \rangle$ a smooth function of the membrane potential. Here we combine the merits of both approaches by replacing the spike train $S_i(t)$ with a continuous auxiliary function $\sigma(U_i(t))$ of the membrane potential. For performance reasons, we choose $\sigma(U)$ to be the negative side of a fast sigmoid (Supplementary Methods), but other monotonic functions which increase steeply and peak at the spiking threshold (e.g. exponential) should work as well. Our auxiliary function yields the replacement

$$\frac{\partial S_i}{\partial w_{ij}} \quad \rightarrow \quad \sigma'(U_i) \frac{\partial U_i}{\partial w_{ij}}. \tag{3}$$

To further compute the derivative $\partial U_i / \partial w_{ij}$ in the expression above, we exploit the fact that for current-based LIF models the membrane potential $U_i(t)$ can be written in integral form as a spike response model (SRM0 [51]):

$$U_i(t) = \sum_j w_{ij} \left( \epsilon * S_j(t) \right) + \left( \eta * S_i(t) \right), \tag{4}$$

where we have introduced the causal membrane kernel $\epsilon$ which corresponds to the postsynaptic potential (PSP) shape and $\eta$ which captures spike dynamics and reset. Due to the latter, $U_i$ depends on its own past through its output spike train $S_i$. While this dependence does not allow us to compute the derivative $\frac{\partial U_i}{\partial w_{ij}}$ directly, it constitutes only a small correction to $U_i$ provided the firing rates are low. Such low firing rates not only seem physiologically plausible, but also can be easily achieved in practice by adding homeostatic mechanisms that regularize neuronal activity levels. Neglecting the second term simply yields the filtered presynaptic activity $\frac{\partial U_i}{\partial w_{ij}} \approx (\epsilon * S_j(t))$ which can be interpreted as the concentration of neurotransmitters at the synapse. Substituting this approximation back into Eq. 2, the gradient descent learning rule for a single neuron takes the form

$$\frac{\partial w_{ij}}{\partial t} = r \int_{-\infty}^{t} ds \quad \underbrace{e_i(s)}_{\text{Error signal}} \quad \underbrace{\alpha * \left( \underbrace{\sigma'(U_i(s))}_{\text{Post}} \underbrace{(\epsilon * S_j)(s)}_{\text{Pre}} \right)}_{\equiv \lambda_{ij}(s)}, \tag{5}$$

where we have introduced the learning rate $r$ and short notation for the output error signal $e_i(s) \equiv \alpha * (\hat{S}_i - S_i)$ and the eligibility trace $\lambda_{ij}$. In practice we evaluate the expression on minibatches and we often use a per-parameter learning rate $r_{ij}$ closely related to RMSprop [52] to speed up learning.

Equation 5 corresponds to the SuperSpike learning rule for output neuron $i$. However, by redefining the error signal $e_i$ as a feedback signal, we will use the same rule for hidden units as well. Before we move on to testing this learning rule, we first state a few of its noteworthy properties: (i) it has a Hebbian term which combines pre- and postsynaptic activity in a multiplicative manner, (ii) the learning rule is voltage-based, (iii) it is a nonlinear Hebbian rule due to the occurrence of $\sigma'(U_i)$, (iv) the causal convolution with $\alpha$ acts as an eligibility trace to solve the distal reward problem due to error signals arriving *after* an error was made [53], and, (v) it is a three factor rule in which the error signal plays the role of a third factor [54]. Unlike most existing three-factor rules, however, the error signal is specific to the postsynaptic neuron, an important point which we will return to later.

## 3 Numerical experiments

To test whether Equation 5 could be used to train a single neuron to emit a predefined target spike pattern, we simulated a single LIF neuron which received a set of 100 spike trains as inputs. The target spike train was chosen as 5 equidistant spikes over the interval of 500ms. The inputs were drawn as Poisson spike trains that repeated every 500ms. We initialized the weights in a regime were the output neuron only showed sub-threshold dynamics, but did not spike (Fig. 1a). Previous methods, starting from this quiescent state, would require the introduction of noise to generate spiking, which would in turn retard the speed with which precise output spike times could be learned. Finally, weight updates were computed by evaluating the integral in Eq. 5 over a fixed interval and scaling the resulting value with the learning rate (Supplementary Methods). After 500 trials, corresponding to 250s of simulated time, the output neuron had learned to produce the desired output spike train (Fig. 1b). However, fewer trials could generate good approximations to the target spike train (Fig. 1c).

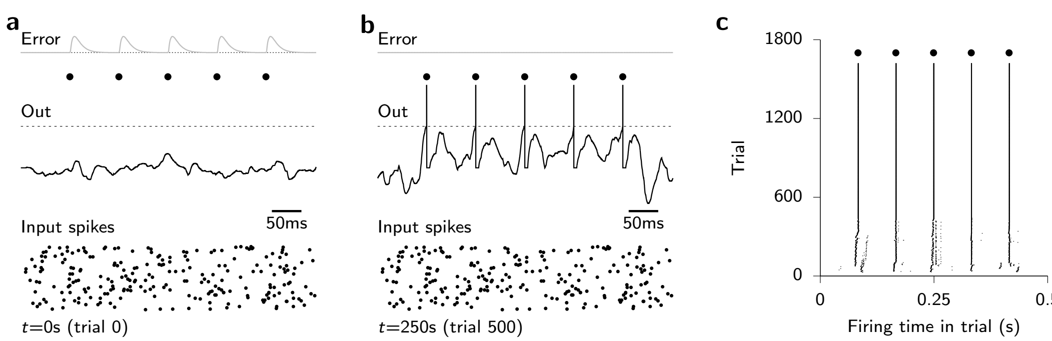


Figure 1: SuperSpike learning of precisely timed output spikes for a single output neuron. (**a**) Snapshot of initial network activity. Bottom panel: Spike raster of the input activity. Middle panel: The membrane potential of the output neuron (solid black line) and its firing threshold (dashed line). Target spikes are shown as black points. Top panel: Error signal (gray solid line). Zero error is indicated for reference as dotted line. (**b**) Same as in (a), but after 250s of SuperSpike learning. (**c**) Spike timing plot showing the temporal evolution of per-trial firing times during training.

