# BP-STDP: Approximating Backpropagation using Spike Timing Dependent Plasticity

**Amirhossein Tavanaei**[1,*] **and Anthony S. Maida**[1]

[1]University of Louisiana at Lafayette, The Center for Advanced Computer Studies, Lafayette, LA 70504, USA
[*]tavanaei@louisiana.edu



## ABSTRACT

The problem of training spiking neural networks (SNNs) is a necessary precondition to understanding computations within the brain, a field still in its infancy. Previous work has shown that supervised learning in multi-layer SNNs enables bio-inspired networks to recognize patterns of stimuli through hierarchical feature acquisition. Although gradient descent has shown impressive performance in multi-layer (and deep) SNNs, it is generally not considered biologically plausible and is also computationally expensive. This paper proposes a novel supervised learning approach based on an event-based spike-timing-dependent plasticity (STDP) rule embedded in a network of integrate-and-fire (IF) neurons. The proposed temporally local learning rule follows the backpropagation weight change updates applied at each time step. This approach enjoys benefits of both accurate gradient descent and temporally local, efficient STDP. Thus, this method is able to address some open questions regarding accurate and efficient computations that occur in the brain. The experimental results on the XOR problem, the Iris data, and the MNIST dataset demonstrate that the proposed SNN performs as successfully as the traditional NNs. Our approach also compares favorably with the state-of-the-art multi-layer SNNs.

## 1 Introduction

In the vein of neural network research, spiking neural networks (SNNs) have attracted recent and long-standing interest due to their biologically plausible realism, theoretical computation power, and power efficiency[1–5]. Neurons in SNNs (analogous to biological neurons in the brain) communicate via discrete spike events. An important question in implementing spiking frameworks is how these networks are trained under supervision while there is no differentiable activation function of continuous values? Multi-layer supervised learning is a crucial concept in SNNs to prove their ability to predict and classify the patterns of stimuli.

The earliest supervised learning in SNNs, SpikeProp, was proposed by Bohte et al. (2002)[6] which develops a gradient descent based learning method similar to traditional backpropagation embedded in multi layer neural networks. SpikeProp minimizes the distance between single desired and output spikes by formulating the spike time as a function of the neuron's membrane potential. Later, Quick-Prop and RProp[7] were introduced as faster versions of SpikeProp. Following the same approach, Booij et al. (2005)[8] and Ghosh et al. (2009)[9] proposed new SpikeProp algorithms minimizing the distance between multiple desired and output spikes to improve the model's coding ability in response to temporal patterns. Gradient descent (GD), as a popular supervised learning approach, is still being employed to develop high performance supervised learnings in spiking platforms either in offline[10] or online[11] manner. Chronotron[12] which utilizes the Victor & Purpura (VP) metric[13] for E-learning (offline) and the synaptic current simulation for I-learning (online) is another example of GD in SNNs. As implemented in Chronotron, the error function in GD can be defined by a difference measure between the post- and presynaptic current (or membrane potential)[14] (online) or a spike train distance metric (offline) such as von Rossum distance[15,16] and inner product of spike trains[17]. The online gradient descent method has also attracted recent interest in deep SNNs[18,19]. Although the online GD methods have been successful in developing supervised learning in multi-layer SNNs, using membrane potential and derivative based approaches are biologically implausible because SNNs only communicate via discrete spike events. Additionally, GD is computationally expensive for multi-layer SNNs. Recently, Xie et al. (2016)[20] developed a normalized spiking backpropagation calculating postsynaptic spike times (which still needs expensive

computations) instead of membrane potential at each time step to improve the algorithm's efficiency.

Another vein of research utilizes modified versions of the Widrow-Hoff learning rule for SNNs. For instance, ReSuMe[21] introduced a single-layer supervised learning approach; and later, it was extended to the multi-layer framework using backpropagation by Sporea et al. (2013)[22]. SPAN[23,24] also used Widrow-Hoff by transforming the discrete spikes to corresponding analog signals. The learning methods mentioned above are more efficient but less accurate than the GD-based approaches. Another efficient, supervised learning method belongs to the perceptron-based approaches such as Tempotron[25] where each spike event is treated as a binary tag for training the perceptron[26,27]. These models present single layer supervised learning. However, the idea of spike/no-spike classification broadens a new supervised learning category incorporating efficient spike-timing-dependent plasticity (STDP)[28–30] and anti-STDP that are triggered according to the neuron's label[31,32]. STDP is a biologically plausible learning rule occurs in the brain in which the presynaptic spikes occur immediately before the current postsynaptic spike strengthen the interconnecting synapses (LTP); otherwise, the synapses are weakened (LTD).

In this paper, we propose novel multi-layer, supervised learning rules to train SNNs of integrate-and-fire (IF) neurons. The proposed approach takes advantages of the both efficient, bio-inspired STDP and high performance backpropagation (gradient descent) rules. First, we show that the IF neurons approximate the rectified linear units. Then, we develop a temporally local learning approach specified by an STDP/anti-STDP rule derived from the backpropagation weight change rules that can be applied at each time step.

## 2 Method

Before proposing the spike-based, temporally local learning rules, we show how biological IF neurons approximate the well-known artificial neurons equipped by rectified linear unit activation function. This approximation takes the first step in converting the rate-based learning rules to temporal learning rules.

### 2.1 Rectified Linear Unit versus IF neuron

A neuron with the rectified linear unit (ReLU) activation function, $f(y)$, receiving input signals, $x_h$, via corresponding synaptic weights, $w_h$, is defined as follows

$$f(y) = \max(0, y)\ , \quad y = \sum_h x_h w_h \tag{1}$$

$$\frac{\partial f}{\partial y} = \begin{cases} 1, & y > 0 \\ 0, & y \leq 0 \end{cases} \tag{2}$$

**Theorem:** The IF neuron approximates the ReLU neuron. Specifically, the membrane potential of the IF neuron approximates the activation value of the ReLU neuron.

