Efficient Training of Supervised Spiking Neural Network via Accurate Synaptic-Efficiency Adjustment Method

Xiurui Xie, Hong Qu, *Member, IEEE*, Zhang Yi, *Fellow, IEEE*, and Jürgen Kurths

***Abstract*— The spiking neural network (SNN) is the third generation of neural networks and performs remarkably well in cognitive tasks, such as pattern recognition. The temporal neural encode mechanism found in biological hippocampus enables SNN to possess more powerful computation capability than networks with other encoding schemes. However, this temporal encoding approach requires neurons to process information serially on time, which reduces learning efficiency significantly. To keep the powerful computation capability of the temporal encoding mechanism and to overcome its low efficiency in the training of SNNs, a new training algorithm, the accurate synaptic-efficiency adjustment method is proposed in this paper. Inspired by the selective attention mechanism of the primate visual system, our algorithm selects only the target spike time as attention areas, and ignores voltage states of the untarget ones, resulting in a significant reduction of training time. Besides, our algorithm employs a cost function based on the voltage difference between the potential of the output neuron and the firing threshold of the SNN, instead of the traditional precise firing time distance. A normalized spike-timing-dependent-plasticity learning window is applied to assigning this error to different synapses for instructing their training. Comprehensive simulations are conducted to investigate the learning properties of our algorithm, with input neurons emitting both single spike and multiple spikes. Simulation results indicate that our algorithm possesses higher learning performance than the existing other methods and achieves the state-of-the-art efficiency in the training of SNN.**

***Index Terms*— Pattern recognition, selective attention mechanism, spiking neural network (SNN), supervised learning.**

I. Introduction

ARTIFICIAL neural networks process information in a biomimetic way, and different generations of neural networks have different bionic extents. Most traditional neural networks belonging to the second generation represent real-valued analog data by the average firing rate of neurons. However, there is substantial evidence that in biological neural systems there exist fast computations that are very likely based on spike firing events [1]–[3]. Motivated by the spike emitting mechanism found in hippocampus electrophysiology experiments, spiking neural network (SNN) with temporal coding is proposed to simulate this mechanism of biological neurons, which has been proved computationally more powerful than the networks with rate coding [4], [5]. It gives a high biomimetic model of biologic neural activity [6], [7], and has been successfully simulated and applied to various artificial intelligence tasks [8]–[12].

Supervised learning is an important biomimetic concept in recognizing processing of neural networks. Compared with other learning mechanism, a supervised learning rule could potentially improve learning speed with the help of an instructor signal. However, the existing supervised training methods employing temporal coding are extremely inefficient. How to train the network efficiently has become the critical problem in the development of SNN [13].

According to the measures of error reduction, the existing supervised learning algorithms for spiking neurons can be classified into two types. The first type reduces errors by defining a cost function and minimizes it by the gradient descent rule. This is a typical mathematical method but it is often inefficient. The second type instructs training by learning windows. Weight changes for some learning windows are usually realized according to the biomimetic mechanism, such as the Hebbian rule [14] and its derivant spike-timing-dependent plasticity (STDP) rule [15], where a synaptic weight is strengthened if the presynaptic neuron emits spikes before the postsynaptic neuron and weakened conversely. Some typical algorithms of these two types are introduced in the following.

A lot of supervised algorithms for SNNs have been proposed recently [13], [16]. SpikeProp is the best well known one [17]. It defines the cost function by the distance of the actual firing time and the target ones, and minimizes it by the gradient descent rule. It is then improved in [18] and [19] to emit multispikes. Tempotron [20] is another gradient descent training algorithm. Employing the distance of the output neuron's voltage and the firing threshold as the cost function, the Tempotron can complete training more efficient than the other algorithms, but it can only be applied to binary

Manuscript received February 6, 2015; revised January 30, 2016; accepted March 5, 2016. Date of publication March 30, 2016; date of current version May 15, 2017. This work was supported by the National Science Foundation of China under Grant 61273308, Grant 61432012, and Grant 61573081.

X. Xie is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: xiexiurui@126.com).

H. Qu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China, and also with the Potsdam Institute for Climate Impact Research, Potsdam 14473, Germany (e-mail: hongqu@uestc.edu.cn).

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

J. Kurths is with the Potsdam Institute for Climate Impact Research, Potsdam 14473, Germany, and also with the Department of Physics, Humboldt University of Berlin, Berlin 12489, Germany (e-mail: kurths@pik-potsdam.de).

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.2016.2541339

classification problems. These algorithms can simulate the spiking learning processes availably, but they are too inefficient to be applied to real-world applications, since voltage state for every time step has to be detected.