### 3.1 Learning in multi-layer spiking neural networks

Having established that our rule can efficiently transform complex spatiotemporal input spike patterns to precisely timed output spike trains in a network without hidden units, we next investigated how well the same rule would perform in multilayer networks. The form of Equation 5 suggests a straight forward extension to hidden layers in analogy to backpropagation of error. Namely, we can use the same learning rule (Eq. 5) for hidden units, with the modification that that $e_i(t)$ becomes a complicated function which depends on the weights and future activity of all downstream neurons. However, this non-locality in space and time presents serious problems, both in terms of biological plausibility and technical feasibility. Technically, this computation requires either backpropagation

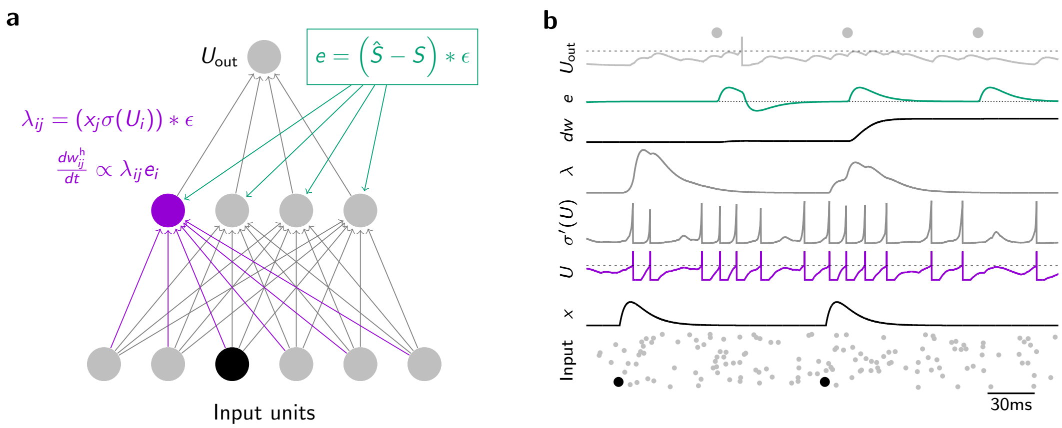


Figure 2: (**a**) Schematic illustration of SuperSpike learning in a network with a hidden layer. Spikes generated by the lower input layer are propagated through the hidden layer in the middle to the output layer at the top. (**b**) Temporal evolution of the dynamical quantities involved in updating a single synaptic weight from an input to a hidden layer unit. For brevity we have suppressed the neuron indices on all the variables. Input spikes (bottom panel) and their associated post-synaptic potentials $x$ sum to the membrane voltage in the hidden unit (purple). Further downstream, the spikes generated in the hidden layer sum at the output unit ($U_{\text{out}}$). Finally, the error signal $e$ (shown in green) is computed from the output spike train. It modulates learning of the output weights and is propagated back to the hidden layer units through feedback weights. Note that the error signal $e$ is strictly causal. The product of presynaptic activity ($x$) with the nonlinear function $\sigma'(U)$ is further filtered in time by $\alpha$ giving rise to the synaptic eligibility trace $\lambda$. In a biological scenario $\lambda$ could for instance be manifested as a calcium transient at the synaptic spine. Finally, temporal coincidence between $\lambda$ and the error signal $e$ determines the sign and magnitude of the plastic weight changes $dw$.

through time through the PSP kernel or the computation of all relevant quantities online as in the case of real-time recurrent learning (RTRL). Here we explore the latter approach since our specific choice of temporal kernels allows us to compute all relevant dynamic quantities and error signals online (Fig. 2b). In our approach, error signals are distributed directly through a feedback matrix to the hidden layer units (Fig. 2a). Specifically, this means that the output error signals are neither propagated through the actual nor the "soft" spiking non-linearity. This idea is closely related to the notion of straight-through estimators in machine learning [14, 52, 55]. We investigated different configurations of the feedback matrix, which can be either (i) symmetric (i.e. the transpose of the feedforward weights), as in the case of backprop, (ii) random as motivated by the recent results on feedback alignment [13] or (iii) uniform, corresponding closest to a single global third factor distributed to all neurons, akin to a diffuse neuromodulatory signal. We first sought to replicate the task shown in Figure 1, but with the addition of a hidden layer composed of 4 LIF neurons. Initially, we tested learning with random feedback. To that end, feedback weights were drawn from a zero mean unit variance Gaussian and their value remained fixed during the entire simulation. The synaptic feedforward weights were also initialized randomly at a level at which neither the hidden units nor the output unit fired a single spike in response to the same input spike trains as used before (Fig. 3a). After training the network for 40s, some of the hidden units had started to fire spikes in response to the input. Similarly, the output neuron had started to fire at intermittent intervals closely resembling the target spike train (not shown). Continued training on the same task for a total of 250s lead to a further refinement of the output spike train and more differentiated firing patterns in a subset of the hidden units (Fig. 3b).

Although, we did not restrict synaptic connectivity to obey Dale's principle, in the present example with random feedback all hidden neurons with positive feedback connections ended up being excitatory, whereas neurons with negative feedback weights generally turned out to be inhibitory at the end of training. These dynamics are a direct manifestation of "feedback alignment" aspect of random feedback learning [13]. Because the example shown in Figure 3 does not strictly require inhibitory neurons in the hidden layer, in most cases the neurons with negative feedback remained quiescent or at low activity levels at the end of learning (Fig. 3b–c).