**Proof:** A non-leaky IF neuron integrates the temporal membrane potentials caused by its input spike trains and fires when its membrane potential, $U(t)$, reaches the neuron's threshold, $\theta$. A simplified formulation of the IF neuron is shown in Eq. 3.

$$U(t) = U(t - \Delta t) + \sum_h w_h(t) s_h(t) \tag{3a}$$

$$\text{if } \ U(t) \geq \theta \ \text{ then } \ r(t) = 1,\ U(t) = U_{\text{rest}} \tag{3b}$$

Where, $s_h(t)$ and $r(t)$ are pre- and postsynaptic spikes at time $t$ ($s_h(t), r(t) \in \{0, 1\}$), respectively. The neuron's membrane potential is reset to resting potential, $U_{\text{rest}}$, (assume $U_{\text{rest}} = 0$) upon firing. By formulating the presynaptic spike train, $G_h(t)$, as the sum of delta Dirac functions (Eq. 4), the input value, $x_h \in [0, 1]$, in Eq. 1, can be determined by integrating over the spiking time interval, $T$, as shown in Eq. 5.

$$G_h(t) = \sum_{t_h^p \in \{s_h(t)=1\}} \delta(t - t_h^p) \tag{4}$$

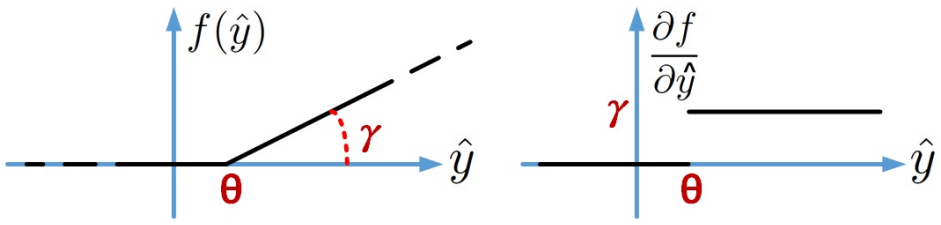


**Figure 1.** IF activation function ($f(\hat{y})$) and its derivative. This function resembles a scaled version of the ReLU function that is shifted by $\theta$.

$$x_h = \frac{1}{K}\int_0^T G_h(t')dt' \tag{5}$$

$K$ is a constant denoting the maximum number of spikes in $T$ ms interval.

In a short time period $(t-\alpha, t]$ between consecutive postsynaptic spikes $r(t-\alpha)$ and $r(t)$, the neuron's membrane potential, following Eq. 3, is obtained by

$$U(t) = \sum_h w_h \Big( \int_{t-\alpha^+}^{t} \sum_{t_h^p} \delta(t' - t_h^p) dt' \Big) \tag{6}$$

The membrane potential calculated above ($U(t)$) is greater than the threshold, $\theta$ ($U(t) \geq \theta$). Thus, as we assumed that the IF neuron is non-leaky and its membrane potential resets to zero upon firing, $R$ postsynaptic spikes demand an accumulated membrane potential (Eq. 7). The accumulated membrane potential, $U^{\text{tot}}$, is obtained by linear summation over the sub-membrane potentials computed in Eq. 6. By this definition, the postsynaptic spike count, $R$, represents the output value, $\hat{y}$, that is passed through a non-linear, threshold-based activation function.

$$U^{\text{tot}} = \hat{y} = \sum_{t^{\text{f}} \in \{r(t)=1\}} U(t^{\text{f}}) \tag{7}$$

$U^{\text{tot}}$ specifies the IF neuron's activity in $T$ ms, which is proportionally related to the postsynaptic spike count, $R$. Therefore, the activation function of the IF neuron can be expressed as

$$f(\hat{y}) = \begin{cases} R = \gamma \hat{y} & \hat{y} > \theta \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where, $\gamma \propto T \cdot \theta^{-1}$ is a constant controlling the postsynaptic spike count. This activation function is similar to a linearly scaled ReLU function that is shifted to right by $\theta$. Figure 1 shows the activation function and its derivative.

This theorem opens a new door to develop new spike-based learning rules (applied to the spiking IF neurons) derived from the traditional learning rules (applied to the ReLU neurons). Specifically, in the next section, we use this theorem to propose an STDP-based backpropagation rule applied to the spiking IF neurons.

### 2.2 Backpropagation using STDP

The proposed learning rules are inspired from the backpropagation update rules reported for neural networks that are equipped by ReLU activation function. Figure 2 shows the network architectures and parameters used to describe the conventional and spiking neural networks in this paper. The main difference between these two networks is their data communication where the neural network (left side network) receives and generates real numbers while the SNN (right side network) receives and generates spike trains in $T$ ms time intervals.

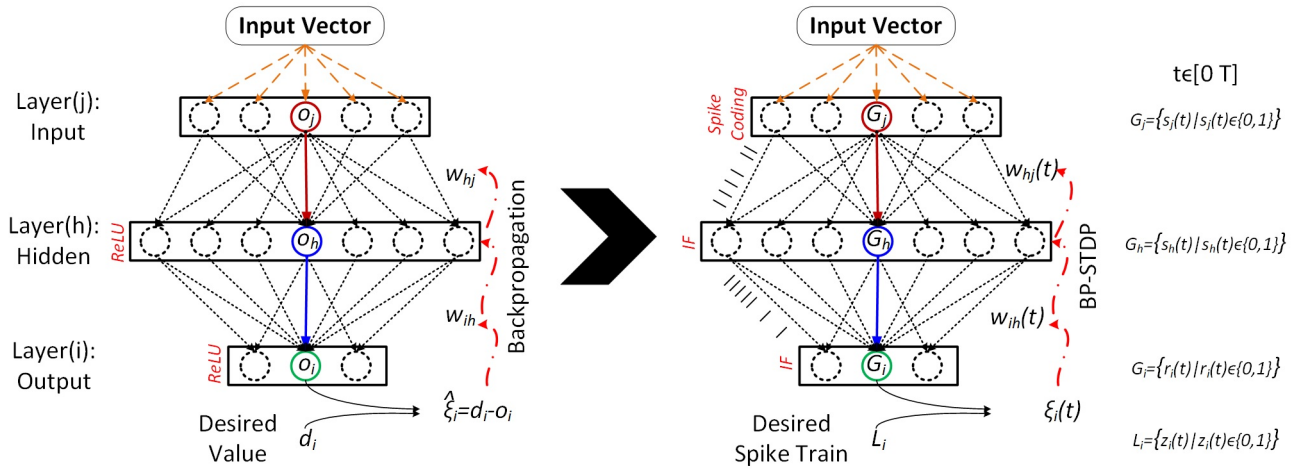


**Figure 2.** The 2-Layer, conventional (left) and spiking (right) network architectures. The SNN receives spike trains representing input feature values in $T$ ms. The learning rules and the network status in the SNN are specified by an additional term as time ($t$). The formulas and parameters are discussed in Eqs. 9 through 21.

