

# A Highly Effective and Robust Membrane Potential-Driven Supervised Learning Method for Spiking Neurons

Malu Zhang, Hong Qu<sup>ID</sup>, *Member, IEEE*, Ammar Belatreche<sup>ID</sup>, *Member, IEEE*,  
Yi Chen, and Zhang Yi<sup>ID</sup>, *Fellow, IEEE*

**Abstract**—Spiking neurons are becoming increasingly popular owing to their biological plausibility and promising computational properties. Unlike traditional rate-based neural models, spiking neurons encode information in the temporal patterns of the transmitted spike trains, which makes them more suitable for processing spatiotemporal information. One of the fundamental computations of spiking neurons is to transform streams of input spike trains into precisely timed firing activity. However, the existing learning methods, used to realize such computation, often result in relatively low accuracy performance and poor robustness to noise. In order to address these limitations, we propose a novel highly effective and robust membrane potential-driven supervised learning (MemPo-Learn) method, which enables the trained neurons to generate desired spike trains with higher precision, higher efficiency, and better noise robustness than the current state-of-the-art spiking neuron learning methods. While the traditional spike-driven learning methods use an error function based on the difference between the actual and desired output spike trains, the proposed MemPo-Learn method employs an error function based on the difference between the output neuron membrane potential and its firing threshold. The efficiency of the proposed learning method is further improved through the introduction of an adaptive strategy, called skip scan training strategy, that selectively identifies the time steps when to apply weight adjustment. The proposed strategy enables the MemPo-Learn method to effectively and efficiently learn the desired output spike train even when much smaller time steps are used. In addition, the learning rule of MemPo-Learn is improved further to help mitigate the impact of the input noise on the timing accuracy and reliability of the neuron firing dynamics. The proposed learning method is thoroughly evaluated on synthetic data and is further demonstrated on real-world classification

tasks. Experimental results show that the proposed method can achieve high learning accuracy with a significant improvement in learning time and better robustness to different types of noise.

**Index Terms**—Classification, gradient descent, spiking neural networks, spiking neurons, supervised learning.

## I. INTRODUCTION

TRADITIONAL rate-coded artificial neural networks assume that sensory information is conveyed in the firing rate of the biological neuron. However, it is unlikely that rate-based coding can convey all the information related to rapid processing of different sensory modalities such as the stimulus modality for vision, smell, and hearing [1]–[4]. Indeed, spike-timing neural activities have been observed in different brain regions, including the retina [5]–[7], the lateral geniculate nucleus [8], and the visual cortex [9], and the view that information is represented by explicit timing of spikes rather than mean firing rates has received increasing attention [10], [11]. These findings have led to a new way of simulating neural networks based on spiking neurons that encode information by the firing times of spikes [12]–[14]. It has been demonstrated that networks of spiking neurons are computationally more powerful than traditional rate-based neurons [15]–[20]. However, their application to real-world problems remain relatively limited mainly due to their inherent computational complexity and the lack of effective and efficient learning methods. Therefore, the development of highly effective and robust learning methods is now needed more than ever to leverage the computational power of these biologically plausible neural models and to increase their applicability in solving real-world problems.

Supervised learning was proposed as a successful concept of information processing in traditional neural networks. The most documented evidence for supervised learning in the central nervous system comes from the studies on the cerebellum and the cerebellar cortex [21], [22]. However, the exact mechanisms underlying supervised learning in the biological neurons remain an open problem [23], [24]. To date, many supervised learning methods have been proposed in order to train the spiking neurons to generate desired sequences of spikes. This type of learning methods can be broadly classified into two groups: spike-driven methods and membrane potential-driven methods.

Manuscript received February 15, 2017; revised October 6, 2017, December 26, 2017, and April 18, 2018; accepted April 22, 2018. This work was supported in part by the National Science Foundation of China under Grant 61573081, in part by the Foundation for Youth Science and Technology Innovation Research Team of Sichuan Province under Grant 2016TD0018, and in part by the Singapore Government's Research, Innovation, and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain) under Grant A1687b0033. (Corresponding author: Hong Qu.)

M. Zhang, H. Qu, and Y. Chen are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: maluzhang@126.com; hongqu@uestc.edu.cn; cheniyi0923@hotmail.com).

A. Belatreche is with the Department of Computer and Information Sciences, Faculty of Engineering and Environment, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: ammar.belatreche@northumbria.ac.uk).

Z. Yi is with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: zhangyi@scu.edu.cn).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TNNLS.2018.2833077

Spike-driven methods use the desired and actual output spikes as the relevant signals for controlling synaptic change. Typical examples of these methods include SpikeProp [25] and the multispikes learning algorithm [26] which construct an error function using the difference between the desired and actual output spikes, then use its gradient as the basis for updating the synaptic connection weights. Remote supervised method (ReSuMe) [23] is another spike-driven method, where synaptic weight changes are driven by the joint effect of two opposite processes: 1) strengthening of the synaptic weights through spike-timing-dependent plasticity (STDP) based on the relative timing of the input and the desired output spike trains and 2) weakening of the synaptic weights through anti-STDP based on the relative timing of the input and the actual output spike trains. In order to enhance the learning performance of ReSuMe, the integration of the delay shift approach with ReSuMe-based weight adjustment has recently been proposed in delay learning ReSuMe (DL-ReSuMe) [24]. Again, the desired and actual output spikes are used as the relevant signals for controlling synaptic change in the chronotron [27] and the Spike Pattern Association Neuron (SPAN) learning methods [28]. Both methods try to minimize the distance between the desired and actual output spike trains. Such distance is defined by the Victor and Purpura metric in the chronotron E-learning method [29], while in the case of the SPAN method, it is based on a metric similar to the van Rossum metric [30]. Common disadvantages of the above-mentioned methods include relatively low learning efficiency and accuracy.

On the other hand, the membrane potential-driven methods emerged recently in an attempt to improve the learning efficiency and accuracy in spiking neurons. Typical examples of these methods include the tempotron [31], perceptron-based spiking neuron learning rule (PBSNLR) [32], and high-threshold projection (HTP) [33]. Compared with their spike-driven counterparts, membrane potential-driven methods take an entirely different approach where the postsynaptic membrane potential (instead of spike times) is considered as the relevant signal for controlling synaptic change. For instance, the tempotron implements a gradient descent dynamics that minimizes an error defined as the difference between the maximum membrane potential and the firing threshold. However, this reliance on the maximum membrane potential as its objective function prevents the binary tempotron learning rule from controlling more than one spike [34]. PBSNLR [32] and HTP [33] perform a perceptron classification on discretely sampled time points of the membrane potential, with the aim to keep membrane potential below threshold at undesired spike times and to make sure a threshold crossing occurs at desired spike times [35]. As they are based on the perceptron learning rule, in theory, the desired output spike train cannot be learned successfully if the sampled time points of the membrane potential are not linearly separable [32]. In addition, both memory and time complexities of the training method increase considerably when a small time step is used. Therefore, further enhancements of the learning performance are still needed for this type of learning methods.

Furthermore, another important aspect often overlooked when designing learning strategies for spiking neural networks (SNNs) is the robustness to noise. Noise is common in spiking neural networks and can significantly affect the learning performance as well as the timing accuracy and reliability of neural responses [36]–[38]. In order to improve noise robustness of the trained neurons, most of the existing supervised learning methods use noisy samples during the training phase (i.e., noisy training) [23], [32]. However, the neurons trained with noisy samples are found to show relatively robust responses only to the stimuli used during the training phase, and their response to new stimuli not seen during the training phase is rather unreliable [23]. Therefore, improving the robustness of learning methods for spiking neurons remains an open problem.

In order to address the above-mentioned limitations of existing supervised learning methods for spiking neurons, we propose, in this paper, a novel highly effective and noise robust membrane potential-driven supervised learning (MemPo-Learn) method for spiking neurons with significant improvement in the learning efficiency. The proposed learning method, called MemPo-Learn, is able to generate desired spike trains with higher accuracy, higher efficiency, and better robustness to input jitter as well as voltage noise. The efficiency of the MemPo-Learn is significantly improved through the introduction of an adaptive strategy, called skip scan training strategy (SSTS), which enables the MemPo-Learn method to accurately and efficiently learn the desired output spike train even when much smaller time steps are used. In addition, we analyze the noise robustness of the proposed MemPo-Learn method and introduce further improvements to make it significantly more robust to noise. The performance of the proposed learning method is thoroughly evaluated on synthetic data and is further demonstrated on real-world classification tasks. Experimental results demonstrate that the proposed method is superior to other supervised methods in terms of the three key performance factors of supervised learning for spiking neurons, namely, learning accuracy, learning efficiency, and robustness to noise.

The remainder of this paper is organized as follows. Section II introduces the neuron model and the learning rule of the proposed MemPo-Learn method. Sections III and IV present a detailed description of the proposed strategies for improving the efficiency and noise robustness of MemPo-Learn. Section V presents a comprehensive experimental evaluation of the proposed MemPo-Learn method on synthetic spatiotemporal data including extensive experiments to explore the effect of different learning parameters on its learning performance. Further demonstration of the proposed learning method on real-world applications is also presented in this section. Finally, Section VI discusses the results and draw conclusions.

## II. MEMPO-LEARN LEARNING RULE

In this section, we begin by presenting the neuron model. Then, the main idea of the proposed MemPo-Learn learning rule is described.

### A. Neuron Model

There are many spiking neuron models that aim to capture the dynamics of biological neurons [12], [39], [40]. The spike-response model (SRM) can give a faithful description of biological neurons [12]. In addition, the SRM model can easily be implemented numerically; hence, it is used in this paper.

In the SRM model, the membrane potential of a neuron  $i$  is represented by a variable  $u_i$  which remains at the resting potential,  $u_{\text{rest}} = 0$ , when there is no spike received from the presynaptic neurons. When a spike produced at a presynaptic neuron  $j$ , a postsynaptic potential (PSP) is induced in neuron  $i$ . After the integration of the PSPs resulting from several incoming spikes, the postsynaptic neuron  $i$  fires a spike when its membrane potential  $u_i$  reaches a certain firing threshold  $\vartheta$ . Let us assume that neuron  $i$  has fired its last spike at time  $\hat{t}$ . After firing the evolution of  $u_i$  is given as

$$u_i(t) = \eta(t - \hat{t}) + \sum_j \omega_{ji} \sum_f \varepsilon_{ji}(t - t_j^f) + u_{\text{rest}} \quad (1)$$

where  $t_j^f$  is the  $f$ th spike of presynaptic neuron  $j$ , and  $\omega_{ji}$  is the synaptic weight from neurons  $j$  to  $i$ . The PSP induced by the spike  $t_j^f$  is determined by the spike response function  $\varepsilon_{ji}(t - t_j^f)$  defined as

$$\varepsilon_{ji}(t - t_j^f) = \frac{t - t_j^f}{\tau} \exp\left(1 - \frac{t - t_j^f}{\tau}\right), \quad \text{if } t - t_j^f > 0 \quad (2)$$

where  $\tau$  is a time decay constant that determines the spread shape of the spike response function. The refractoriness function  $\eta(t - \hat{t})$  is defined as

$$\eta(t - \hat{t}) = -\vartheta \exp\left(-\frac{t - \hat{t}}{\tau_R}\right), \quad \text{if } t - \hat{t} > 0 \quad (3)$$

where  $\tau_R$  is a time decay constant.