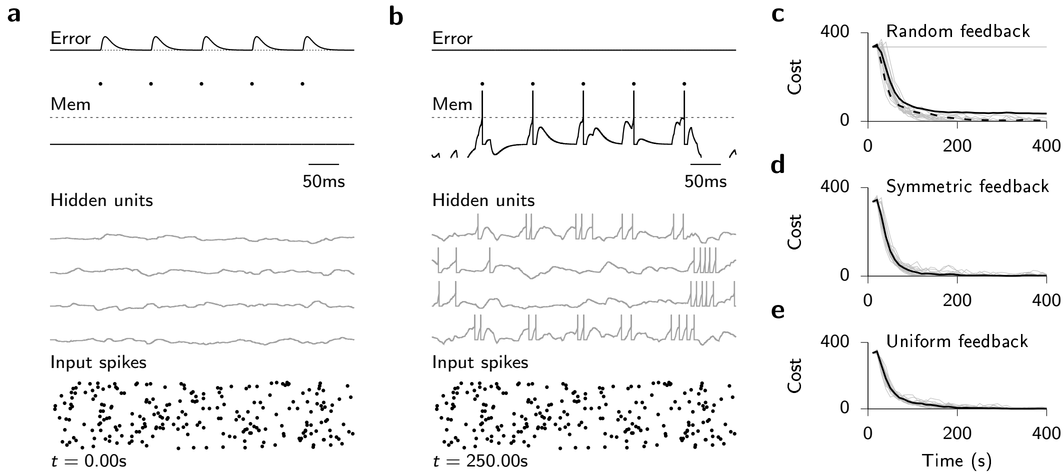


Figure 3: SuperSpike learning with different types of feedback allows to train multi-layer networks. (**a**) Network activity at the initial trial at reference time $t = 0$s. The bottom panel shows the membrane potential traces of the four hidden units. The membrane potential of the output unit is shown in the middle. The dashed line is the output neuron firing threshold. The points correspond to target firing times and the top plot shows the error signal at the output layer. Hidden units receive the same input spikes as shown in Fig. 1a. (**b**) Same as (a), but after 250s of training. The two hidden units which have started to respond to the repeating input spiking pattern are the ones with positive feedback weights, whereas the two hidden neurons which receive negative feedback connections from the output layer (middle traces) respond mostly at the offset of the repeating stimulus. (**c**) Plot of a moving average of the cost during training for a network with random feedback connections. Gray lines correspond to single trials and the black line to the average. The dashed line is the same average but for a network with 8 hidden layer units. (**d**) Same as (d), but for a network with symmetric feedback connections. (**e**) Same as (d–e), but for uniform "all one" feedback connections.

Learning was successful for different initial conditions, although the time for convergence to zero cost varied (Fig. 3d). We did encounter, however, a few cases in which the network completely failed to solve the task. These were the cases in which *all* feedback connections happened to be initialized with a negative value (Fig. 3c). This eventuality could be made very unlikely, however, by increasing in the number of hidden units (Fig. 3c). Other than that, we did not find any striking differences in performance when we replaced the random feedback connections by symmetric (Fig. 3d) or uniform "all one" feedback weights (Fig. 3e).

The previous task was simple enough such that solving it did not require a hidden layer. We therefore investigated whether SuperSpike could also learn to solve tasks that cannot be solved by a network without hidden units. To that end, we constructed a spiking *exclusive-or* task in which four different spiking input patterns had to be separated into two classes. In this example we used 100 input units although the effective dimension of the problem was two by construction. Specifically, we picked three non-overlapping sets of input neurons with associated fixed random firing times in a 10ms window. One set was part of all patterns and served as a time reference. The other two sets were combined to yield the four input patterns of the problem. Moreover, we added a second readout neuron each corresponding to one of the respective target classes (Fig. 4a). The input patterns were given in random order as short bouts of spiking activity at random inter-trial-intervals (Fig. 4b). To allow for the finite propagation time through the network, we relaxed the requirement for precise temporal spiking and instead required output neurons to spike within a narrow window of opportunity which was was aligned with and outlasted each stimulus by 10ms. The output error signal was zero unless the correct output neuron failed to fire within the window. In this case an error signal corresponding to the correct output was elicited at the end of the window. At any time an incorrect spike triggered immediate negative feedback. We trained the network comparing different types of feedback. A network with random feedback quickly learned to solve this task with perfect accuracy (Fig. 4b–c), whereas a network without hidden units was unable to solve the task (Fig. 4d). Perhaps not surprisingly, networks with symmetric feedback connections also learned the task quickly and overall their learning curves were more stereotyped and less noisy (Fig. 4e), whereas networks with uniform feedback performed worse on average (Fig. 4f). Overall these results illustrate that temporally coding spiking multi-layer networks can be trained to solve tasks which cannot be solved by networks without hidden layers. Moreover, these results show that random feedback is beneficial over uniform feedback in some cases.

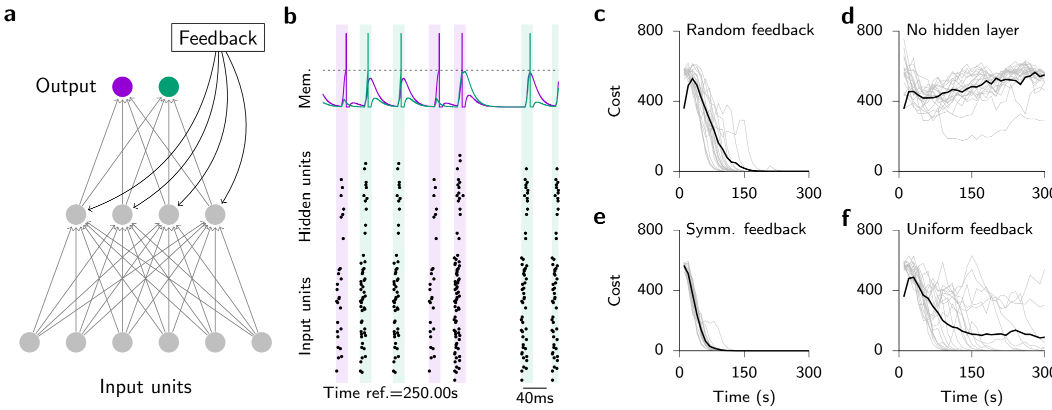


Figure 4: SuperSpike learning of a non-linearly separable problem. (**a**) Sketch of network layout with two output units and four hidden units. (**b**) Snapshot of network activity at the end of training. (**c**) Learning curves of 20 trials with different random initializations (gray) for a network with random Gaussian feedback connections. The average of all trials is given by the black line. (**d**) Same as (d), but for a network without hidden units. (**e**) Same as (d), but for symmetric feedback. (**f**) Same as (d), but for uniform ("all ones") feedback connections.