Non-spiking neural networks equipped by gradient descent (GD) solve an optimization problem in which the squared difference between the desired, $d$, and output, $o$, values is minimized[33,34]. A common objective function computed for $M$ output neurons receiving $N$ training samples is shown in Eq. 9.

$$E = \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{M}(d_{k,i}-o_{k,i})^2 \tag{9}$$

The weight change formula (using GD with learning rate of $\mu$) for a linear output neuron, say $i$, receiving $H$ inputs, $o_h$, (for a single training sample) is achieved by

$$E=(d_i-o_i)^2=\big(d_i-\sum_h o_h w_{ih}\big)^2 \rightarrow \frac{\partial E}{\partial w_{ih}}=-2(d_i-o_i)\cdot o_h \tag{10}$$

By reversing the sign on the derivative, we have

$$\Delta w_{ih}\propto -\frac{\partial E}{\partial w_{ih}} \rightarrow \Delta w_{ih}=\mu(d_i-o_i)o_h \tag{11}$$

By assuming $d_i, o_i$, and $o_h$ as the spike counts of spike trains $L_i, G_i$, and $G_h$[35] (see Eqs. 4 and 12), respectively, the weight change, defined above, can be re-expressed such that it computes the synaptic weight update in an SNN. This assumption is valid upon approximating the spiking IF neurons to the ReLU neurons (Theorem 1). Eq. 13 shows this update rule after $T=50$ ms.

$$G_i(t)=\sum_{t_i^p\in\{r_i(t)=1\}}\delta(t-t_i^p) \tag{12a}$$

$$L_i(t)=\sum_{t_i^q\in\{z_i(t)=1\}}\delta(t-t_i^q) \tag{12b}$$

$$\Delta w_{ih}=\mu\int_0^T\big(L_i(t')-G_i(t')\big)dt'\cdot\int_0^T G_h(t')dt' \tag{13}$$

However, the weight change rule in Eq. 13 is not local in time. To make the learning rule local in time, we break the time interval, $T$, into sub-intervals such that each sub-interval contains zero or one spike. Hence, the learning

rule, in a short time period of Eq. 13, is specified by Eq. 14.

$$\Delta w_{ih}(t) \propto \mu\big(z_i(t) - r_i(t)\big)s_h(t) \tag{14}$$

To implement the formula above, a combination of event-based STDP and anti-STDP can be used. The proposed learning rule updates the synaptic weights using a teacher signal to switch between STDP and anti-STDP. That is, the target neuron undergoes STDP and the non-target ones undergo anti-STDP. The desired spike trains, $\mathbf{z}$, are defined based on the input's label. Therefore, the target neuron is represented by a spike train with maximum spike frequency ($\beta$) and the non-target neurons are silent. Additionally, the learning rule triggers at desired spike times, $z_i(t)$ (the desired spike times are the same for all the target neurons). Eq. 15 shows the weight change that is applied to the output layer of our supervised SNN.

$$\Delta w_{ih}(t) = \mu \cdot \xi_i(t) \sum_{t'=t-\varepsilon}^{t} s_h(t') \tag{15}$$

$$\xi_i(t) = \begin{cases} 1, & z_i(t) = 1,\ r_i \neq 1 \text{ in } [t-\varepsilon, t] \\ -1, & z_i(t) = 0,\ r_i = 1 \text{ in } [t-\varepsilon, t] \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Then, the synaptic weights of output layer are updated by

$$w_{ih}(t) = w_{ih}(t) + \Delta w_{ih}(t) \tag{17}$$

The target neuron is determined by $z_i(t)$ where $z_i(t) = 1$ denotes the target neuron and $z_i(t) = 0$ denotes the non-target neuron. The weight change scenario for the output layer starts with the desired spike times. At desired spike time $t$, the target neuron should fire in a short time interval $[t-\varepsilon, t]$ known as the STDP window. Otherwise, the synaptic weights are increased proportionally by the presynaptic activity (mostly zero or one spike) in the same time interval. The presynaptic activity is denoted by $\sum_{t'=t-\varepsilon}^{t} s_h(t')$ that counts the presynaptic spikes in $[t-\varepsilon, t]$ interval. On the other hand, the non-target neurons upon firing undergo weight depression in the same way. This scenario is inspired from the traditional GD while supporting spatio-temporal, local learning in SNNs.

The learning rule written above works for a single layer SNN trained by supervision. To train a multi-layer SNN, we use the same idea that is inspired from the traditional backpropagation rules. The backpropagation weight change rule applied to a hidden layer of ReLU neurons is shown in Eq. 18.

$$\Delta w_{hj} = \mu \cdot \Big(\sum_i \hat{\xi}_i w_{ih}\Big) \cdot o_j \cdot [o_h > 0] \tag{18}$$

Where, $\hat{\xi}_i$ denotes the difference between the desired and output values ($d_i - o_i$). In our SNN, $\hat{\xi}_i$ is approximated by $\xi_i$ (Eq. 16). The value $[o_h > 0]$ specifies the derivative of ReLU neurons in the hidden layer. Using the approximation of the IF neurons to the ReLU neurons (Eq. 8), similar to the output layer (Eq. 13), the weight change formula can be re-expressed in terms of spike counts in a multi-layer SNN as shown in Eq. 19.

$$\Delta w_{hj} = \mu \int_0^T \Big(\sum_i \xi_i(t') w_{ih}(t')\Big) dt' \cdot \int_0^T \Big(\sum_{t_j^p} \delta(t' - t_j^p)\Big) dt' \cdot \left( \Big[ \int_0^T \sum_{t_h^p} \delta(t' - t_h^p) dt' \Big] > 0 \right) \tag{19}$$

After dividing $T$ into short sub-intervals $[t-\varepsilon, t]$, the temporally local rule for updating the hidden synaptic weights is formulated as follows.

$$\Delta w_{hj}(t) = \begin{cases} \mu \cdot \sum_i \xi_i(t) w_{ih}(t) \cdot \sum_{t'=t-\varepsilon}^{t} s_j(t') & , s_h = 1 \text{ in } [t-\varepsilon, t] \\ 0 & , \text{otherwise} \end{cases} \tag{20}$$

1: for $t = \Delta t : T : \Delta t$ do
        - - First Layer
2:      $U_1[:,t] = \mathbf{s}_j[:,t] * \mathbf{w}_{jh} + U[:,t-\Delta t]$
3:      for $u$ in hidden neurons do
4:          if $U_1[u,t] > \theta_h$ then
5:              $\mathbf{s}_h[u,t] = 1$
6:              $U_1[u,t] = U_{\text{rest}}$
        - - Second Layer
7:      $U_2[:,t] = \mathbf{s}_h[:,t] * \mathbf{w}_{hi} + U_2[:,t-\Delta t]$
8:      for o in output neurons do
9:          if $U_2[o,t] > \theta_o$ then
10:             $\mathbf{r}_i[o,t] = 1$
11:             $U_2[o,t] = U_{\text{rest}}$
        - - Weight Adaptation
12:     if $t \in \mathbf{z}$ then  - - $\mathbf{z} = \{t | z(t) = 1\}$
13:         if $r_i = 0$ in $[t-\varepsilon, t]$ for target neuron:
14:             $\xi_i[target]$= 1
15:         if $r_i = 1$ in $[t-\varepsilon, t]$ for non-target neurons:
16:             $\xi_i[non\ targets]$= [-1]
17:         $derivatives_h = \left[\mathbf{s}_h(t) = 1 \text{ in } [t-\varepsilon, t]\right]$
18:         $\xi_h = \mathbf{w}_{hi} * \xi_i \cdot derivatives_h$
19:         $\mathbf{w}_{hi} += \text{sum}(\mathbf{s}_h(t-\varepsilon \ldots t)) * \xi_i \cdot \mu$
20:         $\mathbf{w}_{jh} += \text{sum}(\mathbf{s}_j(t-\varepsilon \ldots t)) * \xi_h \cdot \mu$

**Figure 3.** The BP-STDP algorithm applied to a multi-layer SNN consisting of input, output and one hidden layer. '*' stands for matrix multiplication. '- -' starts a comment.