### B. MemPo-Learn Learning Rule

The aim of supervised learning is to adjust the synaptic weights of a spiking neural network such as an output neuron emits a desired spike train in response to a given input spike pattern. Therefore, the running time of an output spiking neuron  $i$  can be divided into two sets: the times of desired output spikes denoted by  $t_d$  ( $t_d = \{t_d(1), t_d(2), \dots, t_d(i), \dots\}$ ) and the remaining times, denoted by  $N_{td}$ . Based on these two different time classes, the proposed MemPo-Learn learning method employs two weight update processes: 1) adjusting synaptic weights to make the membrane potential reach the firing threshold at desired output times  $t_d$  and 2) adjusting synaptic weights to maintain the membrane potential lower than the threshold at undesired output times  $N_{td}$ . These two weight update processes are introduced in Sections II-B1 and II-B2.

1) *Weight Update Rule at Desired Output Spikes  $t_d$* : For any time point in  $t_d$ , in order to fire a spike, the value of the neuron membrane potential is expected to cross the firing threshold from below. To achieve this, MemPo-Learn implements a gradient descent learning rule operating on the membrane potential at desired output times, with the aim

to increase it toward the neuron firing threshold. When the membrane potential is below the firing threshold at desired output times, in order to make the membrane potential  $u_i(t)$  reach the firing threshold  $\vartheta$ , an error function is constructed as follows:

$$E_{t_d} = \frac{1}{2}[u_i(t) - \vartheta]^2, \quad \text{if } u_i(t) < \vartheta, \quad t \in t_d \quad (4)$$

where  $\vartheta$  represents the firing threshold and  $u_i(t)$  represents its postsynaptic membrane potential. During the learning process, to turn OFF the effect of threshold crossings at wrong times,  $\hat{t}$  involved in the calculation of  $u_i(t)$  is not the actual but the desired output spike time [32], [33].

In gradient-based learning, changes in the synaptic weights are given as

$$\Delta\omega_{ji} = -\beta_1 \frac{\partial E_{t_d}}{\partial \omega_{ji}} \quad (5)$$

where  $\beta_1$  is the learning rate that defines the size of the synaptic update at desired spiking times.

If the membrane potential  $u_i(t)$  is below the firing threshold  $\vartheta$  at desired output time, according to (5), synaptic weight  $\omega_{ji}$  is increased by the following amount:

$$\Delta\omega_{ji} = -\beta_1[u_i(t) - \vartheta] \sum_f \varepsilon_{ji}(t - t_j^f). \quad (6)$$

2) *Weight Update Rule at Undesired Output Spikes  $N_{td}$* : For any time point in  $N_{td}$ , in order to avoid the occurrence of undesired output spikes, the membrane potential is required to remain below the neuron firing threshold. The proposed MemPo-Learn achieves this by again using a gradient descent learning rule. When the membrane potential is equal to or greater than the neuron firing threshold, to keep the membrane potential  $u_i(t)$  below the firing threshold  $\vartheta$ , an error function at  $N_{td}$  is defined as

$$E_{N_{td}} = \frac{1}{2}[u_i(t) - (\vartheta - p)]^2, \quad \text{if } u_i(t) \geq \vartheta, \quad t \in N_{td} \quad (7)$$

where the parameter  $p$  determines the magnitude of modification on the synaptic weights at  $N_{td}$ . Then, the synaptic weights at  $N_{td}$  are updated according to the following equation:

$$\Delta\omega_{ji} = -\beta_2 \frac{\partial E_{N_{td}}}{\partial \omega_{ji}} \quad (8)$$

where  $\beta_2$  is the learning rate.

In order to drive the membrane potential below the threshold at  $N_{td}$ , the synaptic efficacy  $\omega_{ji}$  is decreased by the following amount:

$$\Delta\omega_{ji} = -\beta_2[u_i(t) - (\vartheta - p)] \sum_f \varepsilon_{ji}(t - t_j^f). \quad (9)$$

### C. Correlation-Based Metric

To quantitatively evaluate the learning performance of the proposed MemPo-Learn method, a correlation-based metric, introduced in [41], is adopted to measure the similarity between the desired and actual output spike trains. The metric, defined in the following equation, is calculated after each learning epoch as follows:

$$C = \frac{\vec{v}_d \cdot \vec{v}_o}{|\vec{v}_d| |\vec{v}_o|} \quad (10)$$

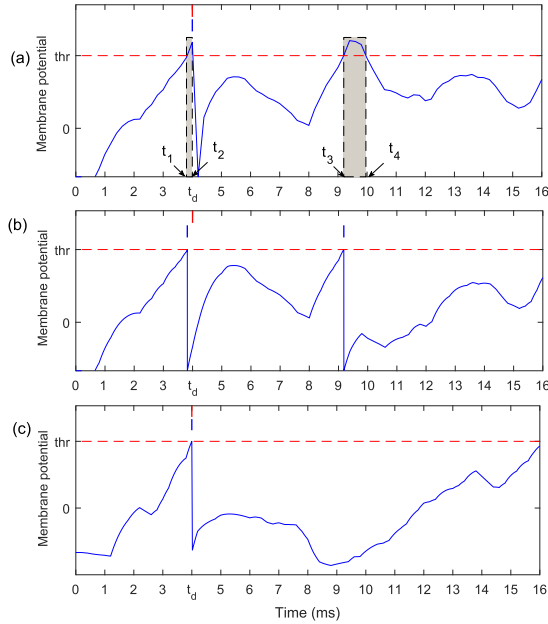


Fig. 1. Learning performance is related to time step. (a) Membrane potential trace after a successful learning with a time step of 1 ms. Desired output time  $t_d$  and actual output time are marked by red vertical bar and blue vertical bar, respectively. (b) When the time step is 0.01 ms, the threshold crossing of the membrane potential within  $[t_1, t_2]$  and  $[t_3, t_4]$  will produce undesired spikes. (c) Membrane potential trace after a successful learning with a time step of 0.01 ms.

where  $\vec{v}_d$  and  $\vec{v}_o$  are the vectors representing a convolution (in discrete time) of desired and actual output spike trains with a low-pass Gaussian filter.  $\vec{v}_d \cdot \vec{v}_o$  is the inner product, and  $|\vec{v}_d|$  and  $|\vec{v}_o|$  are the Euclidean norms of  $\vec{v}_d$  and  $\vec{v}_o$ , respectively.

The Gaussian filter function with parameter  $\sigma$  is given by  $f(t, \sigma) = \exp(-t^2/2\sigma^2)$ , where the parameter  $\sigma$  determines the width of the function. The closer the value of  $C$  comes to 1, the more similar the two spike trains with a value of  $C = 1$  indicating identical spike trains. On the other hand, the closer the value of  $C$  comes to 0, the less similar the two spike trains.

### III. ENHANCING MEMPO-LEARN EFFICIENCY THROUGH THE SKIP SCAN TRAINING STRATEGY

In this section, we first analyze the effect of using a small time step on the learning complexity, and then, we propose a strategy, called STSS.

#### A. Learning With a Small Time Step

Fig. 1(a) shows that using a time step of 1 ms, the membrane potential has been kept below the threshold (i.e.,  $u < \vartheta$ ) at undesired spike times (namely, time points 1, 2, 3, 5, 6, 7, ... ms), and it has been pushed above the firing threshold (i.e.,  $u \geq \vartheta$ ) at desired output time (i.e., at time point 4 ms). As a result, a perfect learning of the desired output spiking time has been achieved. However, this learning may fail when a smaller time step (such as 0.01 ms) is used because the threshold crossing of the membrane potential within  $[t_1, t_2]$  and  $[t_3, t_4]$  will occur at an undesired time [e.g., earlier than the

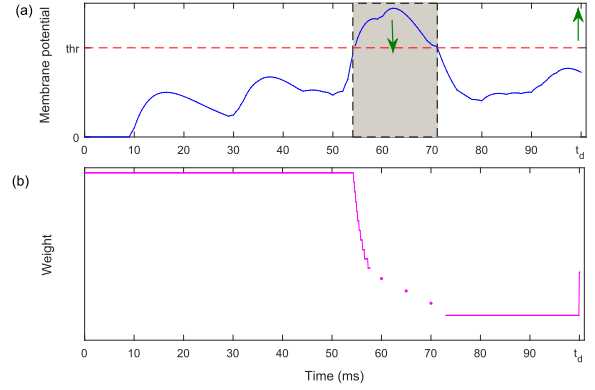


Fig. 2. Learning with a smaller time step is easy to fall into overadjustment. (a) Membrane potential is above threshold at  $N_{td}$  (as depicted by the grayed area). (b) Weight changing during one learning epoch.

desired time or with an additional undesired spike as illustrated in Fig. 1(b)]. This time step related problem arises due to the discrete-time simulation. In order to achieve a successful learning with a small time step, the update of synaptic weights should take the following constraints into account: 1) the membrane potential should remain below the firing threshold at all undesired output times and 2) the membrane potential at desired times  $t_d$  should be equal to the firing threshold [as shown in Fig. 1(c)]. However, like the existing learning methods, it is not easy for MemPo-Learn to meet these two constraints combined. The reasons are outlined in the following.

1) *Overadjustment at  $N_{td}$* : Fig. 2 shows the membrane potential trace of a neuron before learning, in which the membrane potential is above threshold when  $t \in [53, 70]$  ms (as depicted by the gray area). In order to push the membrane potential below the firing threshold with a smaller time step, as shown in Fig. 2(b), the synaptic weight will need to be adjusted continuously and be decreased great deal to be brought below the firing threshold. This overadjustment at  $N_{td}$  may also drive the membrane potential much lower than the firing threshold at desired spiking times  $t_d$ , and also results in a much increased synaptic weights needed at desired spiking times  $t_d$ .