All tasks considered so far were simple enough that they could be solved by most three layer networks with zero error for all types of feedback signals. We hypothesized that the observed indifference to the type of feedback could be due to the task being too simple. To test whether this picture would change for a more challenging task we studied a network with 100 output neurons which had to learn a 3.5 second-long complex spatiotemporal output pattern from cyclically repeating frozen Poisson noise. Specifically, we trained a three layer SNN with 100 input, 100 output, and different numbers of hidden neurons (Fig. 5a). Within 1000s of training with symmetric feedback connections, a network with 32 or more hidden units could learn to emit an output spike pattern which visually matched to the target firing pattern (Fig. 5b–e). After successful learning, hidden unit activity was irregular and at intermediate firing rates of 10–20Hz with a close to exponential inter-spike-interval distribution (Fig. 5c). However, the target pattern was not learned perfectly as evidenced by a non-vanishing van Rossum cost (Fig. 5d) and a number of spurious spikes (Fig. 5b). For this example a simulation without a hidden layer yielded higher performance than a network with 32 hidden units, but both were outperformed by a network in which the number of hidden units was increased to $n_{\mathrm{h}} = 256$ (Fig. 5d,f). When the same network was trained using random feedback, performance was degraded substantially (Fig. 5d) and the output pattern became close to impossible to recognize visually. The addition of a heterosynaptic weight decay term (Supplementary Methods) to the learning rule improved the visual similarity of the output patterns (Fig. 5g) although this change was hardly reflected in a difference in the cost (Fig. 5d). When we trained the network with a learning rule in which we disabled the nonlinear voltage dependence of the learning rule by setting the corresponding term to 1, the output pattern was similarly degraded and we found high levels of synchrony (Fig. 5d,h). These results seem to confirm our intuition, that for more challenging tasks both the nonlinearity of the learning rule and non-random feedback seem to become more important to achieving good performance on the type of spatiotemporal spike pattern transformation tasks we considered here.

# 4 Discussion

In this paper we have derived a three factor learning rule to train deterministic multi-layer SNNs of LIF neurons. Moreover, we have assessed the impact of different types of feedback credit assignment strategies for the hidden units, notably symmetric, random, and uniform. In contrast to previous work [24, 48, 49], we have used a deterministic surrogate gradient approach instead of the commonly used

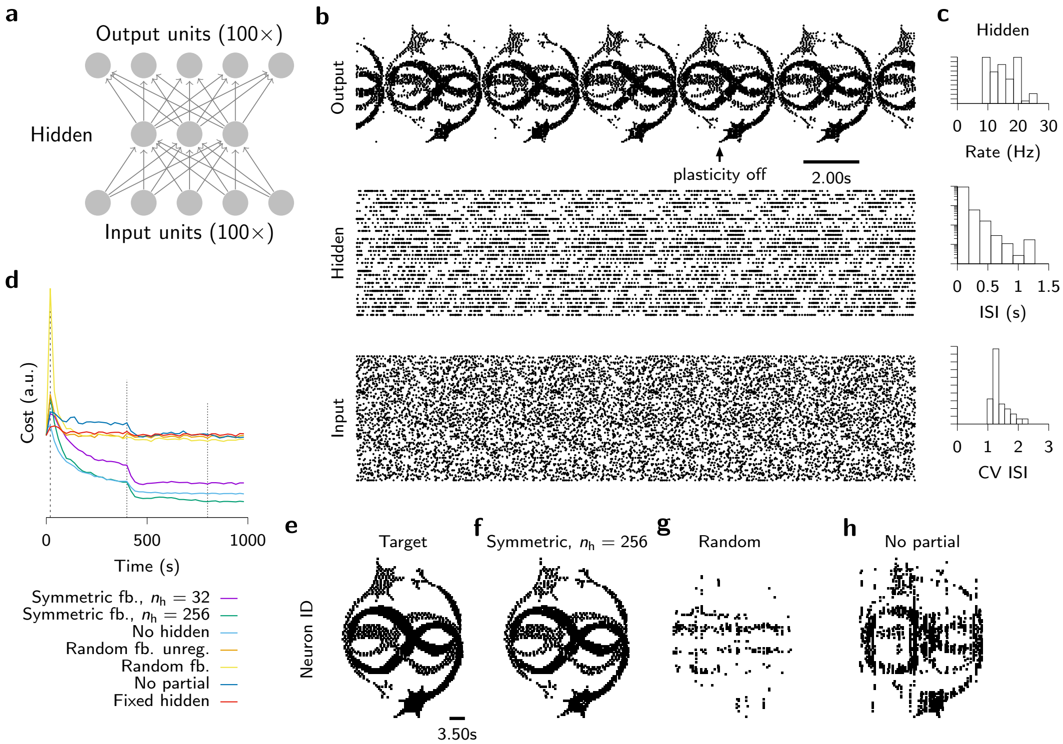


Figure 5: Learning spatiotemporal spike patterns. (**a**) Spike raster of target firing pattern of 100 output neurons. The whole firing pattern has a duration of 3.5s. (**b**) Schematic illustration of the network architecture. (**b**) Snapshot of network activity of the network with symmetric feedback after 1000s of SuperSpike learning. Bottom panel: Spike raster of repeating frozen Poisson input spikes. Middle panel: Spike raster of hidden unit spiking activity. Top panel: Spike raster of output spiking activity. The black arrow denotes the point in time at which SuperSpike learning is switched off which freezes the spiking activity of the fully deterministic network. (**c**) Histograms of different firing statistics of hidden layer activity at then of learning. Top: Distribution of firing rates. Middle: Inter-spike-interval (ISI) distribution on semi-log axes. Bottom: Distribution of coefficient of variation (CV) of the ISI distribution. (**d**) Learning curves for different feedback strategies. No hidden: Without hidden units. No partial: Without voltage nonlinearity. Learning was activated at the dashed line, whereas the learning rate was reduced by a factor of 10 at each dotted line. (**e**) Spike raster of target firing pattern for reference. (**f**) Spike raster snapshots of output activity for networks trained with symmetric feedback. (**g**) Like (f), but with random feedback. (**h**) Like (f), but without voltage nonlinearity.

stochastic gradient approximations. By combining this rule with ideas of straight-through estimators [52, 55] and feedback alignment [13, 14], we could efficiently train and study precisely timed spiking dynamics in multi-layer networks of deterministic LIF neurons without relying on the introduction of extraneous and unavoidable noise present in stochastic models, noise which generally impedes the ability to learn precise spatiotemporal spike-pattern transformations.