Finally, the synaptic weights of hidden layer are updated by

$$w_{hj}(t) = w_{hj}(t) + \Delta w_{hj}(t) \tag{21}$$

The above learning rule can be non-zero when the hidden neuron $h$ fires (postsynaptic spike occurrence). Thus, the weights are updated according to the presynaptic ($s_j(t)$) and postsynaptic ($s_h(t)$) spike times, analogous to the standard STDP rule. Additionally, the derivative of ReLU ($o_h > 0$) is analogous to the spike generation in the IF neurons (see the condition in Eq. 20). Following this scenario for the spatio-temporally synaptic weight change rule, we can build a multi-layer SNN equipped by the STDP-based Backpropagation algorithm, named BP-STDP. Figure 3 demonstrates the BP-STDP algorithm applied to an SNN with one hidden layer.

## 3 Results

We ran three different experiments to evaluate the proposed model (BP-STDP) on the XOR problem, the iris dataset[36], and the MNIST dataset[37]. The parameters used in the experimental setting are shown in Table 1. The synaptic weights for all the experiments are initialized by the Gaussian random numbers with zero mean and unit standard deviation.

### 3.1 XOR problem

The BP-STDP algorithm is evaluated by the XOR problem to show its ability to solve linear inseparability. The dataset contains four data points $\{(0.2,0.2),(0.2,1),(1,0.2),(1,1)\}$ and corresponding labels $\{0,1,1,0\}$. We used 0.2 instead of 0 to activate the IF neurons (to release spikes). The network architecture consists of 2 input, 20 hidden, and 2 output IF neurons. The number of hidden neurons for this problem has little effect on the results. We

**Table 1.** Model parameters.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $\varepsilon$ | 4 ms | $\mu$ | 0.0005 |
| $\beta$ | 250 Hz | $\theta_o$ | $0.025 \times H$ |
| $\theta_h$ | 0.9 | $H$ | {10,...,1500} |
| $\Delta t$ | 1 ms | $U_0$ | 0 |

will investigate the impact of the number of hidden neurons for the MNIST classification task. Each input neuron releases spike trains corresponding to the input values such that the value 1 is represented by a spike train with the maximum spike rate (250 Hz).

Figure 4 shows the training process where each box represents the two output neurons' activities with respect to the four input spike patterns determining $\{0,1,1,0\}$ classes. After around 150 training iterations, the output neurons become selective to the input categories. Figure 5 demonstrates the learning convergence progress using the energy function defined in Eq. 22. This figure shows that the proper learning rates, $\mu$, fall in the range [0.01, 0.0005].

$$\text{MSE} = \frac{1}{N}\sum_{k=1}^{N}\Big(\frac{1}{T}\sum_{t=1}^{T}\xi^k(t)\Big)^2\,,\quad \xi^k(t) = \sum_i \xi_i^k(t) \tag{22}$$

In Eq. 22, $N$ and $\xi_i^k(t)$ denote the training batch size and the error value of output neuron $i$ in response to sample $k$.

### 3.2 Iris dataset

The Iris dataset consists of three different types of flowers (Setosa, Versicolour, and Virginica) represented by the length and width of both petal and sepal (4 features). After normalizing the feature values in the range [0,1], input spike trains are generated. The network architecture in this experiment consists of 4 input, 30 hidden, and 3 output IF neurons. We found, in our preliminary experiments, that using more than 10 hidden neurons does not significantly improve accuracy.

Similar to the results obtained for the XOR problem, Figure 6 illustrates that the SNN converges to be selective to the three flower patterns through training. Figure 7 shows the learning process of the SNN in terms of the MSE defined in Eq. 22. The final evaluation using 5-fold cross validation reported 96% accuracy while the traditional neural network equipped by standard Backpropagation showed 96.7% accuracy. This result shows the success of the proposed BP-STDP algorithm to train temporal SNNs. Furthermore, Table 2 compares our results with other spike-based and traditional supervised learning methods. Our model outperforms (or equally performs) the previous multi-spiking supervised learning algorithms except Lin et al.'s method[17] where develops a spatio-temporal, computationally expensive GD.

### 3.3 MNIST Dataset

To assess the proposed algorithm in solving more complex problems, we evaluate the SNN on MNIST with 784 input, 100 through 1500 hidden, and 10 output IF neurons equipped by BP-STDP. The SNN was trained and tested on 60k training and 10k testing samples. The input spike trains are generated by random lags with the spike rates proportional to the normalized pixel values in the range [0, 1].

Figure 8 shows the output neurons' membrane potentials (after training) in response to the spike trains generated from three randomly selected digits. The target neuron's membrane potential grows fast and reaches the threshold while the other neurons' activities are near zero. Furthermore, this fast response ($<$ 9 ms) reduces the network's response latency. Figure 9a shows the learning process for 1200 training epochs. Each epoch stands for 50 MNIST digits. The MSE track in this plot shows the fast convergence of BP-STDP for the learning rates of 0.001 and 0.0005. Figure 9b shows the MSE values and the accuracy rates for the SNN with 1000 hidden neurons over training. After 100 and 900 training epochs, the performances of 90% and 96% are achieved.