2) *Inadequate Learning for  $t_d$* : The use of the error function proposed in (4) for synaptic update results in the membrane potential exceeding the firing threshold at desired spiking times  $t_d$  (i.e.,  $u(t_d) \geq \vartheta$ ). However, when a smaller time step is used, the membrane potential at  $t_d$  is required to be equal to the firing threshold. Therefore, we use a modified form of (4) as follows:

$$E_{td} = \frac{1}{2}[u_i(t) - \vartheta]^2 \quad \text{if } t \in t_d. \quad (11)$$

It is easy to find that making membrane potential exactly equal to the firing threshold at desired spiking times  $t_d$  is difficult, and more learning opportunities should be given to  $t_d$ . However, all the existing learning methods give only one learning chance to  $t_d$  during one learning epoch.



TABLE I  
MAIN SYMBOLS USED IN SSTS

Symbols	Means
$s_t$	The starting time point of desired output spike train
$t_s$	The simulation time step
$t_f^n$	The first training time point of $n$ th sub-epoch
$Sub(n)$	The set of all the training time points in $n$ th sub-epoch
$s_{len}$	A fixed time length of skip.
$T$	The length of the desired output spike train;
$t_d(i)$	The $i$ th spike of the desired output time.

### B. Skip Scan Training Strategy

The SSTS strategy is proposed to address the overadjustment at  $N_{td}$  and inadequate learning at  $t_d$ . The reason of overadjustment is that, with a small time step, the synaptic weights would be decreased continuously. To avoid this, SSTS divides one learning epoch into many subepochs. In each subepoch, MemPo-Learn skips a period of time denoted by  $s_{len}$  to train the synaptic weights. On the other hand, in order to guarantee enough learning opportunities at  $t_d$ , the MemPo-Learn is applied to update the synaptic weight at  $t_d$  in each subepoch. The meanings of the symbols used in SSTS are shown in Table I, and the detailed pseudocode of the SSTS is shown in Algorithm 1.

#### Algorithm 1 MemPo-Learn Rule Combined With SSTS

1) Divide one learning epoch into many sub-epochs according to SSTS

For  $n = 1 : 1 : s_{len}/t_s$

a) Choosing the first training time point of  $n$ th sub-epoch:

$$t_f^n = s_t + (n - 1)t_s;$$

b) Continue to add  $s_{len}$  to get other training time points

until we reach the end of the spike train  $T$ :

$$Sub(n) = [t_f^n, t_f^n + s_{len}, t_f^n + 2 \times s_{len}, t_f^n + 3 \times s_{len}, \dots];$$

c) Add all the desired output times into the training time points of each sub-epoch:

$$Sub(n) = [t_f^n, t_f^n + s_{len}, t_d(1), t_f^n + 2 \times s_{len}, t_f^n + 3 \times s_{len}, t_d(2), \dots];$$

EndFor

2) Update synaptic weights according to the MemPo-Learn rule

For  $n = 1 : 1 : s_{len}/t_s$

For  $i = 1 : 1 : \text{length}(Sub(n))$

if  $Sub(n, i) \in t_d$

Update synaptic weight according

to (11) and (6);

Endif

if  $Sub(n, i) \notin t_d$

Update synaptic weight according

to (7) and (8);

Endif

Endfor

Endfor

To illustrate the proposed SSTS strategy more clearly, an example of SSTS is shown in Fig. 3. The parameters are set as follows:  $s_t = 0.01$  ms,  $t_s = 0.01$  ms,  $s_{len} = 2$  ms,

TABLE II  
MAIN STEPS TO GENERATE TRAINING TIME POINTS IN EACH SUBEPOCH

$Sub(1)$	a) The first training time point: $t_f^1 = s_t + (n - 1)t_s = 0.01$ ms; b) Continue to add $s_{len}$ to get other training time points: $Sub(1) = \{0.01, 2.01, 4.01, 6.01, 8.01\}$ ms; c) Insert all of the desired output times $t_d(1), t_d(2)$ : $Sub(1) = \{0.01, 2.01, 4.01, t_d(1), 6.01, 8.01, t_d(2)\}$ ms.
$Sub(2)$	a) The first training time point: $t_f^2 = s_t + (n - 1)t_s = 0.02$ ms; b) Continue to add $s_{len}$ to get other training time points: $Sub(2) = \{0.02, 2.02, 4.02, 6.02, 8.02\}$ ms; c) Insert all of the desired output times $t_d(1), t_d(2)$ : $Sub(2) = \{0.02, 2.02, 4.02, t_d(1), 6.02, 8.02, t_d(2)\}$ ms.
...	...
$Sub(200)$	a) The first training time point: $t_f^{200} = s_t + (1 - 1)t_s = 2$ ms; b) Continue to add $s_{len}$ to get other training time points: $Sub(200) = \{2, 4, 6, 8, 10\}$ ms; c) Insert all of the desired output times $t_d(1), t_d(2)$ : $Sub(200) = \{2, 4, t_d(1), 6, 8, t_d(2), 10\}$ ms.

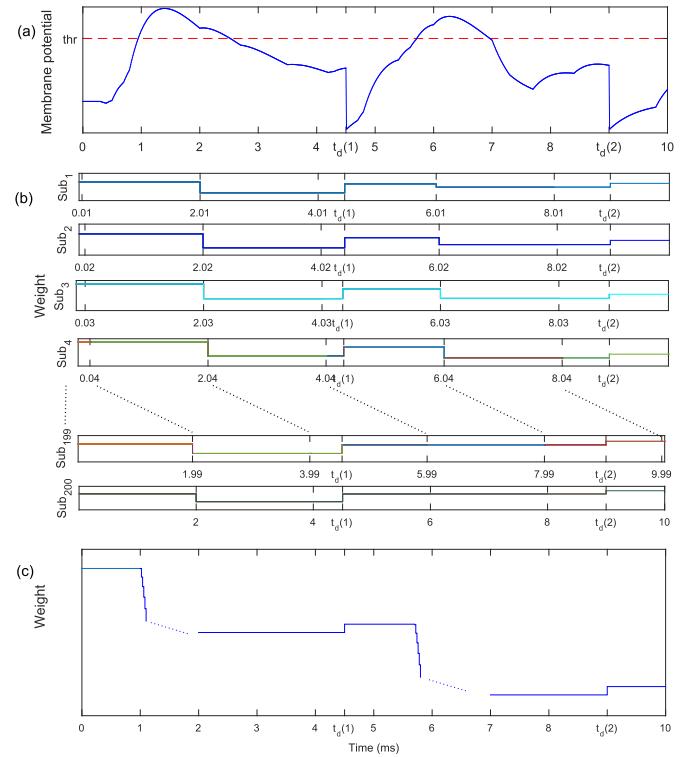


Fig. 3. Illustrative example of SSTS. (a) Membrane potential trace before learning. (b) Weight updating process using SSTS. (c) Weight update process without using SSTS.

and  $T = 10$  ms. The steps for generating training time points in each subepoch are given in Table II. In the  $n$ th subepoch,  $s_t + (n - 1)t_s$  is chosen as the first training time point, and the training time points in the  $n$ th subepoch can be obtained in the same way. It is easy to find when  $n = s_{len}/t_s = 2/0.01 = 200$ , SSTS reaches the last subepoch, and one learning epoch is completed.

Fig. 3(b) shows the weight update process using SSTS, and Fig. 3(c) shows the weight update process without using SSTS. In Fig. 3(b), SSTS divides one learning epoch into many subepochs. In each sub-epoch, all desired output times are

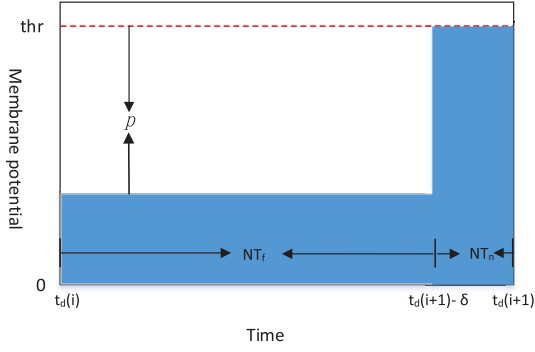


Fig. 4. Illustration of the robust MemPo-Learn.

added into the training time points. In this way, inadequate learning for  $t_d$  is resolved. On the other hand, SSTS divides the continuous  $N_{td}$  period into many subepochs. In each subepoch, persistent decrease of synaptic weights is avoided, which resolves the problem of overadjustment.

#### IV. IMPROVING THE ROBUSTNESS OF MEMPO-LEARN

In this section, we first analyze the robustness of MemPo-learn to noise, then we introduce a strategy that makes MemPo-Learn more robust to noise.

Noise is common in biologically plausible neural networks and can significantly affect the timing accuracy and reliability of the neural responses [36]–[38]. The noise affects the neuron response mainly by: 1) causing spurious spikes to appear or 2) causing desired output spikes to vanish. It is easy to find that if the membrane potential is close to the firing threshold at  $N_{td}$ , the probability of triggering a wrong spike will increase. Therefore, in order to prevent the generation of additional undesired spikes, the membrane potential at  $N_{td}$  should be kept much lower than the firing threshold. On the other hand, to make sure that the neuron will fire nearby  $t_d$ , the membrane potential around  $t_d$  should be strong enough [23].

Based on the above-mentioned analysis, as shown in Fig. 4, we divide the undesired output time  $N_{td}$  into two classes  $NT_f$  (far away from a desired spike) and  $NT_n$  (near a desired spike)

$$NT_f = \{t | t_d(i) < t < t_d(i+1) - \delta\} \quad (12)$$

$$NT_n = \{t | t_d(i+1) - \delta \leq t < t_d(i+1)\} \quad (13)$$

where  $t_d(i)$  and  $t_d(i+1)$  denote the moment of the  $i$ th and  $(i+1)$ th spike in the desired spike train. The parameter  $\delta$  determines the length of  $NT_n$  and  $NT_f$ .

- 1) When  $t \in NT_f$ , to avoid undesired firing, the membrane potential is expected to keep a big distance from the firing threshold. Therefore, the error function of MemPo-Learn at  $NT_f$  is modified as

$$E = \frac{1}{2}[u_i(t) - (\vartheta - p)]^2, \quad \text{if } u_i(t) \geq \vartheta - p, t \in NT_f. \quad (14)$$

According to (14), if  $u_i(t) \geq \vartheta - p$ , the synaptic weights would be reduced to keep the membrane potential lower than the firing threshold by at least  $p$ .