The weight update equations of SuperSpike constitutes a voltage-based nonlinear Hebbian three factor rule with individual synaptic eligibility traces. These aspects each have direct biological interpretations. For instance, a nonlinear voltage dependence has been reported ubiquitously by numerous studies on Hebbian long-term plasticity induction in hippocampus and cortex [56, 57]. Also, the window of temporal coincidence detection in our model is in good agreement with that of STDP [57]. Moreover, the time course of the eligibility traces could be interpreted as a local calcium transient at the synaptic spine level. Finally, the multiplicative coupling of the error signal with the eligibility trace could arise from neuromodulators [53, 54, 58]. However, instead of only one global feedback signal, our work highlights the necessity of a higher dimensional neuromodulatory or electrical feedback signal for learning potentially with some knowledge of the feedforward pathway.

The biological exploration of such intelligent neuromodulation, as well as extensions of our approach to deeper and recurrent SNNs, are left as intriguing directions for future work.

## 5 Acknowledgements

The authors would like to thank Subhy Lahiri and Ben Poole for helpful discussions. FZ was supported by the SNSF (Swiss National Science Foundation) and the Wellcome Trust. SG was supported by the Burroughs Wellcome, Sloan, McKnight, Simons and James S. McDonnell foundations and the Office of Naval Research.

# 6 Supplementary Methods

We trained networks of current-based LIF neurons using a supervised learning approach which we call "SuperSpike". This approach generalizes the back propagation of error algorithm as known from the multi-layer perceptron to deterministic spiking neurons. Because the partial derivative and thus the gradient of deterministic spiking neurons is zero almost everywhere, to make this optimization problem solvable, we introduce a non-vanishing surrogate gradient [52, 55] (cf. Eq. 5).

## 6.1 Neuron model

We use LIF neurons with current-based synaptic input because they can be alternatively formulated via their integral form (cf. Eq. 4). However, to simulate the membrane dynamics we computed the voltage $U_i$ of neuron $i$ as described by the following differential equation

$$\tau^{\mathrm{mem}} \frac{dU_i}{dt} = (U^{\mathrm{rest}} - U_i) + I_i^{\mathrm{syn}}(t) \tag{6}$$

in which the synaptic input current $I_i^{\mathrm{syn}}(t)$ evolves according to

$$\frac{d}{dt} I_i^{\mathrm{syn}}(t) \quad = \quad -\frac{I_i^{\mathrm{syn}}(t)}{\tau^{\mathrm{syn}}} + \sum_{j \in \mathrm{pre}} w_{ij} S_j(t) \quad . \tag{7}$$

The value of $I_i^{\mathrm{syn}}(t)$ jumps by an amount $w_{ij}$ at the moment of spike arrival from presynaptic neurons $S_j(t) = \sum_k \delta(t - t_j^k)$ where $\delta$ denotes the Dirac $\delta$-function and $t_j^k$ $(k = 1, 2, \cdots)$ are firing times of neuron $j$. An action potential is triggered when the membrane voltage of neuron $i$ rises above the threshold value $\vartheta$ (see Table 1 for parameters). Following a spike the voltage $U_i$ remains clamped at $U_i^{\mathrm{rest}}$ for 5ms to emulate a refractory period.

| Parameter | Value |
|---|---|
| $\vartheta$ | -50mV |
| $U^{\mathrm{rest}}$ | -60mV |
| $\tau^{\mathrm{mem}}$ | 10ms |
| $\tau^{\mathrm{syn}}$ | 5ms |

Table 1: Neuron model parameters.

## 6.2 Stimulation paradigms

Depending on the task at hand, we used two different types of stimuli. For simulation experiments in which the network had to learn exact output spike times, we used a set of frozen Poisson spike trains as input. These stimuli consisted of a single draw of $n$, where $n$ is the number of input units, Poisson spike trains of a given duration. These spike trains were then repeated in a loop and had to be associated with the target spike train which was consistently aligned to the repeats of the frozen Poisson inputs. For benchmarking and comparison reasons, the stimulus and target spike trains shown in this paper will be made available on the authors GitHub page.

For classification experiments we used sets of different stimuli. Individual stimuli were drawn as random neuronal firing time offsets from a common stimulus onset time. Stimulus order was chosen randomly and with randomly varying inter-stimulus-intervals.

## 6.3 Plasticity model

The main ingredients for our supervised learning rule for spiking neurons (SuperSpike) are summarized in Equation 5 describing the synaptic weight changes. As already eluded to in the main text, the learning rule can be interpreted as a nonlinear Hebbian three factor rule. The nonlinear Hebbian term detects coincidences between presynaptic activity and postsynaptic depolarization. These spatiotemporal coincidences at the single synapse $w_{ij}$ are then stored transiently by the temporal convolution with the causal kernel $\alpha$. This step can be interpreted as a synaptic eligibility trace, which in neurobiology could for instance be implemented as a calcium transient or a related signaling cascade (cf. Fig. 2b). In the model, all the complexity of neural feedback of learning is absorbed into the per-neuron signal $e_i(t)$. Because it is unclear if and how such error feedback is signaled to individual neurons in biology here we explored different strategies which are explained in more detail below. For practical reasons, we integrate Eq. 5 over finite temporal intervals before updating the weights. The full learning rule can be written as follows:

$$\Delta w_{ij}^k = r_{ij} \int_{t_k}^{t_{k+1}} \underbrace{e_i(s)}_{\text{Error signal}} \quad \alpha * \left( \underbrace{\sigma'(U_i(s))}_{\text{Post}} \underbrace{(\epsilon * S_j)(s)}_{\text{Pre}} \right) \, ds \tag{8}$$

In addition to the neuronal dynamics as described in the previous section, the evaluation of Eq. 5 can thus coarsely be grouped as follows: i) evaluation of presynaptic traces, ii) evaluation of Hebbian coincidence and computation of synaptic eligibility traces, iii) computation and propagation of error signals, and iv) integration of Eq. 5 and weight update. We will describe each part in more detail in the following.