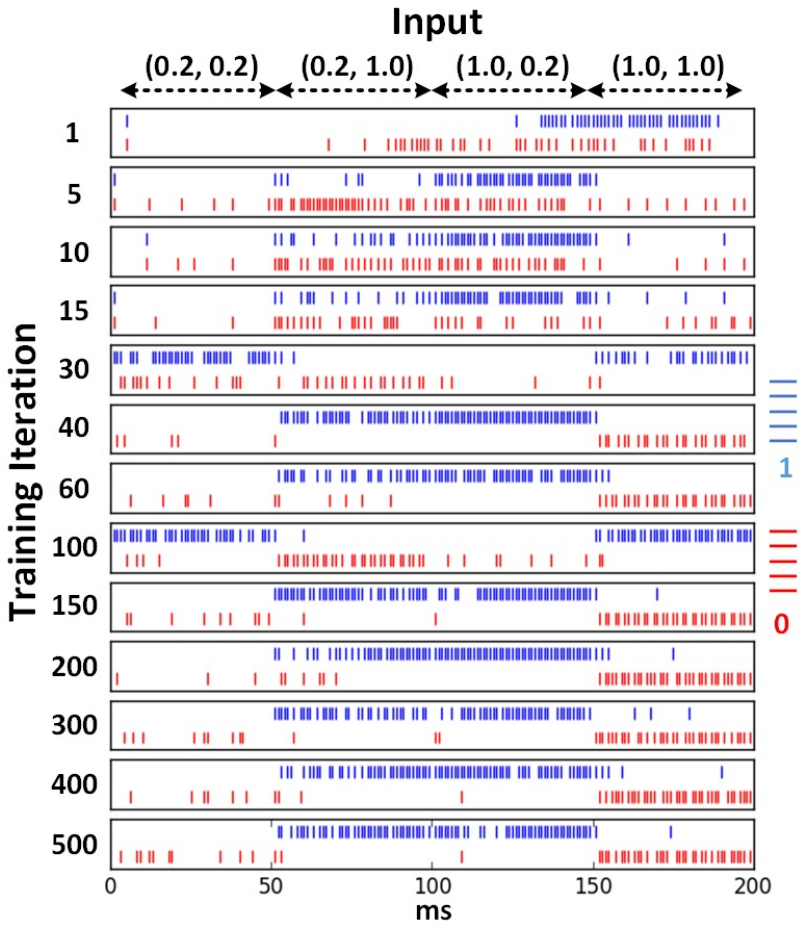


**Figure 4.** Spike trains released from the output neurons in response to four pairs of spike trains representing $\{(0.2, 0.2), (0.2, 1), (1, 0.2), (1, 1)\}$ values in 1 through 500 iterations. We used high spike rates for better visualization.

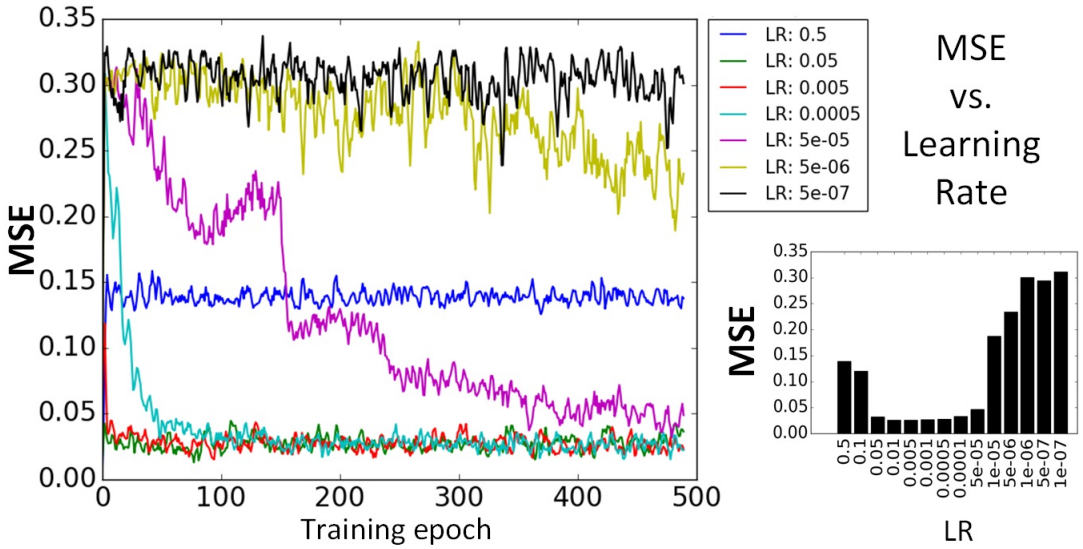


**Figure 5.** MSE of the XOR learning process (left: through 500 iterations, right: after training).

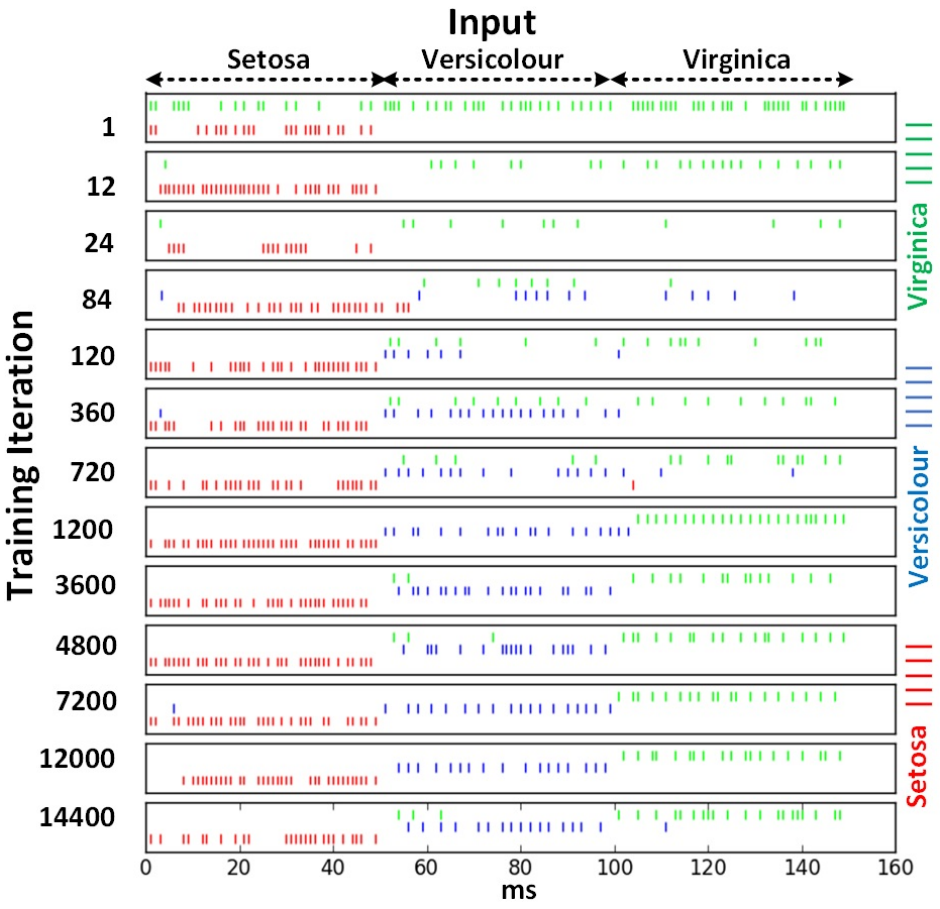


**Figure 6.** Spike trains released from the output neurons in response to the spike trains representing Setosa, Versicolour, and Virginica.