- 2) When  $t \in NT_n$ , the error function is defined as

$$E = \frac{1}{2}[u_i(t) - (\vartheta - p)]^2, \quad \text{if } u_i(t) \geq \vartheta, t \in NT_n. \quad (15)$$

Equation (15) is different from (14), because the period of  $NT_n$  is close to the desired output time. If the membrane potential is kept much lower than the threshold, it will make the spiking neuron hard to output a spike at  $t_d$ .

The error function of the robust MemPo-Learn at  $t_d$  is similar to that of MemPo-Learn, i.e., it is also defined by (4), which ensures a threshold crossing at desired output times. After a successful learning with the robust MemPo-Learn, as shown in Fig. 4, the trace of the neuronal membrane potential only appears in the shaded part.

#### V. SIMULATION RESULTS

Extensive experiments were conducted to thoroughly evaluate the performance of the proposed learning method and assess its tolerance to different parameter variations. We further demonstrate the proposed learning method on real-world classification tasks.

##### A. Learning Performance of MemPo-Learn

In this section, we investigate the effect of different parameters on the learning performance, including the length of spike trains, the number of the synaptic inputs, and the firing rate of spike trains. We compare our method against competitive learning rules for spiking neurons, namely, ReSuMe and PBSNLR which are typical learning methods of the spike-driven methods and the membrane potential-driven methods, respectively. In these simulations, the time step is set to 1 ms.

1) *Effect of the Spike Trains Length*: In these simulations, a neuron with 400 synaptic inputs is trained to reproduce a desired sequence of spikes. Every input spike train and the desired output spike train are generated according to a homogeneous Poisson process with firing rates of 10 and 100 Hz, respectively. Each experiment is repeated for 20 trials for different input and desired output pairs. The initial synaptic weights are randomly drawn from the interval  $[0, 0.05]$  using a uniform distribution. In Fig. 5, the length of the desired output spike trains varies from 400 to 2800 ms with an interval of 400 ms. The average maximum  $C$  value scored during training, the average number of epochs, and the average computing time required to reach the maximum  $C$  are calculated and reported for benchmarking.

Fig. 5(a) illustrates the change in learning accuracies of MemPo-Learn, PBSNLR, and ReSuMe. The learning accuracies of all the methods are very high (with  $C$  reaching 1) when the length of spike trains varies from 400 to 800 ms. While the learning accuracy for ReSuMe and PBSNLR starts declining when the length of spike train exceeds 1200 and 2000 ms, respectively, the learning accuracy of the proposed MemPo-learn is maintained at  $C = 1$  until the length of spike train exceeds 2800 ms. In addition, Fig. 5(b) shows that MemPo-Learn requires much fewer epochs than ReSuMe and PBSNLR in order to reach the maximum accuracy.

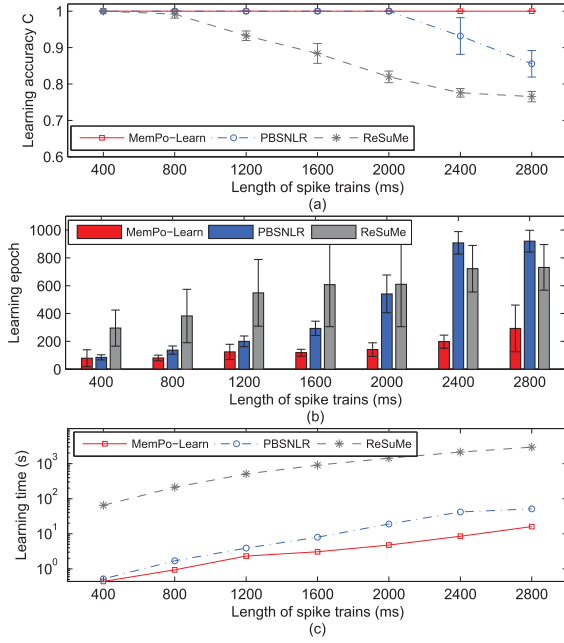


Fig. 5. Comparison of learning performance when the length of desired spike trains increases gradually. Time step = 1 ms.

For instance, for a spike train length of 2000 ms, ReSuMe and PBSNLR require about 600 learning epochs to reach the maximum value of  $C$ , while MemPo-Learn requires only about 200 learning epochs. Moreover, as shown in Fig. 5(c), the learning times of MemPo-Learn and PBSNLR are much better than that of ReSuMe. In addition, the required learning time of MemPo-Learn is comparable with that of PBSNLR for short spike trains (up to 1200 ms) but it is clearly much lower for spike trains of duration greater than or equal to 1600 ms. For example, PBSNLR learning takes almost four times longer than MemPo-Learn when the length of the spike train is 2400 ms, which is a significant improvement in learning efficiency.

2) *Effect of the Number of the Synaptic Inputs:* In this section, we investigate the effect of the number of synaptic inputs. The length of the input and the desired output spike trains is set to 800 ms. Every input spike train and the desired output spike train are generated according to a homogeneous Poisson process with firing rates of 10 and 100 Hz, respectively. The number of synaptic inputs varies from 50 to 500. The experimental results are shown in Fig. 6.

Fig. 6(a) shows that a small number of synaptic inputs result in a low learning accuracy for the three learning methods: MemPo-Learn, ReSuMe, and PBSNLR. However, their learning accuracy increases with the increase of the number of synaptic inputs. MemPo-Learn can quickly reach a very high value of  $C$  (close to 1) using a relatively small number of synaptic inputs. For example, when the number of synaptic inputs is 100, the learning accuracy of MemPo-Learn is almost 1, while the learning accuracies of ReSuMe and PBSNLR are much lower at 0.85 and 0.75, respectively. In terms of learning efficiency, Fig. 6(b) shows a downtrend in the required number of learning epochs for different methods.

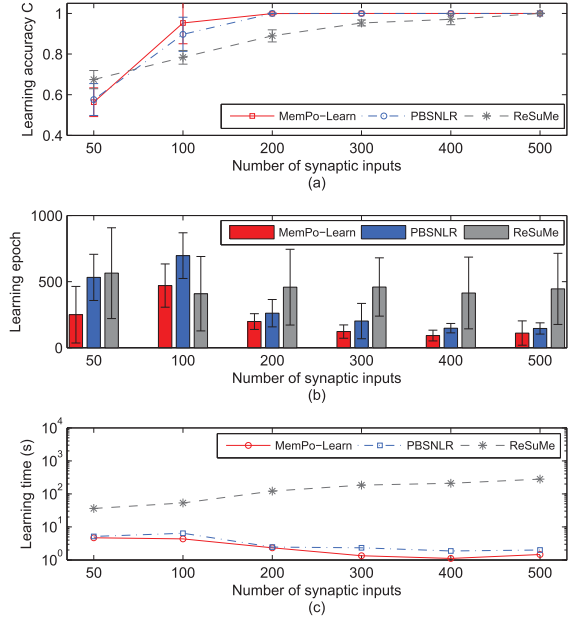


Fig. 6. Comparison of learning performance when the number of the synaptic input increases gradually. Time step = 1 ms.

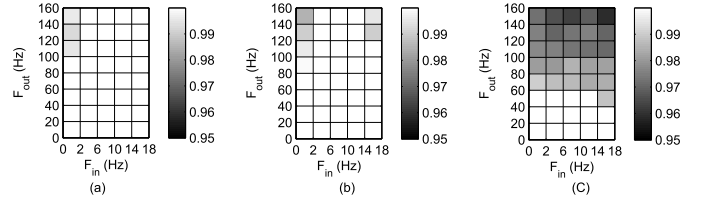


Fig. 7. Comparison of the learning performance against gradually increased firing rates of the input and desired output spike trains. Time step is set to 1 ms. (a) MemPo-Learn. (b) PBSNLR. (c) ReSuMe.

However, the number of required epochs for the proposed MemPo-Learn remains much lower than that of PBSNLR and ReSuMe irrespective of the number of inputs used. Fig. 6(c) clearly illustrates the superiority of MemPo-Learn in terms of the required learning time. Again, the learning efficiency of MemPo-Learn is better than that of PBSNLR and ReSuMe irrespective of the number of inputs. For example, when the number of synaptic input is 100, the learning time of ReSuMe, PBSNLR, and MemPo-Learn are 53.1, 6.4, and 4.3 s, respectively.

3) *Effect of the Firing Rate of the Spike Trains:* The following experiments aim to evaluate the effect of the firing rate of the spike trains. The firing rates of the input spike trains ( $F_{in}$ ) are varied from 2 to 18 Hz with an interval of 4 Hz. The firing rates of the desired output spike trains ( $F_{out}$ ) are varied from 20 to 160 Hz with an interval of 20 Hz. The number of the synaptic inputs is set to 400, and the length of the spike trains is set to 800 ms. The learning is continued for 1000 learning epochs, and the maximum obtained learning accuracy is reported in Fig. 7.

In Fig. 7, MemPo-Learn, PBSNLR, and ReSuMe reach the highest learning accuracy for the lowest value of  $F_{out}$ , and there is a trend that the learning accuracies of all methods decrease with the increase of  $F_{out}$ . On the other hand, the area

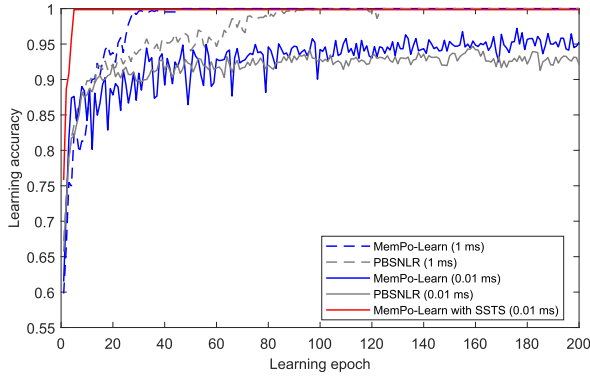


Fig. 8. Comparison of learning performance with different time steps.

in which MemPo-Learn achieves high performance is larger than that of PBSNLR and ReSuMe. For example, when  $F_{in} = 18$  Hz, the performance of MemPo-Learn is 1 for all values of  $F_{out}$  in  $[20, 160]$  Hz. However, in the case of  $F_{in} = 18$  Hz, the learning accuracy of PBSNLR is 1 only when  $F_{out}$  in  $[20, 120]$  Hz, and the learning accuracy of ReSuMe is 1 only when  $F_{out}$  in  $[20, 40]$  Hz.