Algorithms instructing training by learning windows, such as the STDP window, have been proposed recently [15]. In this respect, the remote supervised learning method (ReSuMe) is a typical one. Motivated by the Widrow–Hoff rule, it applies both the STDP and anti-STDP learning windows to drive training [21]. The perceptron-based spiking neuron learning rule (PBSNLR) using a learning window is based on the presynaptic voltage function to instruct training, which improves training efficiency [22]. The synaptic weight association training (SWAT) utilizes STDP learning window and the Bienenstock–Cooper–Munro learning rule [23] to direct learning and achieves success. Some other algorithms using learning windows are also introduced in [13] and [24]. These training algorithms employing learning windows are often more efficient than these with the gradient descent rule. Besides these supervised learning algorithms, some effective classifiers are also introduced recently, such as the efficient self-regulating evolving spiking neural (SRESN) classifier [25] and the robust classifier proposed in [26] for image recognition.

Compared with the human brain, these methods are far from reaching comparable recognition performance. Thorpe and Imbert [3] found that humans can analyze and classify visual patterns in 100 ms, although at least ten synaptic stages are involved, whereas the SNN often consumes several seconds or minutes to learn a pattern. Studies on the mechanisms of visual processing in monkeys [27] have revealed that the neurons in areas V4, TEO, and temporal cortex of the ventral stream show response selectivities for stimulus attributes that are important for object vision, such as shape, color, and texture. Multiple objects presented at the same time in the visual field compete for neural representation [28]. The selective attention mechanism of the primate visual system guarantees the information validity and the processing efficiency.

Motivated by the selective attention mechanism of the primate visual system [27], [28], an efficient learning algorithm, the accurate synaptic-efficiency adjustment (ASA) method is proposed in this paper to improve the efficiency of training SNN. Our algorithm only selects target spike time as attention areas and ignores the states of other time. Besides, it employs voltage difference to evaluate training errors, and uncovers the relationship between weight variation and voltage distance. This approach enables the ASA to calculate weight adjustment of each synaptic efficacy accurately. Experimental results show that our method achieves the state-of-the-art efficiency and extends the application of SNN, since its firing time can be set to an arbitrary real number without affecting the learning efficiency.

The rest of this paper is organized as follows. Section II introduces the related work briefly, and Section III elaborates our algorithm and its theoretical analysis. In Section IV, the properties of the ASA are investigated. Section V tests the classification capability of our algorithm on the UCI data set. Section VI gives conclusions and future works.

II. Spiking Neural Model

In this paper, the simple version of the spike response model (SRM_0 for short) is adopted because of its simplicity and effectiveness [6]. In the SRM_0, each neuron integrates the voltage sum of all the presynaptic spikes, and emits a spike when its voltage reaches the threshold. Once a spike j is generated at t_i^j, it inspires a voltage ϵ_j which is transmitted by its synapse to a postsynaptic neuron. The voltage of a postsynaptic neuron is described as

$$u(t) = \eta(t - \hat{t_o}) + \sum_{j \in \Gamma_j} w_j \epsilon_j\left(t - t_i^j\right) + u^{\text{ext}} \tag{1}$$

with

$$\epsilon_j(s_j) = \left[\exp\left(-\frac{s_j}{\tau_1}\right) - \exp\left(-\frac{s_j}{\tau_2}\right)\right] H(s_j) \tag{2}$$

where $H(\cdot)$ is a Heaviside step function which is set to 1 when $s_j \geq 0$ and otherwise 0. $s_j = t - t_i^j$, t is the current time, $\hat{t_o}$ is the most recent output spike time of the postsynaptic neuron, w_j is the weight of the presynaptic neuron emitting the jth input spike, and $\eta(t - \hat{t_i})$ is the function of the refractory period. Γ_j is a set containing input spikes emitted by all the presynaptic neurons. t_i^j is the jth firing time of the input spike train. u^{ext} is the external voltage to the neuron i, and τ_1 and τ_2 are constant parameters. In order to simplify the derivation of our algorithm, we set $\tau_1 = 2\tau_2$ in this paper.

The STDP learning window [15] is adopted in this paper, which is based on the STDP learning rule [6] and represented in (3). It gives the relationship of the weight adjustment magnitude and the spike time deviations

$$W_{\text{ind}}(s_2) = \begin{cases} A_{\text{pre}} \exp(-s_2/\tau), & \text{if } s_2 \geq 0 \\ -A_{\text{post}} \exp(s_2/\tau_{\text{post}}), & \text{if } s_2 < 0 \end{cases} \tag{3}$$

with $s_2 = t - t_i$ denoting the time distance from the input time t_i and the current time t. Since the input spikes fired after t have no contribution to the current voltage states, we set $A_{\text{post}} = 0$ in this paper. It indicates that only synapses with input spikes emitted before the current time are adjusted, and for simplicity, A_{pre} is set to 1.

III. ASA Learning Algorithm

A. Algorithm Description

The supervised learning in SNN aims at finding the relationship between the input and target output spikes. For traditional algorithms, whether they are based on the time distance between the output spike time t_o and the target time t_d with $\Delta w = F(t_o, t_d)$, such as the SpikeProp, ReSuMe, and so on, or based on the voltage distance between the voltage $u(t)$ and the threshold ϑ with $\Delta w = F(u(t), \vartheta)$, such as the Tempotron and so on, they all require detecting voltage states at all time intervals no matter if they contain useful information. However, studies in [27] and [28] state the existence of competition among multiple stimulus in primate visual systems. Only the stimulus winning the competition for representation in visual cortex will gain further access to the memory systems but not all stimuli. The selective attention mechanism shows

the attention-enhanced information and filters out irrelevant information.