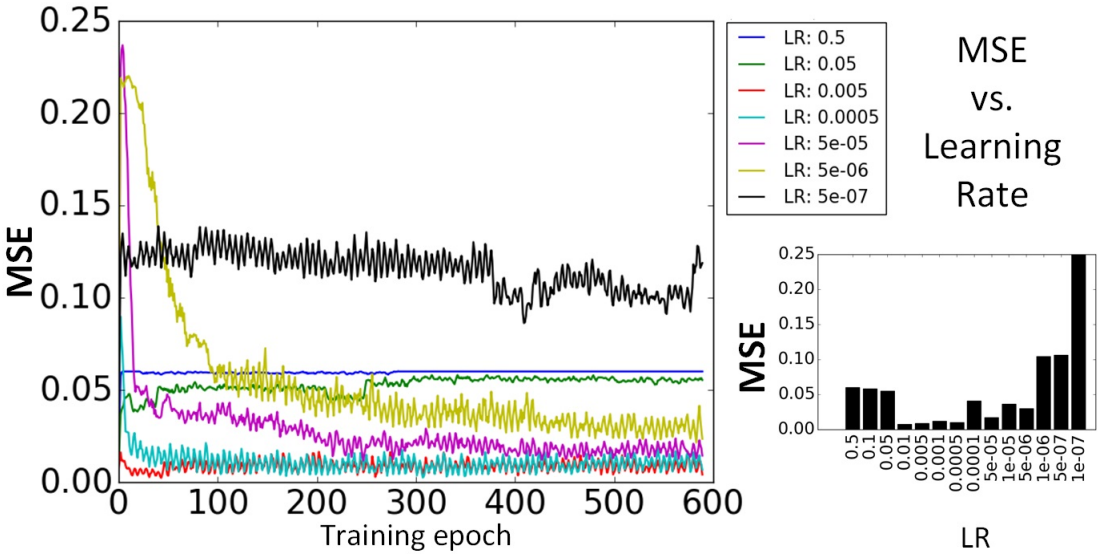


**Figure 7.** MSE of the learning process applied to the Iris dataset (left: through 500 iterations, right: after training).

**Table 2.** Accuracy of Iris classification using our method (BP-STDP) in comparison with the other spiking supervised approaches and the classical ANN, SVM, and Naive Bayes methods.

| ***Model*** | ***Accuracy %*** |
|---|---|
| SpikeProp (Bohte et al. 2002)[6] | 96.1 |
| MSGD; Xu et al. (2013)[10] | 94.4 |
| Wang et al. (2014)[31] | 86.1 |
| Yu et al. (2014)[26] | 92.6 |
| SWAT; Wade et al. (2010)[38] | 95.3 |
| multi-ReSuMe; Sporea et al. (2013)[22] | 94.0 |
| Xie et al. (2016)[20] | 96.0 |
| Lin et al. (2017)[17] | 96.7 |
| SVM and Bayesian[31] | 96.0 |
| ANN (30 hidden neurons) | 96.7 |
| **BP-STDP** | **96.0** |

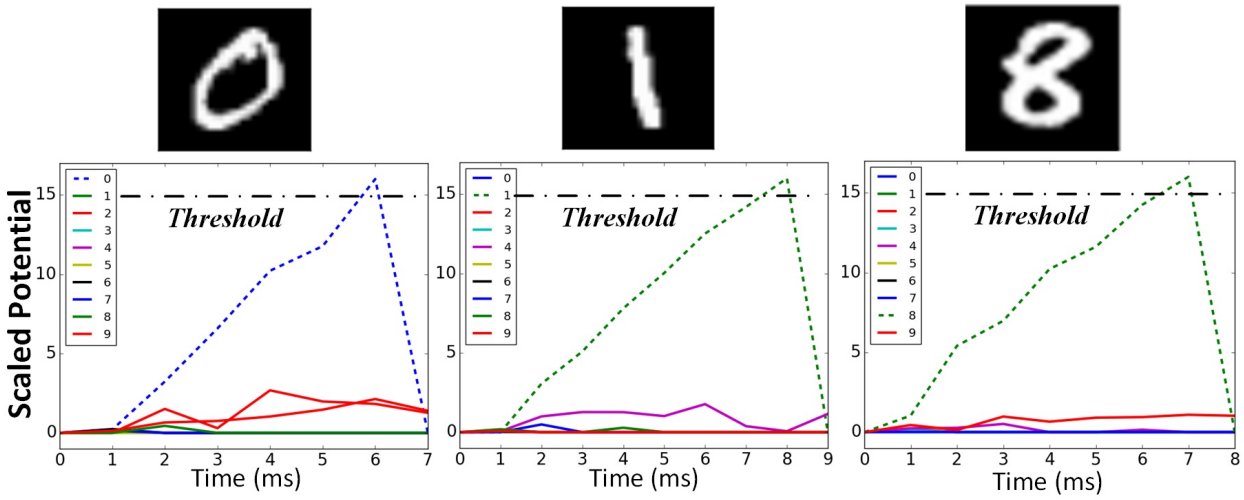


**Figure 8.** Temporal membrane potentials of output neurons in response to randomly selected '0', '1', and '8' handwritten digits.