### B. Learning Performance of MemPo-Learn Combined With SSTS

In this section, we investigate the learning performance of MemPo-Learn combined with SSTS with a time step of 0.01 ms.

1) *Effect of the Time Step*: In the following experiments, a neuron with 400 input synapses is trained to emit a desired sequence of spikes with a length of 400 ms. Every input spike train and desired output spike train are generated randomly according to the homogeneous Poisson process with firing rates of 10 and 100 Hz, respectively. Each experiment is repeated for 20 trials for different input and desired output pairs, and average learning accuracy  $C$  in each learning epoch is reported. The experimental results are shown in Fig. 8.

In Fig. 8, when time step is 1 ms, both MemPo-Learn and PBSNLR reach a value of  $C = 1$  at epochs of 47 and 124, respectively, then the learning parameters become stable. In this case, MemPo-Learn reaches  $C = 1$  after 47 epochs which amounts to  $47 \times 0.006 = 0.282$  s, and PBSNLR reach  $C = 1$  after 124 epochs which amounts to  $124 \times 0.0058 = 0.719$  s. The learning performance of MemPo-Learn and PBSNLR drops significantly with a time step of 0.01 ms. For instance, the  $C$  value for MemPo-Learn (0.01 ms) at epoch 47 is 5% lower than that obtained using a time step of 1 ms, and the  $C$  value for PBSNLR (0.01 ms) at epoch 124 is 6% lower than that obtained using a time step of 1 ms. Moreover, both MemPo-Learn and PBSNLR are unable to reach  $C = 1$  even if it is allowed to run for as long as 1000 learning epochs. Therefore, with a small time step, MemPo-Learn and PBSNLR's learning accuracy drops and the required number of learning epochs as well as the learning time increase. On the other hand, the learning efficiency of MemPo-Learn combined with SSTS is significantly improved. For example, MemPo-Learn combined with SSTS can reach a high learning

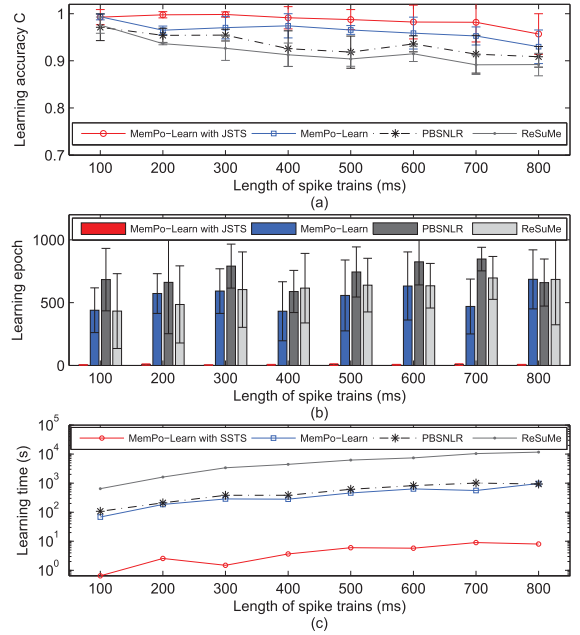


Fig. 9. Comparison of the learning performance against gradually increased length of the desired spike trains. Time step is set to 0.01 ms.

accuracy ( $C$  close to 1) after only five epochs which amounts to learning time of  $5 \times 0.93 = 4.65$  s.

2) *Effect of the Spike Trains Length*: In these simulations, the number of synaptic inputs is 400. Every input spike train and the desired output spike train are generated according to a homogeneous Poisson process with rates of 10 and 100 Hz, respectively. The length of the desired output spike trains varies from 100 to 800 ms with an interval of 100 ms, and the time step is set to 0.01 ms.

As shown in Fig. 9, MemPo-Learn combined with SSTS achieves better learning performance than MemPo-Learn, PBSNLR, and ReSuMe. The learning accuracy of MemPo-Learn combined SSTS is higher than that of the other methods when the spike train length varies from 100 to 800 ms. For example, when the length of the desired spike train is 700 ms, the average learning accuracy of MemPo-Learn combined with SSTS is about 0.98, while the learning accuracies of MemPo-Learn, PBSNLR, and ReSuMe are about 0.95, 0.93, and 0.91, respectively. Moreover, the required learning epochs and learning time for MemPo-Learn combined with SSTS are much lower. For example, when the length of spike trains is set to 700 ms, MemPo-Learn takes almost 60 times longer than MemPo-Learn combined with SSTS to complete learning and PBSNLR takes almost 110 times longer. Thus, the learning efficiency of MemPo-Learn combined with SSTS is clearly improved in comparison with MemPo-Learn without SSTS and PBSNLR.

3) *Effect of the Number of the Synaptic Inputs*: In the following experiments, the performance of MemPo-Learn combined with SSTS is evaluated for various values of the number of the synaptic inputs 500, 400, 300, 200, 100, and 50. In these simulations, the length of spike train is 400 ms, and the time step is 0.01 ms. Every input spike train and the desired



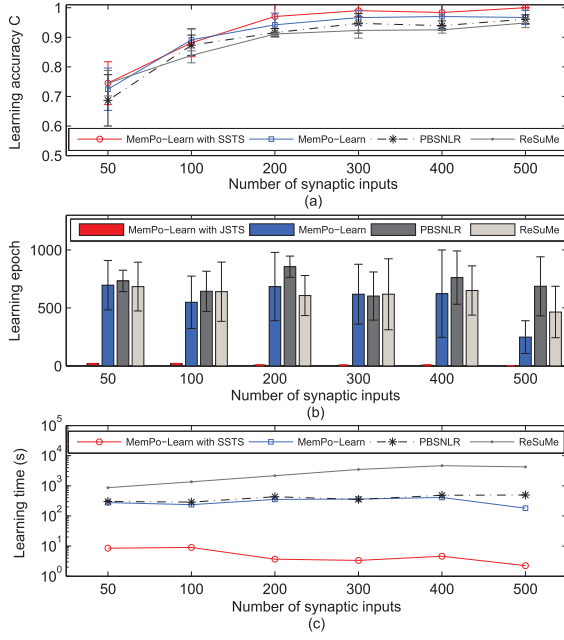


Fig. 10. Comparison of the learning performance against gradually increased number of the synaptic input. Time step is set to 0.01 ms.

output spike train are generated according to a homogeneous Poisson process with rates of 10 and 100 Hz, respectively.

In Fig. 10, when the number of synaptic inputs varies from 50 to 100, we can see that the learning accuracy curves of different methods are comparable. However, when the number of synaptic input exceeds 200, the learning accuracy of MemPo-Learn combined with SSTS is overall higher than that of other methods. When the number of synaptic inputs is 500, the learning accuracy of MemPo-Learn combined with SSTS is very high (measure  $C$  is almost equal to 1), while the highest learning accuracy of other three methods is only 0.96. As for learning efficiency, the proposed MemPo-Learn combined with SSTS remarkably outperforms both MemPo-Learn, PBSNLR, and ReSuMe in terms of both required number of epochs as well as required learning time. This is clearly reflected in a dramatic reduction in both the required number of epochs, which is illustrated by red bars in Fig. 10(b), and the required learning time which is illustrated by red curve in Fig. 10(c). For example, when the number of synaptic inputs is 500, the required number of learning epochs for MemPo-Learn, PBSNLR, and ReSuMe are 248, 686, and 468, respectively. However, MemPo-Learn combined with SSTS requires only about three learning epochs which clearly is an impressive improvement.

4) *Effect of the Firing Rate of the Spike Trains*: In the following experiments, we aim to evaluate the effect of the firing rate of the spike trains. The firing rates of the input spike trains ( $F_{in}$ ) vary from 6 to 18 Hz with an interval of 4 Hz. The firing rates of the desired output spike trains ( $F_{out}$ ) vary from 20 to 160 Hz with an interval of 20 Hz. The number of the synaptic input is 400, and the length of the spike trains is 400 ms. The learning is continued for 1000 learning epochs, and the maximum obtained learning accuracy is reported in Fig. 11.

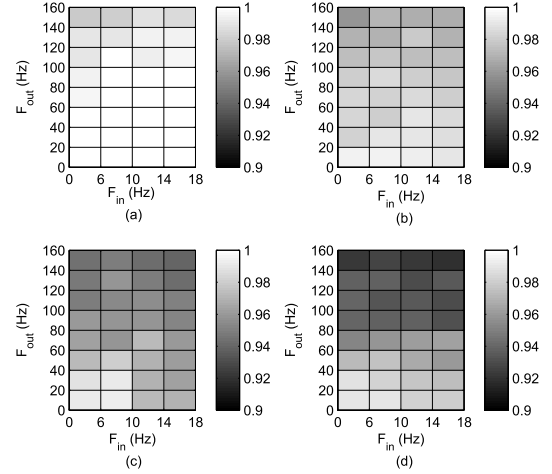


Fig. 11. Comparison of the learning performance against gradually increased firing rates of the input and desired output spike trains. Time step is set to 0.01 ms.

In Fig. 11, all of the learning methods reach their highest learning accuracy when  $F_{in} = 6$  Hz and  $F_{out} = 20$  Hz. When  $F_{in} = 18$  Hz and  $F_{out} = 160$  Hz, all of these three methods reach their lowest values of performance. On the other hand, the area in which MemPo-Learn combined with SSTS achieves high learning accuracy is larger than that of other methods.

### C. Robustness to Noise

In this section, we investigate the noise robustness of the neuron trained by different learning methods. A neuron with 400 synaptic inputs is trained to output the desired spike train with a length of 800 ms. Every input spike train and the desired output spike train are Poisson spike trains with rates 10 and 100 Hz, respectively. After training, the reliability of the target recall is tested against two noise cases: 1) background noise on the membrane potential and 2) input jittering noise.

1) *Robustness to Membrane Potential Noise*: In this case, background membrane potential noise is considered as the noise source. After training, the trained neuron is subjected to simulated background Gaussian white noise. The mean value of the added noise is 0, and its variance  $\sigma_b$  is systematically increased within the range of [0.03, 0.33] mV. For each value of  $\sigma_b$ , 20 experiments are carried out. A correlation measure  $C$  [41] of a distance between the desired and actual output spike trains is calculated. The experimental results are shown in Fig. 12.