Motivated by this mechanism, not all spike times but only the target ones are selected as the attention enhanced information and be detected both in the training and testing phases in our algorithm. The neuron states at untarget spike times are filtered out. Besides, a cost function based on the voltage difference at the attention point is employed instead of the traditional time difference. A learning process is completed when the voltage of the output neuron is equal to the threshold at all target time points, and after training, the output neuron is fired when its voltage is equal to the threshold at a target time point. The refractory period is added after each target spike and output spike in training and testing, respectively.

Given n presynaptic neurons and one postsynaptic neuron, the input spike train is denoted by $T_i = \{t_i^1, t_i^2, t_i^3 \ldots t_i^M\}$, where t_i^j denotes the jth input spike, and the desired output spike train is $T_d = \{t_d^1, t_d^2, \ldots, t_d^N\}$ in our algorithm. T_i contains all input spikes emitted by all input neurons, and without limiting the number of spikes, one input neuron can generate. Different patterns are denoted by different target spike trains. Consequently, our algorithm can be applicable to multiple spikes networks and multiple pattern recognition.

Employing the SRM_0 model, weight adjustment of our algorithm at the detected target spike time t_d for an input j is calculated as follows:

$$\Delta w_j = \frac{\gamma_j \left(\vartheta - u_{\mathrm{td}}^{\mathrm{out}}\right)}{\epsilon_j(s_j)} \tag{4}$$

with

$$\gamma_j = \frac{W_{\mathrm{ind}}(s_j)}{\sum_{k=m_1}^{m_2} W_{\mathrm{ind}}(s_k)} \tag{5}$$

where Δw_j represents the weight variation of the synapse emitting the jth input spike, ϑ is the firing threshold, and $u_{\mathrm{td}}^{\mathrm{out}}$ is the voltage of the output neuron at t_d. $s_j = t_d - t_i^j$ denotes the time distance from the input time t_i^j to the target time t_d, and similarly, $s_k = t_d - t_i^k$. $\epsilon_j(s_j)$ is the potential function illustrated in (2). γ_j is the normalized STDP parameter with $W_{\mathrm{ind}}(s_j)$ calculated by (3), and m_1 and m_2 are the input index boundaries satisfying $t_d - t_i^j \in [t_1, t_2]$ for the input spike time sequence.

In order to improve the learning efficiency and avoid the $\epsilon_j(s_j)$ calculated by (2) going to infinitesimal or zero when s_j is too large, the proper values of t_1 and t_2 are obtained by setting a voltage threshold ϑ_v. The voltage inspired by the input t_i^j is set to 0 at time t if this $t - t_i^j$ is not in the interval $[t_1, t_2]$, which are shown in Fig. 1. The values of t_1 and t_2 are derived in Theorem 2 in Section III.B.

In summary, the detailed pseudocode of our ASA algorithm is shown in Algorithm 1.

Supposing that there are M input spikes and N target spikes. The maximum learning epoch is defined to E_{po}, and the time length is T. Since our algorithm only trains at the target time T_d, the time complexity of our algorithm is $O(MNE_{po})$. For the traditional algorithms, such as the ReSuMe [21] and the PBSNLR [22], weights are modified at all time intervals with the length T. Obviously, our algorithm has less time complexity than that of the ReSuMe and the PBSNLR, because the length of T_d is shorter than that of T.

In the spatial complexity, the ASA and the ReSuMe need space of the same size to storage the target spike train, the input spike train, and the weight matrix, whereas the PBSNLR requires relatively large storage space, because all the postsynaptic potentials induced by every synapse at the time of all samples need to be calculated and stored before training [22].

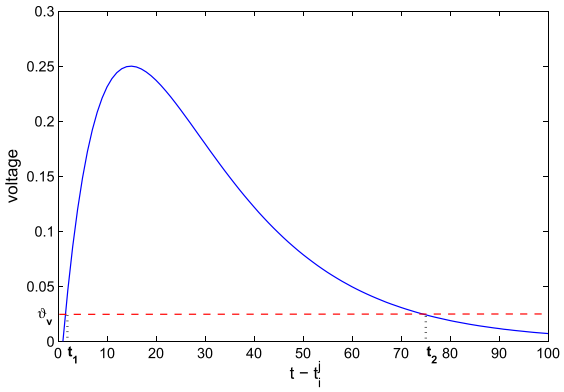

Fig. 1. Voltage $\epsilon_j(s_j)$ caused by the input spike t_i^j is above ϑ_v when $t - t_i^j \in [t_1, t_2]$.