### 6.3.1 Presynaptic traces

Because $\epsilon$ is a double exponential filter, the temporal convolution in the expression of the presynaptic traces (Eq. 8), can be evaluated efficiently online by exponential filtering twice. Specifically, we explicitly integrate the single exponential trace

$$\frac{dz_j}{dt} = -\frac{z_j}{\tau_{\text{rise}}} + S_j(t)$$

in every time step which is then fed into a second exponential filter array

$$\tau_{\text{decay}} \frac{d\tilde{z}_j}{dt} = -\tilde{z}_j + z_j$$

with $\tilde{z}_j(t) \equiv (\epsilon * S_j)(t)$ which now implements the effective shape of a PSP in the model. In all cases we chose the time constants $\tau_{\text{rise}} = 5\text{ms}$ and $\tau_{\text{decay}} = 10\text{ms}$.

### 6.3.2 Hebbian coincidence detection and synaptic eligibility traces

To evaluate the Hebbian term we evaluate the surrogate partial derivative $\sigma'(U_i)$ in every time step. For efficiency reasons we use the partial derivative of the negative half of a fast sigmoid $f(x) = \frac{x}{1+|x|}$ which does not require the costly evaluation of exponential functions in every. Specifically, we compute $\sigma'(U_i) = (1 + |h_i|)^{-2}$ with $h \equiv \beta (U_i - \vartheta)$ where $\vartheta$ is the neuronal firing threshold and $\beta = 1\text{mV}$ unless mentioned otherwise.

We compute the outer product between the presynaptic traces $\tilde{z}_j$ and the surrogate partial derivatives $\sigma'(U_i)$ in every time step. Because the presynaptic traces decay to zero quickly in the absence of spikes, we approximate them to be exactly zero when their numerical value drops below machine precision of $1 \times 10^{-7}$. This allows us to speed up the computation of the outer product by skipping these presynaptic indices in the computation.

To implement the synaptic eligibility trace as given by the temporal filter $\alpha$, we filter the values of Hebbian product term with two exponential filters just like in the case of the presynaptic traces $z_j$ above. It is important to note, however, that these traces now need to be computed for each synapse $w_{ij}$ which makes the algorithm scale as $O(n^2)$ for $n$ being the number of neurons. This makes it the most obvious target for future optimizations of our algorithm. Biologically, this complexity could be implemented naturally simply because synaptic spines are electrical and ionic compartments in which a concentration transient of calcium or other messengers decays on short timescales. For SuperSpike to work, it is important that these transients are long enough to temporally overlap with any causally related error signal $e_i(t)$. Formally the duration of the transient in the model is given by the filter kernel shape used to compute the van Rossum distance. We used a double-exponentially filtered kernel which has the same shape as a PSP in the model, but other kernels are possible.

### 6.3.3 Error signals

We distinguish two types of error signals: Output error signals and feedback (error) signals. Output error signals are directly tied to output units for which a certain target signal exists. Their details depend on the underlying cost function we are trying to optimize. Feedback signals, on the other hand, are derived from output error signals by sending them back to the hidden units. In this study we used two slightly different classes of output error signals and three different types of feedback.

At the level of output errors we distinguish between the cases in which our aim was to learn precisely timed output spikes. In these cases the output error signals were exactly given by $e_i = \alpha * (\hat{S}_i - S_i)$ for an output unit $i$. As can be seen from this expression, the error signal $e_i$ only vanishes if the target and the output spike train exactly match with the temporal precision of our simulation. In simulations in which we wanted to classify input spike patterns rather than generate precisely timed output patterns, we introduced some slack into the computation of the error signal. For instance, as illustrated in Figure 4, we gave instantaneous negative error feedback as described by $e_i = -\alpha * S_i^{\mathrm{err}}$ for each erroneous additional spike $S_i^{\mathrm{err}}$. However, since for this task we did not want the network to learn precisely timed output spikes, we only gave a positive feedback signal $e_i = \alpha * S_i^{\mathrm{miss}}$ at the end of a miss trial, i.e. when a stimulus failed to evoke an output spike during the window of opportunity when it should have (see section on Stimuli above).

### 6.3.4 Weight updates

To update the weights, the time continuous time series corresponding to the product of error/feedback signal and the synaptic eligibility traces $\lambda_{ij}$ were not directly added to the synaptic weights, but first integrated in a separate variable $m_{ij}$ in chunks of $t_b = 0.5257\mathrm{s}$. Specifically, we computed $m_{ij} \to m_{ij} + g_{ij}$ with $g_{ij}(t) = e_i(t)\,\lambda_{ij}(t)$ at each time step. For stimuli exceeding the duration $t_b$ this can thus be seen as the continuous time analogue to mini batch optimization. We chose $t_b$ on the order of half a second as a good compromise between computational cost and performance for synaptic updates and added 257 simulation time steps to minimize periodic alignment of the update step with the stimulus. At the end of each interval $t_b$, all weights were updated according to $w_{ij} \to w_{ij} + r_{ij} m_{ij}$ with the per parameter learning rate $r_{ij}$. In addition to that, we enforced the constraint for individual weights to remain in the interval $-0.1 < w_{ij} < 0.1$. After updating the weights, the variables $m_{ij}$ were reset to zero.

### 6.3.5 Per-parameter learning rates

To facilitate finding the right learning rate and the speed up training times in our simulations, we implement a per-parameter learning rate heuristic. To compute the per-parameter learning rate, in addition to $m_{ij}$ we integrated another auxiliary quantity $v_{ij} \to \max(\gamma v_{ij}, g_{ij}^2)$. Here $\gamma = \exp(-\Delta/\tau_{\mathrm{rms}})$ ensures a slow decay of $v_{ij}$ for $g_{ij} = 0$. Consequently, $v_{ij}$ represents an upper estimate of the variance of the surrogate gradient for each parameter on the characteristic timescale $\tau_{\mathrm{rms}}$. With these defintions, the per-parameter learning rate was defined as $r_{ij} \equiv \frac{r_0}{\sqrt{v_{ij}}}$. This choice is motivated by the RMSProp optimizer which is commonly used in the field of deep learning [52]. However, RMSProp computes a moving exponential average over the $g_{ij}^2$. We found that introducing the max function rendered training more stable while simultaneously yielding excellent convergence times. We call this slightly modified version "RMaxProp" (compare also AdaMax [59]).