Fig. 12 shows that the correlation  $C$  of all the three methods is high when the intensity of noise is small. However, it decreases when the noise intensity is gradually increased. The correlation  $C$  curves of MemPo-Learn, PBSNLR, and ReSuMe decline relatively early and quickly. However, the correlation  $C$  of the neuron trained by R-MemPo-Learn always maintains high values with the increase of  $\sigma_b$ , and is significantly higher than other methods. These results confirm that the neuron trained by R-MemPo-Learn is significantly less sensitive to noise.

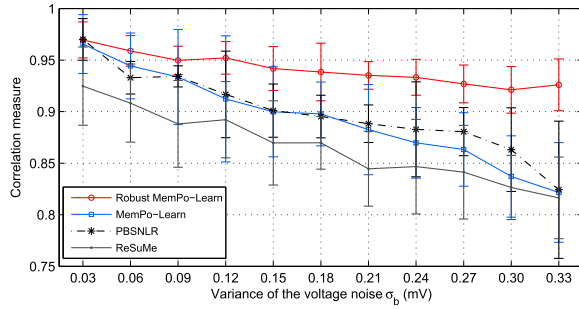


Fig. 12. Antinoise capability of different learning algorithms against background voltage noise.

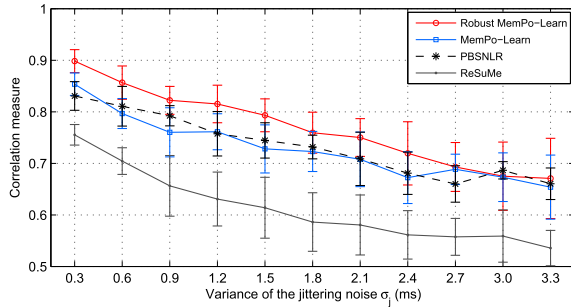


Fig. 13. Antinoise capability of different learning algorithms against jittering noise.

2) *Robustness to Input Spike Time Jitter*: In this case, input jittering noise is considered as the noise source. After learning, we jitter the input spike times. The jitter intervals are randomly drawn from a Gaussian distribution with mean 0 and variance  $\sigma_j \in [0.3, 3.3]$  ms. In addition, some spikes are randomly removed (with a probability of 0.05) or added (at the times generated by a 1-Hz homogeneous Poisson process). The resulting plots of  $C$  are presented in Fig. 13.

As shown in Fig. 13, an increase in the noise intensity results in a decreased correlation between the desired and the actual output spike trains. The measure  $C$  scored by MemPo-Learn, PBSNLR, and ReSuMe drops more sharply than that of the robust MemPo-Learn. That is, neuron trained by the robust MemPo-Learn is clearly more robust to noise than the neuron trained by other methods.

#### D. Effect of Learning Parameters $p$ and $s_{len}$

Two major learning parameters involved in our method are  $p$  and  $s_{len}$ . In this section, we aim to investigate the effect of these parameters on the learning performance.

1) *Effect of Parameter  $p$* : The role of  $p$  is to make the membrane potential below the firing threshold at undesired firing times  $N_{id}$ . It determines the magnitude of modification on the synaptic weights at  $N_{id}$ . To look into the effect of  $p$ , we conduct several experiments with a time step of 1 ms. A neuron with 200 synaptic inputs is trained to emit a desired sequence of spikes with a length of 500 ms. Every input spike train and desired output spike train are generated randomly according to a homogeneous Poisson process with rate  $r = 10$  and 100 Hz, respectively. Here we choose  $p = 0.01, 0.05, 0.1, 1, 3, 6$ , and 9 mV. If the number

TABLE III  
IMPACT OF  $p$  ON THE LEARNING PERFORMANCE

$p$	0.01	0.05	0.1	1	3	6	9
$C$	1	1	1	1	1	1	Failure
Epochs	87	79	65	92	119	189	-

TABLE IV  
IMPACT OF  $s_{len}$  ON THE LEARNING PERFORMANCE

$s_{len}$	1 ms	2 ms	3 ms	4 ms	5 ms	6 ms	7 ms
$C$	1	1	1	1	1	1	1
Epochs	5	3	3	2	1	1	1

of learning epochs exceeds 500, we regard this training as a failure. The accuracy value  $C$  and the number of epochs needed to reach  $C$  are given in Table III.

Table III reveals that  $C$  equals 1, though the values of  $p$  range from 0.01 to 6 mV, and it means that MemPo-Learn has the advantage of parameter insensitivity. A larger  $p$  can result in a faster learning speed, but when  $p$  is increased above a critical value (e.g., 0.1 mV in our experiments), the learning will slow down or even fail. A smaller  $p$  means a smaller adjustment for synaptic weights, which results in more learning epochs. Weight updating inevitably changes not only the membrane potential at current time but also the membrane potential at other times, so it will affect the precious learning results. A larger  $p$  has a bigger interference on the previous learning results, and it results in more learning epochs. Moreover, if the value of  $p$  is too large, the learning process will fall into overadjustment or even fail.

2) *Effect of Parameter  $s_{len}$* : We have proposed the SSTS strategy to resolve the problems of overadjustment and inadequate learning chances at  $t_d$ . The most important parameter in SSTS is  $s_{len}$ . To investigate the effect of  $s_{len}$ , we conduct several experiments with a setup similar to Table III but with a time step of 0.01 ms and the neuron is trained by MemPo-Learn combined with SSTS. The parameter  $s_{len}$  is varied between 1 and 7 with a unit step. If the number of learning epochs exceeds 10, we regard this training as a failure. The value of accuracy  $C$  and the number of epochs needed to reach it are given in Table IV.

As given in Table IV,  $s_{len}$  has little effect on accuracy since the values of  $C$  are all equal to 1. A large  $s_{len}$  results in a smaller number of learning epochs, but when  $s_{len}$  is increased above a critical value (e.g., 5 ms in our experiment), the number of learning epochs keeps the same. For SSTS, the number of subepochs  $n = s_{len}/t_s$ , a larger  $s_{len}$  means more subepochs. According to SSTS, the times of desired output spikes are added into each subepoch. So a larger  $s_{len}$  will give more learning chances to  $t_d$ . When  $s_{len}$  is large enough,  $t_d$  has adequate learning chances to make the membrane potential equal to the threshold. In this case, if we continue to increase the value of  $s_{len}$ , it may not do much to raise the learning speed.

#### E. Classification

Spiking neural networks have been applied to various classification tasks [44]–[53]. In most cases, the spiking

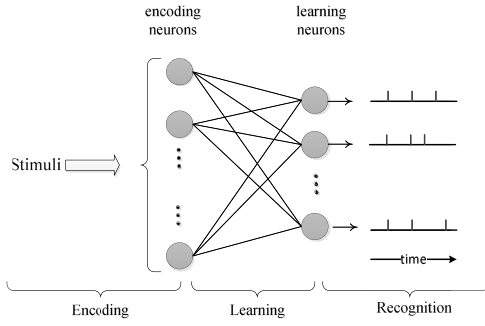


Fig. 14. General structure and information process of the SNN. It contains three functional parts: encoding, learning, and recognition.

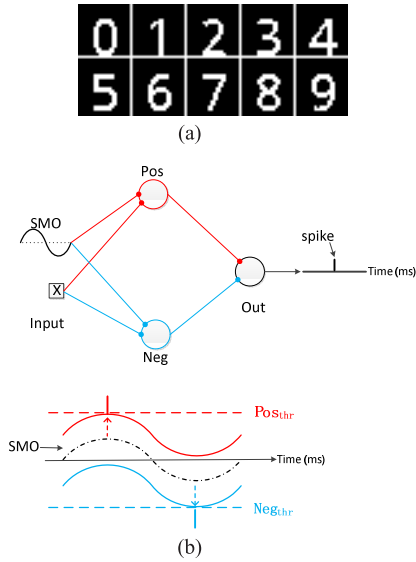


Fig. 15. (a) OCR samples. (b) Illustration of the phase encoding schema. The encoding schema is adapted from [51] and [52].

neuron-based classifiers make decisions using single spike only or by using analog or binary signal representation. Here, we illustrate this ability of our method in a classification task proposed by Yu *et al.* [51], [52], where a spiking neuron-based computational model (as shown in Fig. 14) is proposed for spike sequences classification. In this experiment, we adopt this computational model to evaluate the capability of the proposed learning method in practical applications, including optical character recognition (OCR) and sound event classification.

1) *Optical Character Recognition*: An OCR task is considered in this experiment where images of digits 0–9 are used. Each image has a size of  $20 \times 20$  black/white pixels. Sample images are shown in Fig. 15(a). In the encoding part, a phase encoding method is used to convert the images into spatiotemporal spike patterns [51], [52]. The mechanism of the phase encoding is shown in Fig. 15(b). Each encoding unit consists of a positive neuron (Pos), a negative neuron (Neg), and an output neuron. Each encoding neuron is assigned to a pixel and a subthreshold membrane potential oscillation. (For more details about phase coding, see [51], [52].)

The learning part of the spiking neural network is composed of one layer of 10 spiking neurons, with each learning neuron

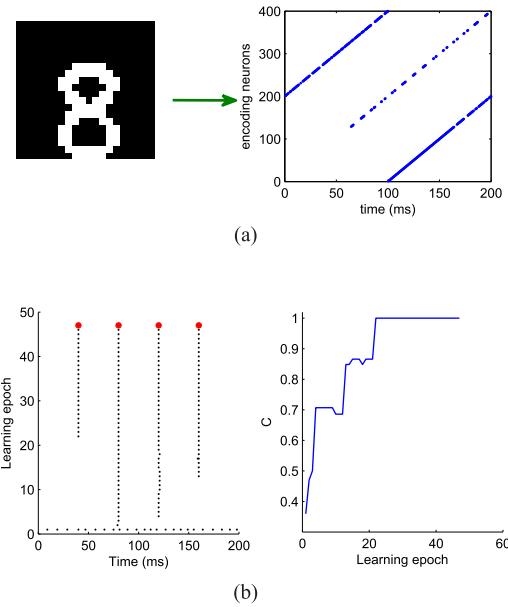


Fig. 16. Learning performance of the proposed method on the OCR recognition task. (a) Phase encoding results of a given image sample. Each dot denotes a spike. (b) Output spikes of the learning neuron corresponding to digit “8.”

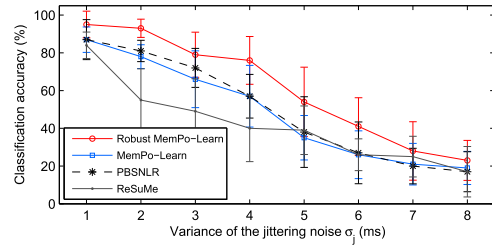


Fig. 17. Robustness of different methods against the jittering noise.

corresponding to one category. Each learning neuron is trained to fire a desired sequence of spikes ([40, 80, 120, 160] ms) when a corresponding pattern is present, and not to spike when other patterns are presented.