Algorithm 1 ASA Learning Algorithm

Definition:
T_i: the set of input spikes, which contains spikes emitted all presynaptic neurons $\{t_i^1, t_i^2, t_i^3, \ldots, t_i^M\}$.
T_d: the set contains target spikes $\{t_d^1, t_d^2, \ldots, t_d^N\}$ of the postsynaptic neuron.
ϑ: firing threshold of the spiking neurons.

Initialization:
The weight matrix W is initialized randomly.

Training:
For each target time t_d^k:
If the voltage of the postsynaptic neuron at t_d^k is not equal to ϑ:
Step 1: Find all input spikes in T_i satisfying the condition $t_d^k - t_i^j \in [t_1, t_2]$, and save them to the set T_m.
Step 2: Modify the corresponding weight of each spike in T_m according to (4)-(5).
Else
Stop training of this target time t_d^k.
EndIf
EndFor

Testing:
Test the voltage at all target time points. If there is voltage at a target time not equal to ϑ, continue training until the voltage at all target time points is equal to ϑ or it achieves the maximum learning epochs.

B. Theoretical Derivation

In this section, we give a theoretical derivation of our algorithm.

Lemma 1: In the SRM_0 model, we provide that $u^{\mathrm{out}}(t_d)$ denotes the voltage function of the output neuron at the target time t_d, and w_j expresses the weight of the synapse emitting the jth input spike t_i^j. Assuming that in the training process, the refractory period is added at each target time, then

$$\frac{\partial u^{\mathrm{out}}(t_d)}{\partial w_j} = \epsilon_j\left(t_d - t_i^j\right). \tag{6}$$

Proof: Suppose that the output neuron only detects input spikes with the index range $j \in [m_1, m_2]$. According to the SRM_0 model described in (1), the voltage of the output neuron is

$$u(t) = \sum_{j=m_1}^{m_2} w_j \epsilon_j\left(t - t_i^j\right) + u^{\mathrm{ext}} + \eta(t - \hat{t_o}). \tag{7}$$

Clearly, at an arbitrary target time t_d with the refractory period added to the most recent target time $\hat{t_d}$, we do not need to detect the output spike. Then, the voltage at t_d is

$$u^{\mathrm{out}}(t_d) = \sum_{j=m_1}^{m_2} w_j \epsilon_j\left(t_d - t_i^j\right) + u^{\mathrm{ext}} + \eta(t_d - \hat{t_d}). \tag{8}$$

Since the $\epsilon_j(t_d - t_i^j)$ and $\eta(t_d - \hat{t_d})$ are constants at a fixed target time t_d, then

$$\frac{\partial u^{\mathrm{out}}(t_d)}{\partial w_j} = \epsilon_j\left(t_d - t_i^j\right). \tag{9}$$

The result follows.

Theorem 1: For any given neurons of SNN, assume that only input spikes with index $j \in [m_1, m_2]$ are detected and trained at a target spike time t_d, the refractory period is added at each target time in the training, and Δu_j is the variation of postsynaptic voltage inspired by the input spike t_i^j, and γ_j is the proportion of Δu_j in postsynaptic voltage variations Δu. When $\sum_{j=m_1}^{m_2} \gamma_j = 1$ holds, then

$$\Delta w_j = \frac{\gamma_j\left(\vartheta - u_{\mathrm{td}}^{\mathrm{out}}\right)}{\epsilon_j\left(t_d - t_i^j\right)} \tag{10}$$

makes the output voltage to reach the firing threshold ϑ at time t_d.

Proof: Suppose that before training, the voltage of the output neuron at time t_d is denoted by $u_{\mathrm{td}}^{\mathrm{out}}$, and $u_{\mathrm{td}}^{\mathrm{out2}}$ after training one epoch. Clearly

$$\Delta u_j = \frac{\partial u^{\mathrm{out}}(t_d)}{\partial w_j} \Delta w_j. \tag{11}$$

Then, the postsynaptic voltage after training is

$$u_{\mathrm{td}}^{\mathrm{out2}} = u_{\mathrm{td}}^{\mathrm{out}} + \sum_{j=m_1}^{m_2} \frac{\partial u^{\mathrm{out}}(t_d)}{\partial w_j} \Delta w_j. \tag{12}$$

According to Lemma 1 and (10), we get

$$\begin{aligned} \Delta u &= \sum_{j=m_1}^{m_2} \epsilon_j\left(t_d - t_i^j\right) \Delta w_j \\ &= \sum_{j=m_1}^{m_2} \epsilon_j\left(t_d - t_i^j\right) \frac{\gamma_j\left(\vartheta - u_{\mathrm{td}}^{\mathrm{out}}\right)}{\epsilon_j\left(t_d - t_i^j\right)}. \end{aligned} \tag{13}$$

Since

$$\sum_{j=m_1}^{m_2} \gamma_j = 1 \tag{14}$$

clearly

$$\Delta u = \vartheta - u_{\mathrm{td}}^{\mathrm{out}} \tag{15}$$

and then

$$u_{\mathrm{td}}^{out2} = u_{\mathrm{td}}^{\mathrm{out}} + \vartheta - u_{\mathrm{td}}^{\mathrm{out}} = \vartheta. \tag{16}$$

The result follows.