### 6.3.6 Regularization term

In some experiments with random feedback we added a heterosynaptic regularization term to the learning rule of the hidden layer weights to avoid pathologically high firing rates. In these experiments the full learning rule was

$$\frac{\partial w_{ij}^{\mathrm{hid}}}{\partial t} = r_{ij} \int_{t_k}^{t_{k+1}} \underbrace{e_i(s)}_{\text{Error signal}} \left( \alpha * \left( \underbrace{\sigma'(U_i(s))}_{\text{Post}} \underbrace{(\epsilon * S_j)(s)}_{\text{Pre}} \right) - \underbrace{\rho w_{ij} e_i(s) z_i^4}_{\text{Regularizer}} \right) ds \qquad (9)$$

where we introduced the regularization strength parameter $\rho$. We made the regularization term explicitly dependent on the square of the error signal to ensure it would be zero in cases where the task was solved perfectly. Moreover, we used the fourth power of an exponential synaptic trace $z_i$ which evolved according to the following differential equation

$$\frac{dz_i}{dt} = -\frac{z_i}{\tau_{\text{het}}} + S_i(t)$$

the weight and forth power rate-dependence was motivated from previous work [60] and to regularize high firing rates more strongly.

## References


[1] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85–117, January 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836. doi: 10.1038/nature14539.

[3] Daniel L. K. Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *PNAS*, 111(23):8619–8624, June 2014. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1403112111.

[4] Patrick McClure and Nikolaus Kriegeskorte. Representational Distance Learning for Deep Neural Networks. *Front. Comput. Neurosci.*, 10, 2016. ISSN 1662-5188. doi: 10.3389/fncom. 2016.00131.

[5] Lane McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen Baccus. Deep Learning Models of the Retinal Response to Natural Scenes. *NIPS*, pages 1369–1377, 2016.

[6] Adam H. Marblestone, Greg Wayne, and Konrad P. Kording. Toward an Integration of Deep Learning and Neuroscience. *Front. Comput. Neurosci*, page 94, 2016. doi: 10.3389/fncom. 2016.00094.

[7] Peter U. Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci*, page 99, 2015. doi: 10.3389/fncom.2015. 00099.

[8] Jordan Guergiuev, Timothy P. Lillicrap, and Blake A. Richards. Biologically feasible deep learning with segregated dendrites. *arXiv:1610.00161 [q-bio]*, October 2016. arXiv: 1610.00161.

[9] Emre Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Neuromorphic Deep Learning Machines. *arXiv:1612.05596 [cs]*, December 2016. arXiv: 1612.05596.

[10] Mihai A. Petrovici, Sebastian Schmitt, Johann Klähn, David Stöckel, Anna Schroeder, Guillaume Bellec, Johannes Bill, Oliver Breitwieser, Ilja Bytschok, Andreas Grübl, Maurice Güttler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, Vitali Karasenko, Mitja Kleider, Christoph Koke, Alexander Kononov, Christian Mauch, Paul Müller, Johannes Partzsch, Thomas Pfeil, Stefan Schiefer, Stefan Scholze, Anand Subramoney, Vasilis Thanasoulis, Bernhard Vogginger, Robert Legenstein, Wolfgang Maass, René Schüffny, Christian Mayr, Johannes Schemmel, and Karlheinz Meier. Pattern representation and recognition with accelerated analog neuromorphic systems. *arXiv:1703.06043 [cs, q-bio, stat]*, March 2017. arXiv: 1703.06043.

[11] Stephen Grossberg. Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, 11(1):23–63, January 1987. ISSN 1551-6709. doi: 10.1111/j.1551-6708. 1987.tb00862.x.

[12] Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, January 1989. ISSN 0028-0836. doi: 10.1038/337129a0.

[13] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, November 2016. ISSN 2041-1723. doi: 10.1038/ncomms13276.

[14] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the Machine: Random Backpropagation and the Learning Channel. *arXiv:1612.02734 [cs]*, December 2016. arXiv: 1612.02734.

[15] Zhibin Liao and Gustavo Carneiro. On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units. *arXiv:1508.00330 [cs]*, August 2015. arXiv: 1508.00330.

[16] Chris Eliasmith, Terrence C. Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A Large-Scale Model of the Functioning Brain. *Science*, 338 (6111):1202–1205, November 2012. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1225266.

[17] Thomas Mesnard, Wulfram Gerstner, and Johanni Brea. Towards deep learning with spiking neurons in energy based models with contrastive Hebbian plasticity. *arXiv preprint arXiv:1612.03214*, 2016.

[18] Dominik Thalmeier, Marvin Uhlmann, Hilbert J. Kappen, and Raoul-Martin Memmesheimer. Learning universal computations with spikes. *arXiv:1505.07866 [q-bio]*, May 2015. arXiv: 1505.07866.

[19] Sophie Denève and Christian K. Machens. Efficient codes and balanced networks. *Nat Neurosci*, 19(3):375–382, March 2016. ISSN 1097-6256. doi: 10.1038/nn.4243.

[20] L. F. Abbott, Brian DePasquale, and Raoul-Martin Memmesheimer. Building functional networks of spiking model neurons. *Nat Neurosci*, 19(3):350–355, March 2016. ISSN 1097-6256. doi: 10.1038/nn.4241.

[21] Wieland Brendel, Ralph Bourdoukan, Pietro Vertechi, Christian K. Machens, and Sophie Denéve. Learning to represent signals spike by spike. *arXiv:1703.03777 [q-bio]*, March 2017. arXiv: 1703.03777.

[22] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, June 1996. doi: 10.1038/381520a0.

[23] Tim Gollisch and Markus Meister. Rapid Neural Coding in the Retina with Relative Spike Latencies. *Science*, 319(5866):1108–1111, February 2008. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1149639.

[24] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal Spike-Timing-Dependent Plasticity for Precise Action Potential Firing in Supervised Learning. *Neural Computation*, 18(6):1318–1348, April 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.6.1318.

[25] Sander M. Bohte and Michael C Mozer. Reducing Spike Train Variability: A Computational Theory Of Spike-Timing Dependent Plasticity. *NIPS*, pages 201–208, 2005.

[26] Taro Toyoizumi, Jean-pascal Pfister, Kazuyuki Aihara, and Wulfram Gerstner. Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model. *Adv Neural Inf Process Syst*, pages 1409–1416, 2005.

[27] Filip Ponulak and Andrzej Kasiński. Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Computation*, 22(2): 467–510, October 2009. ISSN 0899-7667. doi: 10.1162/neco.2009.11-08-901.