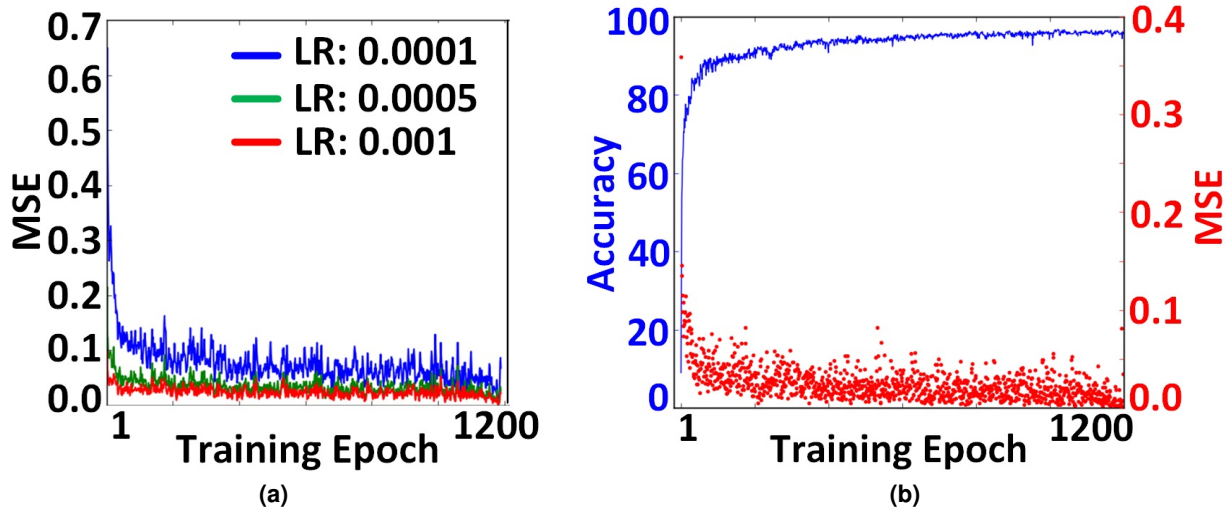


**Figure 9.** (a): MSE of the SNN over training with $\mu = 0.001, 0.0005, 0.0001$. (b): MSE and accuracy of the SNN with $\mu = 0.0005$. $H = 1000$ and each training epoch stands for 50 images.

**Table 3.** MNIST classification performance of the proposed BP-STDP applied to SNNs in comparison with traditional Backpropagation applied to conventional neural networks. '*' denotes a random distortion of training set.

| ***Model*** | ***H=300*** | ***H=1000*** |
|---|---|---|
| ANN[39] | 95.3 | 95.5 |
| ANN*[39] | 96.4 | 96.2 |
| **BP-STDP** | 95.7 | **96.6** |
| *Deep SNNs using GD of membrane potentials* | | |
| Lee et al. (2016)[18] | 98.71 | |

To examine the impact of the number of hidden neurons on performance, we applied the BP-STDP rule to six SNNs with 100 through 1500 hidden IF neurons. Figure 10 shows the accuracy rates over training for these SNNs. The best accuracy rates belong to the networks with more than 500 hidden neurons.

Table 3 compares the proposed supervised learning method (BP-STDP) with the traditional backpropagation algorithm (GD). This comparison confirms the success of the bio-inspired BP-STDP rule applied to the temporal SNN architecture. This table also shows the performance of the recently developed deep SNN using backpropagation[18]. The deep SNN accuracy is 2.1% higher than the BP-STDP's accuracy; however, it develops computationally expensive derivatives of the neurons' membrane potentials instead of their spike events.

## 4 Discussion

BP-STDP introduces a new vein of supervised learning for SNNs and it showed promising performances comparable with the state-of-the-art conventional gradient descent approaches. BP-STDP provides bio-inspired local learning rules which take spike times into consideration as well as spike rates in different layers of presynaptic and postsynaptic IF neurons. Bengio et al.[40] showed that the synaptic weight change is proportional to the presynaptic spike event and the postsynaptic temporal activity that is analogous to the STDP rule and confirms Hinton's idea that says STDP can be associated with the postsynaptic temporal rate[41]. In this paper, we showed that the backpropagation update rules can derive spatio-temporal learning rules, which implement biologically plausible STDP in SNNs.

The proposed algorithm is inspired from the backpropagation update rules derived for the conventional networks of ReLU neurons to develop biologically plausible, temporally local learning rules in an SNN. This matter was

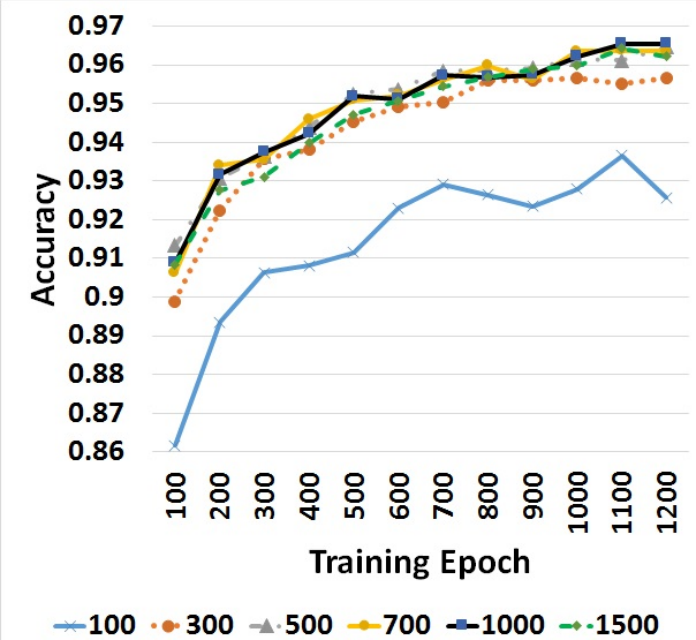


**Figure 10.** Performance of BP-STDP applied to the SNN with 100 through 1500 hidden IF neurons over 100 through 1200 training epochs.

accomplished by an initial approximation of the IF neurons to the ReLU neurons to support the spike-based communication scheme in SNNs. The spiking supervised learning rules offer a combination of STDP and anti-STDP applied to spiking neural layers corresponding to the temporal presynaptic and postsynaptic neural activities. Therefore, we take advantages of both accurate gradient descent and efficient, temporally local STDP in spiking frameworks. The main question is that how does error propagation corresponds to the spiking behavior of IF neurons in multi-layer network architectures? To answer this question, let us assume the error value as a signal that either stimulates ($\sum_i \xi_i > 0$) or suppresses ($\sum_i \xi_i < 0$) the IF neuron to fire. Stimulating (suppressing) a neuron refers to increasing (decreasing) its membrane potential that is proportionally controlled by its input weights and presynaptic spike events. Thus, as the BP-STDP update rules change the synaptic weights temporally based on the error signal and the presynaptic spike times, it manipulates the neural activities (action potentials) of hidden layers at each time step.

The experimental results showed the success of BP-STDP in supervised learning embedded in bio-inspired SNNs. The XOR problem proved the ability of BP-STDP to classify non-linearly separable samples represented by spike trains through $T$ ms time intervals. The complex problems of IRIS and MNIST classification demonstrated comparable performances (96.0% and 96.6% respectively) to the conventional backpropagation algorithm and spiking gradient descent approaches while BP-STDP offers more biologically plausible and temporally local learning rules to take one step closer to the efficient computations that occur in the brain. To the best of our knowledge, this approach is the first high performance model addressing temporal and biological learning concerns without directly implementing the computationally expensive gradient descent rules.