Theorem 1 indicates that all approaches for finding γ_j satisfying $\sum_{j=m_1}^{m_2} \gamma_j = 1$ are available. The normalized STDP rule expressed in (5) is employed in our algorithm because of its high bionics and application performance.

To choose proper values for the parameters m_1 and m_2, Lemma 2 and Theorem 2 are derived and illustrated as follows.

Lemma 2: For an arbitrary input spike t_i^j, supposing that $\tau_1 = 2\tau_2$, then its potential function $\epsilon_j(t - t_i^j)$ has its maximum value at $t = t_i^j + 2\tau_2 \ln 2$, and

$$\begin{cases} \dfrac{\partial \epsilon_j\left(t - t_i^j\right)}{\partial t} > 0, & \text{if } t_i^j \le t < t_i^j + 2\tau_2 \ln 2 \\ \dfrac{\partial \epsilon_j\left(t - t_i^j\right)}{\partial t} < 0, & \text{if } t > t_i^j + 2\tau_2 \ln 2. \end{cases} \tag{17}$$

Proof: When $t < t_i^j$, $\epsilon_j(t - t_i^j) = 0$. When $t \ge t_i^j$, by (2)

$$\begin{aligned} \epsilon_j\left(t - t_i^j\right) &= \exp\left(-\frac{t - t_i^j}{\tau_1}\right) - \exp\left(-\frac{t - t_i^j}{\tau_2}\right) \\ &= \exp\left(\frac{t_i^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) - \exp\left(\frac{t_i^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right). \end{aligned} \tag{18}$$

Taking the partial derivatives

$$\begin{aligned} \frac{\partial \epsilon_j\left(t - t_i^j\right)}{\partial t} &= -\frac{1}{\tau_1}\exp\left(\frac{t_i^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) \\ &\quad + \frac{1}{\tau_2}\exp\left(\frac{t_i^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right) \end{aligned} \tag{19}$$

and considering $(\partial \epsilon_j(t - t_i^j)/\partial t) > 0$, we get

$$-\frac{1}{\tau_1}\exp\left(\frac{t_i^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) + \frac{1}{\tau_2}\exp\left(\frac{t_i^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right) > 0 \tag{20}$$

$$\exp\left(\frac{t_i^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right) > \frac{\tau_2}{\tau_1}\exp\left(\frac{t_i^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) \tag{21}$$

$$\exp\left(\frac{t}{\tau_1} - \frac{t}{\tau_2}\right) > \frac{\tau_2}{\tau_1}\exp\left(\frac{t_i^j}{\tau_1} - \frac{t_i^j}{\tau_2}\right). \tag{22}$$

According to $\tau_1 = 2\tau_2$

$$\left(\frac{\tau_2 - \tau_1}{\tau_1\tau_2}\right)t > -\ln 2 + \left(\frac{\tau_2 - \tau_1}{\tau_1\tau_2}\right)t_i^j \tag{23}$$

$$t < t_i^j + 2\tau_2 \ln 2. \tag{24}$$

In the same way, we can derive that if the derivative $(\partial\epsilon_j(t-t_i^j)/\partial t) < 0$, then $t > t_i^j + 2\tau_2 \ln 2$. Consequently, we obtain the maximum value at $t = t_i^j + 2\tau_2 \ln 2$.

Theorem 2: Supposing that there is a voltage threshold $\vartheta_v < 1/4$, $\tau_1 = 2\tau_2$, and all input spikes with $\epsilon_j(t - t_i^j) < \vartheta_v$ are not detected at time t. Then, an input t_i^j is detected and trained at t_d only if $t_d - t_i^j \in [-\tau_1 \ln(1 + \sqrt{1-4\vartheta_v}/2), -\tau_1 \ln(1 - \sqrt{1-4\vartheta_v}/2)]$.

Proof: According to the function properties of the $\epsilon_j(t - t_i^j)$ derived from Lemma 2, the boundary values of t that satisfy the condition $\epsilon_j(t - t_i^j) \geq \vartheta_v$ are the solutions of $\epsilon_j(t - t_i^j) = \vartheta_v$

$$\exp\left(-\frac{t-t_i^j}{\tau_1}\right) - \exp\left(-\frac{t-t_i^j}{\tau_2}\right) = \vartheta_v \tag{25}$$

since $\tau_1 = 2\tau_2$

$$\exp\left(-\frac{t-t_i^j}{\tau_1}\right) - \left[\exp\left(-\frac{t-t_i^j}{\tau_1}\right)\right]^2 - \vartheta_v = 0. \tag{26}$$