In the recognition part, the relative confidence criterion is used for decision making, where the input pattern will be decided by one of the neurons that generates the most similar spike train to the target spike train.

After phase coding, different images can be converted into corresponding spatiotemporal spike patterns. Fig. 16(a) demonstrates an encoding result of a given image sample, in which the output spikes are sparsely distributed over the encoding time window. To further illustrate the learning process of the MemPo-Learn rule, Fig. 16(b) shows the learning performance of digit “8.” The learning neuron corresponding to digit “8” can successfully produce the desired spike train after about 25 learning epochs.

To study the noise robustness of the proposed method on classification, after learning, the reliability of the target recall is tested against two noise cases: 1) background noise on the membrane potential and 2) input jittering noise. Figs. 17 and 18 show the classification accuracies of different learning algorithms against jittering noise and background voltage noise, respectively.

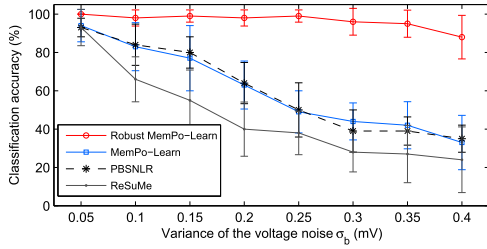


Fig. 18. Robustness of different methods against the background voltage noise.

As can be seen in Fig. 17, the performance of all four methods decreases with increasing noise level. While both PBSNLR and MemPo-Learn without the noise robustness strategy show comparable response, the robust version of MemPo-Learn remarkably outperforms both MemPo-Learn without the robustness strategy and PBSNLR. In addition, the robustness of ReSuMe is relatively lower than that of other three methods.

In Fig. 18, we can see that when the intensity of noise is small, the classification accuracy of all four methods is very high and comparable. The classification accuracy of all four methods decreases with increasing noise level. However, the classification accuracy of PBSNLR, ReSuMe, and MemPo-Learn without the robustness strategy decreases more sharply than the robust MemPo-Learn. The computational model trained by the robust MemPo-Learn rule can maintain a high classification accuracy ( $\sim 90\%$ ) even when the voltage noise reaches a considerably high level ( $\sim 0.4$  mV).

2) *Sound Event Classification*: In this section, we carry out experiments to show the performance of our proposed learning method on a sound recognition task. A total of 10 sounds are selected from the Real World Computing Partnership [54] sound scene database in real acoustic environments. The selected categories cover a wide range of sound events, including horn, bells5, bottle1, buzzer, cymbals, kara, metal15, phone4, whistle1, and whistle3. For each event, 40 files are randomly selected as training samples and another 40 files are selected for testing samples. After training, the average classification accuracy for each method is reported in clean and at 20, 10, and 0-dB signal-to-noise ratio (SNR) for the “speech babble” noise environment, taken from the NOISEX’92 database [55].

The encoding method proposed in [56] is used to convert the sound events into spatiotemporal spike patterns. According to the encoding method, the sound is converted from its original domain to a representation in the frequency domain by fast Fourier transform over several windows. Then, a 1-D order filter is used in the feature extraction stage to select the local maximum in the power spectrum as a keypoint, followed by the temporal coding scheme to produce the output spatiotemporal spike patterns. Fig. 19 demonstrates an encoding result of a bottle sound in both clean and 10-dB noise. (For more details about the coding method, see [56].)

The learning part of the spiking neural network is composed of one layer of 10 spiking neurons, with each learning neuron corresponding to one category. Each neuron is trained to

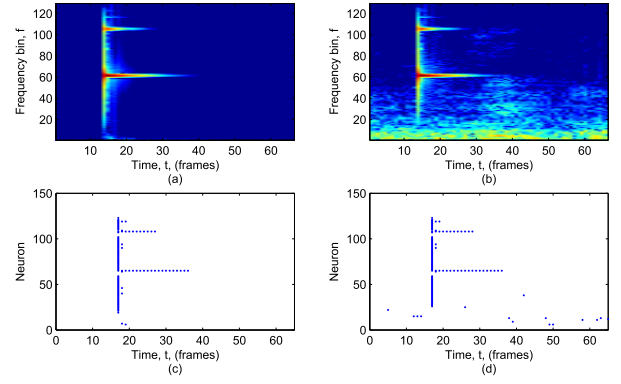


Fig. 19. Examples of encoded spatiotemporal spike patterns. (a) and (b) Bottle sound in clean and 10-dB noise condition, with the corresponding encoded spike trains shown below.

TABLE V  
CLASSIFICATION ACCURACY OF DIFFERENT  
METHODS FOR THE SOUND EVENT TASK

Methods	Clean	20dB	10dB	0dB	Average
R-MemPo-Learn	97.8%	97.1%	<b>96.4%</b>	<b>91.1%</b>	<b>95.6%</b>
MemPo-Learn	97.1%	96.2%	95.2%	88.7%	94.3%
PBSNLR	96.9%	96.3%	95.8%	87.2%	94.0%
ReSuMe	95.3%	92.3%	90.3%	85.2%	90.7%
DNN-5 layers	97.5%	97.2%	87.5%	20.2%	75.6%
CNN-5 layers	<b>98.7%</b>	<b>97.3%</b>	91.52%	38.5%	81.5%
CNN-7 layers	97.2%	95.2%	92.7%	25.7%	77.7%

fire a spike when a corresponding pattern is present, and the desired firing time is when the postsynaptic membrane potential reaches its maximum value. When other patterns are presented, the membrane potential of the learning neuron is trained to below the firing threshold.

In the recognition part, the input pattern will be decided by one of the neurons that generates the most similar spike to the desired spike time. In addition, if all learning neurons remain silent, the learning neuron with the strongest activation state represents the class association. Table V gives the sound event classification performance of different methods.

The experimental results are presented in Table V. It can be seen that the proposed robust MemPo-Learn method performs well for each of the noise conditions, achieving an average accuracy of 95.6%. It can also maintain an accuracy of over 91% in the challenging 0-dB SNR condition. The results also show that the classification accuracy of convolutional neural network (CNN) and deep neural network (DNN) is high under clean and low-noise environment, while the performance decreases dramatically with the increase of the noise level. For example, the CNN-5 model can achieve a classification accuracy of 98.7% under clean condition, while the accuracy decreases to 38.5% under the 0-dB SNR condition. Therefore, the robustness of the proposed method is better than the traditional neural networks.

## VI. CONCLUSION

Analysis of the experiments revealed that the learning performance of MemPo-Learn is considerably better than that



of ReSuMe in terms of the learning accuracy and efficiency. The difference in the learning performance between MemPo-Learn and ReSuMe is due to the difference in the training mechanisms. MemPo-Learn is a membrane potential-driven method, using the postsynaptic membrane potential rather than postsynaptic spike times as the relevant signal for synaptic changes. In this way, the adjustment of synaptic weights is direct, and it will decrease the difficulty and complexity of the training process. In addition, compared to PBSNLR, MemPo-Learn has obvious advantage in terms of learning efficiency. This is mainly attributed to the use of the gradient descent in MemPo-Learn where the magnitude of the weight changes is determined by the learning rate and the difference between the desired and the actual membrane potentials, unlike PBSNLR where weight adjustment is based on the learning rate only.

A small time step can be much closer to continuous time and is extremely important for real-time applications of SNNs. However, using a small time step learning is more difficult and time consuming. Hence, SSTS was proposed to improve the efficiency of small time step-based learning. SSTS divides one learning epoch into many subepochs, in each subepoch, SSTS consists of two main operations: 1) all desired output times are added into monitor time points to resolve inadequate learning for  $t_d$  and 2) jumping to monitor the membrane potential in each subepoch to resolve overadjustment at  $N_{td}$ . By using SSTS, we not only overcome overadjustment and inadequate learning for  $t_d$ , but we also improve the learning efficiency significantly.

In the future work, we will explore how to further extend MemPo-Learn to multiple layer deep networks of spiking neurons. It is expected that such an approach would improve the application range and memory capacity of spiking neurons. Another interesting future direction is to search for efficient and biological plausible input and output encoding methods for multiple spikes that can further improve the application performance. Another interesting idea to pursue in the future is to look at how possibly the information theory can be used to derive novel analytic measures of performance and predict network performance based on the quality of input.

## APPENDIX EXPERIMENTAL DETAILS

Unless otherwise stated, our experiments run on MATLAB 7.12.0 on a quad-core system with 16-GB RAM in Windows environment. All parameters of our algorithm are empirical values. For traditional algorithms, the parameter value scopes provided by their corresponding references are employed in our simulations, and many different values in these scopes are tested to find the one achieving the highest accuracy. During the learning process, MemPo-Learn uses the storage space in exchange for substantial savings in calculation time. All the PSPs induced by every synapse at different time steps need to be calculated and stored before training. Unless the learning neuron can output the target output spikes precisely, the experiments will stop at the upper limit of 1000 learning epochs. In all of the experiments, the value of the neuron model is set as:  $\vartheta = 1$  mV,  $\lambda = 2$ ,  $\tau = 7$  ms, and  $\tau_R = 5$  ms.

The experiments of CNN and DNN are run on Python 3.6.1 with TensorFlow 1.3.0 on a quad-core system with 16-GB RAM in Windows environment, and the CNN and DNN are trained on spectrogram using short-time Fourier transform (STFT). The STFT is performed with 50 filters and a 16-kHz sampling frequency. The audio signal is down sampling into 50 frames with 50% overlap. The DNN construct consists of five fully connected layers with the size set as 1024-512-256-64-10. The CNN-5 model consists of one convolutional layer, one pooling layer, followed by three fully connected layers. The CNN-7 model consists of two convolutional layers, and each convolutional layer is followed by a pooling layer. Similar to the CNN-5 model, the CNN-7 model is equipped with three fully connected layers.