## 5 Conclusion

This paper showed that IF neurons approximate rectified linear units, if the neurons' activities are mapped to the spike rates. Hence, a network of spiking IF neurons can undergo backpropagation learning applied to conventional NNs. We proposed a temporally local learning rule (derived from the traditional backpropagation updates) incorporating the STDP and anti-STDP rules embedded in a multi-layer SNN of IF neurons. This model (BP-STDP) takes advantages of the bio-inspired, efficient STDP rule in spiking platforms and the power of GD for training multi-layer networks. Also, converting the GD-based weight change rules to the spike-based STDP rules is much easier and

computationally inexpensive than developing a spiking GD rule. The experiments on the XOR problem showed that the proposed SNN can classify non-linearly separable patterns. Furthermore, the final evaluations on the Iris and MNIST datasets demonstrated high classification accuracies comparable to the state-of-the-art, multi-layer networks of traditional and spiking neurons.

The promising results of the BP-STDP model warrants our future investigation to develop a deep SNN equipped by BP-STDP and regularization modules. The deep SNN can be utilized for larger pattern recognition tasks while preserving efficient, brain-like computations.

## References


1. Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural networks* **10**, 1659–1671 (1997).
2. Ghosh-Dastidar, S. & Adeli, H. Spiking neural networks. *Int. journal neural systems* **19**, 295–308 (2009).
3. Maass, W. To spike or not to spike: that is the question. *Proc. IEEE* **103**, 2219–2224 (2015).
4. Neil, D., Pfeiffer, M. & Liu, S.-C. Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 293–298 (ACM, 2016).
5. Kasabov, N., Dhoble, K., Nuntalid, N. & Indiveri, G. Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks* **41**, 188–201 (2013).
6. Bohte, S. M., Kok, J. N. & La Poutre, H. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**, 17–37 (2002).
7. McKennoch, S., Liu, D. & Bushnell, L. G. Fast modifications of the spikeprop algorithm. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, 3970–3977 (IEEE, 2006).
8. Booij, O. & tat Nguyen, H. A gradient descent rule for spiking neurons emitting multiple spikes. *Inf. Process. Lett.* **95**, 552–558 (2005).
9. Ghosh-Dastidar, S. & Adeli, H. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural networks* **22**, 1419–1431 (2009).
10. Xu, Y., Zeng, X., Han, L. & Yang, J. A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Networks* **43**, 99–113 (2013).
11. Xu, Y., Yang, J. & Zhong, S. An online supervised learning method based on gradient descent for spiking neurons. *Neural Networks* **93**, 7–20 (2017).
12. Florian, R. V. The chronotron: a neuron that learns to fire temporally precise spike patterns. *PloS one* **7**, e40233 (2012).
13. Victor, J. D. & Purpura, K. P. Metric-space analysis of spike trains: theory, algorithms and application. *Network: computation neural systems* **8**, 127–164 (1997).
14. Huh, D. & Sejnowski, T. J. Gradient descent for spiking neural networks. *arXiv preprint arXiv:1706.04698* (2017).
15. van Rossum, M. C. A novel spike distance. *Neural computation* **13**, 751–763 (2001).
16. Zenke, F. & Ganguli, S. SuperSpike: Supervised learning in multi-layer spiking neural networks. *arXiv preprint arXiv:1705.11146* (2017).
17. Lin, X., Wang, X. & Hao, Z. Supervised learning in multilayer spiking neural networks with inner products of spike trains. *Neurocomputing* **237**, 59–70 (2017).
18. Lee, J. H., Delbruck, T. & Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Front. neuroscience* **10** (2016).

**19.** Wu, Y., Deng, L., Li, G., Zhu, J. & Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *arXiv preprint arXiv:1706.02609* (2017).

**20.** Xie, X., Qu, H., Liu, G., Zhang, M. & Kurths, J. An efficient supervised training algorithm for multilayer spiking neural networks. *PloS one* **11**, e0150329 (2016).

**21.** Ponulak, F. & Kasiński, A. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Comput.* **22**, 467–510 (2010).

**22.** Sporea, I. & Grüning, A. Supervised learning in multilayer spiking neural networks. *Neural computation* **25**, 473–509 (2013).

**23.** Mohemmed, A., Schliebs, S., Matsuda, S. & Kasabov, N. SPAN: Spike pattern association neuron for learning spatio-temporal spike patterns. *Int. J. Neural Syst.* **22**, 1250012 (2012).

**24.** Mohemmed, A., Schliebs, S., Matsuda, S. & Kasabov, N. Training spiking neural networks to associate spatio-temporal input–output spike patterns. *Neurocomputing* **107**, 3–10 (2013).

**25.** Gütig, R. & Sompolinsky, H. The tempotron: a neuron that learns spike timing-based decisions. *Nat. neuroscience* **9**, 420 (2006).

**26.** Yu, Q., Tang, H., Tan, K. C. & Yu, H. A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing* **138**, 3–13 (2014).

**27.** Xu, Y., Zeng, X. & Zhong, S. A new supervised learning algorithm for spiking neurons. *Neural computation* **25**, 1472–1511 (2013).

**28.** Markram, H., Gerstner, W. & Sjöström, P. J. Spike-timing-dependent plasticity: a comprehensive overview. *Front. synaptic neuroscience* **4** (2012).

**29.** Song, S., Miller, K. D. & Abbott, L. F. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. neuroscience* **3**, 919–926 (2000).

**30.** Caporale, N. & Dan, Y. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.* **31**, 25–46 (2008).

**31.** Wang, J., Belatreche, A., Maguire, L. & Mcginnity, T. M. An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing* **144**, 526–536 (2014).

**32.** Tavanaei, A. & Maida, A. S. A spiking network that learns to extract spike signatures from speech signals. *Neurocomputing* **240**, 191–199 (2017).

**33.** Bishop, C. M. *Pattern recognition and machine learning* (springer, 2006).

**34.** Goodfellow, I., Bengio, Y. & Courville, A. *Deep learning* (MIT press, 2016).

**35.** Tavanaei, A., Masquelier, T. & Maida, A. Representation learning using event-based STDP. *arXiv preprint arXiv:1706.06699* (2017).

**36.** Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals human genetics* **7**, 179–188 (1936).

**37.** LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

**38.** Wade, J. J., McDaid, L. J., Santos, J. A. & Sayers, H. M. Swat: a spiking neural network training algorithm for classification problems. *IEEE Transactions on Neural Networks* **21**, 1817–1830 (2010).

**39.** LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

**40.** Bengio, Y., Mesnard, T., Fischer, A., Zhang, S. & Wu, Y. Stdp-compatible approximation of backpropagation in an energy-based model. *Neural computation* (2017).

**41.** Hinton, G. How to do backpropagation in a brain. In *Invited talk at the NIPS'2007 Deep Learning Workshop*, vol. 656 (2007).