According to the condition, $\vartheta_v < (1/4)$

$$\Delta = 1 - 4\vartheta_v > 0. \tag{27}$$

Therefore, this quadratic equation has two solutions

$$\exp\left(-\frac{t-t_i^j}{\tau_1}\right) = \frac{1 \pm \sqrt{1-4\vartheta_v}}{2} \tag{28}$$

then

$$t - t_i^j = -\tau_1 \ln\left(\frac{1 \pm \sqrt{1-4\vartheta_v}}{2}\right). \tag{29}$$

According to the curvilinear properties of the quadratic function, $\epsilon_j(t - t_i^j) \geq \vartheta_v$ when

$$-\tau_1 \ln\left(\frac{1+\sqrt{1-4\vartheta_v}}{2}\right) \leq t - t_i^j \leq -\tau_1 \ln\left(\frac{1-\sqrt{1-4\vartheta_v}}{2}\right). \tag{30}$$

Then, to each target time t_d, an input t_i^j is detected when $t_d - t_i^j \in [t_1, t_2]$, with

$$t_1 = -\tau_1 \ln\left(\frac{1+\sqrt{1-4\vartheta_v}}{2}\right) \tag{31}$$

$$t_2 = -\tau_1 \ln\left(\frac{1-\sqrt{1-4\vartheta_v}}{2}\right). \tag{32}$$

The result follows.

Since only spikes satisfying $t - t_i^j \in [t_1, t_2]$ are detected and trained, this mechanism avoids the emergence of infinitesimal or zero values calculated by (2) when s_j is too large. Ignoring the presynaptic spikes exerting minor influence over postsynaptic neurons, the conclusion of Theorem 2 improves the training efficiency. It is applied to all the spiking neural algorithms in our simulations for fair comparison.

In the derivation of the ASA algorithm, the interferences of other spikes are ignored. However, in most practical applications especially when recognizing multiple patterns, the interferences between different target spikes in training cannot be neglected. In this case, our algorithm requires more epochs to offset these interferences and gets convergent.

Fig. 2. Some images of the data set LabelMe. The original 256 × 256 color images are converted into 8-bit grayscale images.

IV. Simulation Results on Learning Properties

In this section, different learning properties of our algorithm are explored through some simulations. First, the grayscale images of the LabelMe data set are employed to demonstrate the classification capability and the robustness of our algorithm. Then, the synthetic data on various cases are applied to investigate the learning efficiency of our algorithm comprehensively. Simulation parameters are listed in the Appendix.

A. Encoding Method and Network Structure

Images from the urban and natural scene categories of the LabelMe data set [29] shown in Fig. 2 are used here to explore some learning properties of our method.

How to encode these images is an important problem of image recognition. Combining the temporal encoding and phase encoding, a feature-dependent phase encoding algorithm proposed in [30] is applied to our simulations. By this encoding algorithm, each encoding neuron is associated with a receptive field (RF), and the intensity value of each pixel in the RF is converted into a precisely timed action potential by a logarithmic intensity transformation

$$t_i = t_{\max} - \ln(ax_i + 1) \tag{33}$$

where x_i is the intensity value of pixel i, $t_{\max}$ is the maximum value of the encoding window, a is a scaling factor, and t_i is the firing time of neuron i. After which, a periodic oscillation is employed which is described as a cosine function [31]

$$i_{\text{osc}} = A_1 \cos(wt + \varphi_i) \tag{34}$$

where A_1 is the magnitude of the membrane oscillations, w is the phase angular velocity, and φ_i is the phase shift of the ith neuron in the RF defined by

$$\varphi_i = \varphi_0 + (i-1)\Delta\varphi \tag{35}$$

where φ_0 is the initial phase and $\Delta\varphi$ is the constant phase difference between nearby photoreceptor cells.

The encoding process is shown in Fig. 3. With an RF of size $n \times m$, the intensity value of each pixel in this RF is applied to calculate a firing time t_i of a encoding neuron by (33), and then one obtains its periodic oscillation value i_{osc} using (34). For the convenience of reconstruction, each i_{osc} is removed to its nearest peak, and the peak values of these $n \times m$ encoding neurons are encoded to the firing time of one neuron in the end. Consequently, intensity values of the $n \times m$ pixels in an RF are encoded to the firing time of one neuron. In our

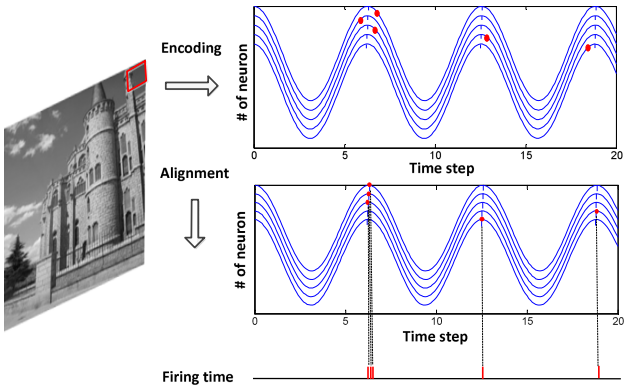

Fig. 3. Encoding process of the phase encoding algorithm.