## REFERENCES

- [1] A. Mohammed, S. Schliebs, S. Matsuda, and N. Kasabov, "Training spiking neural networks to associate spatio-temporal input-output spike patterns," *Neurocomputing*, vol. 107, pp. 3–10, May 2013.
- [2] P. A. Cariani, "Temporal codes and computations for sensory representation and scene analysis," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1100–1111, Sep. 2004.
- [3] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, no. 6535, pp. 33–36, 1995.
- [4] J. Gautrais and S. Thorpe, "Rate coding versus temporal order coding: A theoretical approach," *BioSystems*, vol. 48, nos. 1–3, pp. 57–65, 1998.
- [5] M. J. Berry and M. Meister, "Refractoriness and neural precision," *J. Neurosci.*, vol. 18, no. 6, pp. 2200–2211, Mar. 1998.
- [6] V. J. Uzzell and E. J. Chichilnisky, "Precision of spike trains in primate retinal ganglion cells," *J. Neurophysiol.*, vol. 92, no. 2, pp. 780–789, 2004.
- [7] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, Feb. 2008.
- [8] P. Reinagel and R. C. Reid, "Temporal coding of visual information in the thalamus," *J. Neurosci.*, vol. 20, no. 14, pp. 5392–5400, 2000.
- [9] W. Bair and C. Koch, "Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey," *Neural Comput.*, vol. 8, no. 6, pp. 1185–1202, 1996.
- [10] W. Wang, B. Subagdjia, A.-H. Tan, and J. A. Starzyk, "Neural modeling of episodic memory: Encoding, retrieval, and forgetting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 10, pp. 1574–1586, Oct. 2012.
- [11] V. A. Nguyen, J. A. Starzyk, W.-B. Goh, and D. Jachyra, "Neural network structure for spatio-temporal long-term memory," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 971–983, Jun. 2012.
- [12] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, Aug. 2002.
- [13] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 295–308, Aug. 2009.
- [14] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [15] W. Maass, "Fast sigmoidal networks via spiking neurons," *Neural Comput.*, vol. 9, no. 2, pp. 279–304, 1997.
- [16] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 211–217.
- [17] R. Kempter, W. Gerstner, and J. L. van Hemmen, "Spike-based compared to rate-based Hebbian learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 11, 1999, pp. 125–131.
- [18] A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature Neurosci.*, vol. 2, no. 11, pp. 947–957, 1999.
- [19] E. I. Knudsen, "Supervised learning in the brain," *J. Neurosci.*, vol. 14, no. 7, pp. 3985–3997, 1994.
- [20] H. Qu, X. Xie, Y. Liu, M. Zhang, and L. Lu, "Improved perception-based spiking neuron learning rule for real-time user authentication," *Neurocomputing*, vol. 151, pp. 310–318, Mar. 2015.
- [21] W. T. Thach, "On the specific role of the cerebellum in motor learning and cognition: Clues from PET activation and lesion studies in man," *Behavioral Brain Sci.*, vol. 19, no. 3, pp. 411–433, 1996.

- [22] M. Ito, "Mechanisms of motor learning in the cerebellum," *Brain Res.*, vol. 886, nos. 1–2, pp. 237–245, Dec. 2000.
- [23] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, 2010.
- [24] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "DL-ReSuMe: A delay learning-based remote supervised method for spiking neurons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3137–3149, Dec. 2015.
- [25] S. M. Bohte, J. N. Kok, and H. L. Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, 2002.
- [26] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Netw.*, vol. 43, pp. 99–113, Jul. 2013.
- [27] R. V. Florian, "The chronotron: A neuron that learns to fire temporally precise spike patterns," *PLoS ONE*, vol. 7, no. 8, p. e40233, 2012.
- [28] A. Mohammed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *Int. J. Neural Syst.*, vol. 22, no. 4, p. 1250012, 2012.
- [29] J. D. Victor and K. P. Purpura, "Metric-space analysis of spike trains: Theory, algorithms and application," *Netw., Comput. Neural Syst.*, vol. 8, no. 2, pp. 127–164, 1997.
- [30] M. C. W. van Rossum, "A novel spike distance," *Neural Comput.*, vol. 13, no. 4, pp. 751–763, Apr. 2001.
- [31] R. Gütiğ and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neurosci.*, vol. 9, no. 3, pp. 420–428, Feb. 2006.
- [32] Y. Xu, X. Zeng, and S. Zhong, "A new supervised learning algorithm for spiking neurons," *Neural Comput.*, vol. 25, no. 6, pp. 1472–1511, 2013.
- [33] R.-M. Memmesheimer, R. Rubin, B. P. Ölveczky, and H. Sompolinsky, "Learning precisely timed spikes," *Neuron*, vol. 82, no. 4, pp. 925–938, 2014.
- [34] R. Gütiğ, "Spiking neurons can discover predictive features by aggregate-label learning," *Science*, vol. 351, no. 6277, p. aab4113, 2016.
- [35] C. Albers, M. Westkott, and K. Pawelzik, "Learning of precise spike times with homeostatic membrane potential dependent synaptic plasticity," *PLoS ONE*, vol. 11, no. 2, p. e0148948, 2016.
- [36] A. Zador, "Spikes: Exploring the neural code," *Science*, vol. 277, no. 5327, pp. 772–773, 1997.
- [37] E. Schneidman, I. Segev, and N. Tishby, "Noise and information in neural codes," Ph.D. dissertation, Dept. Neurobiol., Hebrew Univ., Jerusalem, Israel, 2001.
- [38] M. van Rossum, B. J. O'Brien, and R. G. Smith, "Effects of noise on the spike timing precision of retinal ganglion cells," *J. Neurophysiol.*, vol. 89, pp. 2406–2419, May 2003.
- [39] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952.
- [40] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.
- [41] S. Schreiber, J. M. Fellous, D. Whitmer, P. Tiesinga, and T. J. Sejnowski, "A new correlation-based measure of spike timing reliability," *Neurocomputing*, vols. 52–54, pp. 925–931, Jun. 2003.
- [42] F. Ponulak, "Analysis of the ReSuMe learning process for spiking neural networks," *Int. J. Appl. Math. Comput. Sci.*, vol. 18, no. 2, pp. 117–127, 2008.
- [43] A. Kasiński and F. Ponulak, "Comparison of supervised learning methods for spike time coding in spiking neural networks," *Int. J. Appl. Math. Comput. Sci.*, vol. 16, no. 1, pp. 101–113, 2006.
- [44] J. A. Wall, L. McDaid, L. P. Maguire, and T. M. McGinnity, "Spiking neural network model of sound localization using the interaural intensity difference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 574–586, Apr. 2012.
- [45] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1539–1552, Oct. 2013.
- [46] H. Qu, S. X. Yang, A. R. Willms, and Z. Yi, "Real-time robot path planning based on a modified pulse-coupled neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 11, pp. 1724–1739, Nov. 2009.
- [47] J. Hu, H. Tang, K. C. Tan, H. Li, and L. Shi, "A spike-timing-based integrated model for pattern recognition," *Neural Comput.*, vol. 25, no. 2, pp. 450–472, 2013.
- [48] M. Zhang, H. Qu, X. Xie, and J. Kurths, "Supervised learning in spiking neural networks with noise-threshold," *Neurocomputing*, vol. 219, pp. 333–349, Jan. 2017.
- [49] J. Hu, H. Tang, K. C. Tan, and H. Li, "How the brain formulates memory: A spatio-temporal model," *IEEE Comput. Intell. Mag.*, vol. 11, no. 2, pp. 56–68, May 2016.
- [50] H. Qu, Z. Yi, and S. X. Yang, "Efficient shortest-path-tree computation in network routing based on pulse-coupled neural networks," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 995–1010, Jun. 2013.
- [51] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns," *PLoS ONE*, vol. 8, no. 11, p. e78318, 2013.
- [52] Q. Yu, R. Yan, H. Tang, K. C. Tan, and H. Li, "A spiking neural network system for robust sequence recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 621–635, Mar. 2016.
- [53] M. Zhang, H. Qu, A. Belatreche, and X. Xie, "EMPD: An efficient membrane potential driven supervised learning algorithm for spiking neurons," *IEEE Trans. Cogn. Develop. Syst.*, to be published, doi: 10.1109/TCDS.2017.2651943.2017.
- [54] Real World Computing Partnership. (2000). *RWCP Sound Scene Database in Real Acoustic Environments*. [Online]. Available: <http://tosa.mri.co.jp/sounddb/indexe.htm>
- [55] A. Varga and H. J. M. Steeneken, "Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Commun.*, vol. 12, no. 3, pp. 247–251, Jul. 1993.
- [56] R. Xiao, R. Yan, H. Tang, and K. C. Tan, "A spiking neural network model for sound recognition," in *Proc. Int. Conf. Cognit. Syst. Signal Process.*, 2016, pp. 584–594.



**Malu Zhang** is currently pursuing the Ph.D. degree from the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include neural networks, intelligent computation, deep learning, and optimization.



**Hong Qu** (M'09) received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2006.

From 2007 to 2008, he was a Post-Doctoral Fellow with the Advanced Robotics and Intelligent Systems Lab, School of Engineering, University of Guelph, Guelph, ON, Canada. From 2014 to 2015, he was a Visiting Scholar with the Potsdam Institute for Climate Impact Research, Potsdam, Germany, and the Humboldt University of Berlin, Berlin, Germany.

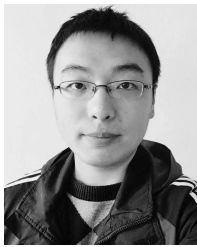
He is currently a Professor with the Computational Intelligence Laboratory, School of Computer Science and Engineering, University of Electronic Science and Technology of China. His current research interests include neural networks, machine learning, and big data.



**Ammar Belatreche** (M'09) received the Ph.D. degree in computer science from Ulster University, Derry, U.K.

He was a Lecturer in computer science with the School of Computing and Intelligent Systems, Ulster University. He is currently a Senior Lecturer in computer science with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. His current research interests include bioinspired adaptive systems, machine learning, pattern recognition, data analytics, capital market engineering, and image processing and understanding.

Dr. Belatreche is a Fellow of the Higher Education Academy and an Associate Editor of *Neurocomputing*. He has served as a Program Committee Member and a reviewer for several international conferences and journals.



**Yi Chen** is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include neural networks, intelligent computation, deep learning, and optimization.



**Zhang Yi** (F'16) received the Ph.D. degree in mathematics from the Institute of Mathematics, Chinese Academy of Science, Beijing, China, in 1994.

He is currently a Professor with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China. He has co-authored three books: *Convergence Analysis of Recurrent Neural Networks* (Kluwer Academic Publishers, 2004), *Neural Networks: Computational Models and Applications* (Springer, 2007), and *Subspace Learning of Neural Networks* (CRC Press,

2010). His current research interests include neural networks and big data.

Dr. Yi was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2009 to 2012 and the IEEE TRANSACTIONS ON CYBERNETICS in 2014.