[28] Ammar Mohemmed, Stefan Schliebs, Satoshi Matsuda, and Nikola Kasabov. Span: spike pattern association neuron for learning spatio-temporal spike patterns. *Int. J. Neur. Syst.*, 22 (04):1250012, June 2012. ISSN 0129-0657. doi: 10.1142/S0129065712500128.

[29] Răzvan V. Florian. The Chronotron: A Neuron That Learns to Fire Temporally Precise Spike Patterns. *PLOS ONE*, 7(8):e40233, August 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0040233.

[30] J. D. Victor and K. P. Purpura. Metric-space analysis of spike trains: theory, algorithms and application. *Network: Computation in Neural Systems*, 8:127–164, 1997.

[31] Brian Gardner and André Grüning. Supervised Learning in Spiking Neural Networks for Precise Temporal Encoding. *PLOS ONE*, 11(8):e0161335, August 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0161335.

[32] Christian Albers, Maren Westkott, and Klaus Pawelzik. Learning of Precise Spike Times with Homeostatic Membrane Potential Dependent Synaptic Plasticity. *PLOS ONE*, 11(2):e0148948, February 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0148948.

[33] Raoul-Martin Memmesheimer, Ran Rubin, Bence P. Ölveczky, and Haim Sompolinsky. Learning Precisely Timed Spikes. *Neuron*, 82(4):925–938, May 2014. ISSN 0896-6273. doi: 10.1016/j.neuron.2014.03.026.

[34] Johanni Brea, Walter Senn, and Jean-Pascal Pfister. Matching Recall and Storage in Sequence Learning with Spiking Neural Networks. *J. Neurosci.*, 33(23):9565–9575, June 2013. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.4098-12.2013.

[35] Danilo Jimenez Rezende and Wulfram Gerstner. Stochastic variational learning in recurrent spiking networks. *Front. Comput. Neurosci*, 8:38, 2014. doi: 10.3389/fncom.2014.00038.

[36] Aditya Gilra and Wulfram Gerstner. Predicting non-linear dynamics: a stable local learning scheme for recurrent spiking neural networks. *arXiv:1702.06463 [cs, q-bio]*, February 2017. arXiv: 1702.06463.

[37] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, 14(11):2531–2560, 2002. ISSN 0899-7667. doi: 10.1162/089976602760407955.

[38] Peter Auer, Harald Burgsteiner, and Wolfgang Maass. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks*, 21(5):786–795, June 2008. ISSN 0893-6080. doi: 10.1016/j.neunet.2007.12.036.

[39] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9(3):420–428, March 2006. ISSN 1097-6256. doi: 10.1038/nn1643.

[40] Robert Gütig. Spiking neurons can discover predictive features by aggregate-label learning. *Science*, 351(6277):aab4113, March 2016. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aab4113.

[41] Robert Urbanczik and Walter Senn. A Gradient Learning Rule for the Tempotron. *Neural Comput*, 21(2):340–352, 2009. ISSN 0899-7667. doi: 10.1162/neco.2008.09-07-605.

[42] Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1):17–37, 2002.

[43] S. McKennoch, Dingding Liu, and L. G. Bushnell. Fast Modifications of the SpikeProp Algorithm. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3970–3977, 2006. doi: 10.1109/IJCNN.2006.246918.

[44] Sumit Bam Shrestha and Qing Song. Adaptive learning rate of SpikeProp based on weight convergence analysis. *Neural Networks*, 63:185–198, March 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.12.001.

[45] Simon de Montigny and Benoît R. Mâsse. On the Analytical Solution of Firing Time for SpikeProp. *Neural Computation*, pages 1–13, August 2016. ISSN 0899-7667. doi: 10.1162/NECO_a_00884.

[46] Sumit Bam Shrestha and Qing Song. Robust learning in SpikeProp. *Neural Networks*, 86:54–68, February 2017. ISSN 0893-6080. doi: 10.1016/j.neunet.2016.10.011.

[47] Ioana Sporea and André Grüning. Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 25(2):473–509, February 2013. ISSN 0899-7667, 1530-888X. doi: 10.1162/NECO_a_00396. arXiv: 1202.2249.

[48] Brian Gardner, Ioana Sporea, and André Grüning. Learning Spatiotemporally Encoded Pattern Transformations in Structured Spiking Neural Networks. *Neural Comput*, 27(12):2548–2586, October 2015. ISSN 0899-7667.

[49] Nicolas Fremaux, Henning Sprekeler, and Wulfram Gerstner. Functional Requirements for Reward-Modulated Spike-Timing-Dependent Plasticity. *J. Neurosci.*, 30(40):13326–13337, October 2010. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.6249-09.2010.

[50] M. C. W. van Rossum. A Novel Spike Distance. *Neural Computation*, 13(4):751–763, April 2001. ISSN 0899-7667. doi: 10.1162/089976601300014321.

[51] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: from single neurons to networks and models of cognition*. Cambridge University Press, Cambridge, 2014. ISBN 978-1-107-06083-8.

[52] G. Hinton. Neural networks for machine learning. Coursera, video lectures., 2012.

[53] Eugene M. Izhikevich. Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling. *Cereb Cortex*, 17(10):2443 –2452, October 2007. doi: 10.1093/cercor/bhl152.

[54] Nicolas Frémaux and Wulfram Gerstner. Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules. *Front Neural Circuits*, page 85, 2016. doi: 10.3389/fncir.2015.00085.

[55] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432 [cs]*, August 2013. arXiv: 1308.3432.

[56] A. Artola, S. Bröcher, and W. Singer. Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature*, 347 (6288):69–72, September 1990. doi: 10.1038/347069a0.

[57] Daniel E. Feldman. The Spike-Timing Dependence of Plasticity. *Neuron*, 75(4):556–571, August 2012. ISSN 0896-6273. doi: 10.1016/j.neuron.2012.08.001.

[58] Verena Pawlak, Jeffery R Wickens, Alfredo Kirkwood, and Jason N D Kerr. Timing is not Everything: Neuromodulation Opens the STDP Gate. *Front Synaptic Neurosci*, 2:146, 2010. ISSN 1663-3563. doi: 10.3389/fnsyn.2010.00146.

[59] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

[60] Friedemann Zenke, Everton J. Agnes, and Wulfram Gerstner. Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nat Commun*, 6, April 2015. doi: doi:10.1038/ncomms7922.