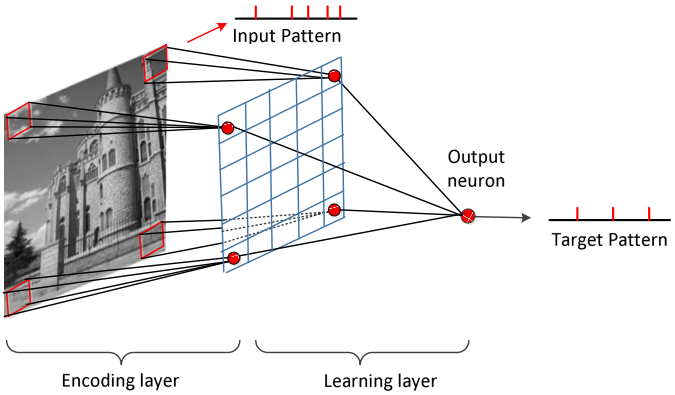

Fig. 4. Network structure devised in our simulation. It includes an encoding layer, a learning layer, and an output neuron. Each input neuron emits 64 spikes encoded by its connected 64 pixels.

simulations, we set $a = 0.0032$, $t_{\max} = 0.6$ s, $A_1 = 1$, $\varphi_0 = 0$, $\Delta\varphi = 2\pi/68$, $\omega = 100\pi$ rps, and $m = n = 8$.

The spiking network structure devised in our simulation is shown in Fig. 4, which contains three parts, the encoding layer, the learning layer, and the output neuron to generate target outputs. The encoding layer converts 64 pixel values into 64 spikes time of one input neuron using the method shown in Fig. 3. The learning layer learns the input pattern by the fully connected structure, and the output neuron gives the output pattern.

B. Learning Performance

To demonstrate the learning performance of our learning algorithm on image recognition tasks, two images are chosen and recognized by a network with 1024 input neurons and one output neuron which is described in Fig. 4. Using an RF of size 8×8, a gray scale of 256×256 is encoded into the firing states of 1024 neurons with each neuron emitting 64 spikes. Each target pattern is defined as a sequence of five spikes, [180, 280, 380, 480, 580] ms for image1, and [120, 220, 320, 420, 520] ms for image2.

The learning performance of our algorithm is shown in Fig. 5, in which the blue quadrate points denote the voltage of the output neuron before training at target time points, and the red circular points show the voltage after training. It indicates that after learning, all the voltage is equal to the threshold 10 at target time. The firing states at the untarget time are ignored. This simulation shows the neuron states of our algorithm after a successful learning.

C. Memory Performance

The memory performance is an important property of an algorithm indicating how much information a network can load when employing the algorithm. In this section, we investigate the memory performance of our proposed method compared with the traditional algorithms and their improved version. Since the amount of information can be measured by the number of patterns, in our simulation, the number of patterns a network can load is tested to measure the memory performance.

The SNNs with the same network settings as the previous experiments are employed here. The target output spike train for each image is generated according to the homogeneous Poisson process. The number of input images is gradually increased which are from the LabelMe database [29]. To evaluate the learning performance quantitatively in our simulations, the correlation-based measure C [32] described in the Appendix is employed to denote the similarity between the target and the actual output spike trains. A group of patterns is memorized successfully if the successful recall is above 0.8.

This simulation tests the number of patterns that an algorithm can memorize with different numbers of target spikes. To investigate the memory capability of our proposed selective attention learning mechanism, we apply the selective attention learning mechanism to the traditional PBSNLR [22] and ReSuMe [21], which is the ASA-based PBSNLR and the ASA-based ReSuMe, and compare them with the ASA, the traditional PBSNLR and ReSuMe. Like the ASA, the ASA-based PBSNLR and ReSuMe are trained only at the target time points, and use the error $\vartheta - u_{\text{td}}^{\text{out}}$ employed in (4). They are trained with the learning windows proposed in PBSNLR and ReSuMe instead of our normalized STDP.

The comparison results are shown in Fig. 6, which indicate that employing our proposed selective attention learning mechanism, the memory capabilities of both PBSNLR and ReSuMe are improved significantly. Besides, the ASA has substantially the same memory performance with the ASA-based PBSNLR and the ASA-based ReSuMe, which prove that the high memory capability of our algorithm is mainly due to the selective attention learning mechanism. For all algorithms, the more spikes in the target spike train, the smaller memory capability an algorithm possessing, because there are more information required to be processed for each pattern.

D. Algorithm Robustness

The robustness is another important property of a learning algorithm indicating the tolerate ability of different kinds of noise. In this simulation, we add different levels of partially occluded, Gaussian, and salt-and-pepper noise to the input images to investigate the robustness of our algorithm. Noise is added only in the test phases but not in the training processes.

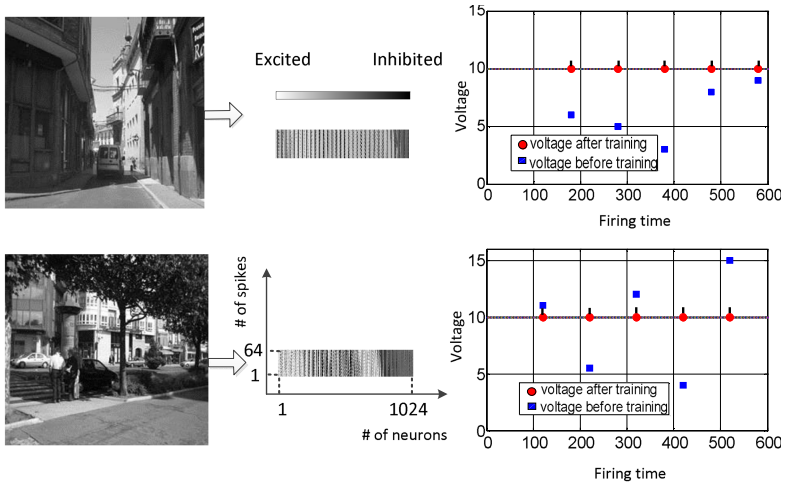

Fig. 5. Learning performance of the ASA. Left: original grayscale images. Middle: encoding results. Using an RF with size 8×8, a gray scale of 256×256 is encoded into the states of 1024 neurons with each neuron emitting 64 spikes. Right: voltage states before and after training, with the black bars denoting the target spike train. Seven and six learning epochs are required to achieve convergent for these two images, respectively.

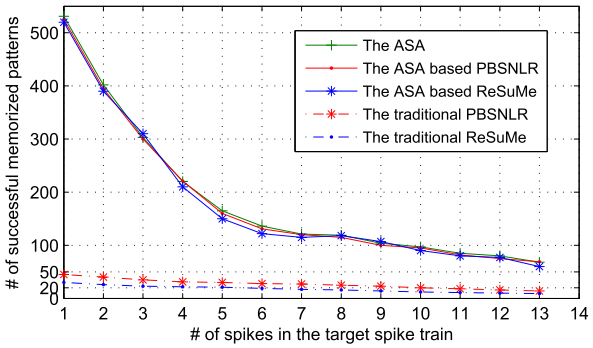

Fig. 6. Number of patterns memorized by the ASA, the traditional PBSNLR, the ASA-based PBSNLR, the traditional ReSuMe, and the ASA-based ReSuMe with different numbers of spikes in the target spike train.

A square with length a is applied to partially occlude an image at a random location. Besides, a Gaussian noise with means $m = 0$ and variance υ, and the salt-and-pepper noise specified by the noise density d is employed and shown in Fig. 7(a). These noises added to input images shift some of the firing times of the encoded spatiotemporal pattern; 50 different images are tested, and the average accuracy is obtained shown in Fig. 7(b).

In this simulation, the SNNs with the same settings as previous experiments are employed with original images for training and noisy images for testing. Simulation results shown in Fig. 7(b) indicate that our algorithm is more resistant to these noises to some extent. This is mainly because our algorithm tests voltage variations instead of traditional spike time variations. Consequently, some slight shifts on the input firing time have a smaller influence on the voltage than the precise firing time.

For instance, successful training results of the traditional precise spike time mechanism and our voltage detection mechanism are shown in Fig. 8(a) and (b), respectively, and their corresponding noisy testing results are shown in Fig. 8(c) and (d). The blue and red bars denote the target firing trains of two classes. It indicates that when there are some shifts derived by noise on input firing time, in the traditional precise spike time mechanism, the actual output train is completely different from the previous one because of the numerous refractory periods after output spikes. Differently, our algorithm detects spikes only in target ones, and ignores output spikes and their refractory periods in other time points. Fig. 8(d) shows that even some shifts are occurred on the input firing train, it only affects the voltage to some extent. The target firing train of the correct class still has the minimum voltage error compared with other classes, leading to the same recognition results as that without noise.

Since neurons cannot capture the information outside of the target intervals in our algorithm, the accuracy reduces faster than the traditional algorithms when the noise intensity is large enough which is shown in Fig. 7(b). But in these cases, all of these algorithms have bad antinoise performance.

E. Learning Efficiency

In this section, the learning efficiency of our algorithm is investigated compared with the traditional ReSuMe [21] and PBSNLR [22]. To evaluate the training efficiency of these algorithms extensively, four simulations are conducted on different cases of synthetic data, and the convergent epochs and training accuracy for each randomly generated pattern are tested. Since the training accuracy is tested in these simulations, the training and testing processes share the same data set generated by the homogeneous Poisson process.

1) The first simulation is devised to research the learning efficiency at different time lengths of spike trains. In this simulation, there are 400 input neurons and one output neuron. Each input and output neuron generates spikes by homogeneous Poisson processes with the rate $r_1 = 10$ Hz and $r_2 = 50$ Hz, respectively, ranging the time length from 200 to 2800 ms. In this manner, each neuron emits multiple spikes with firing rate r, then the average number of spikes a neuron generating

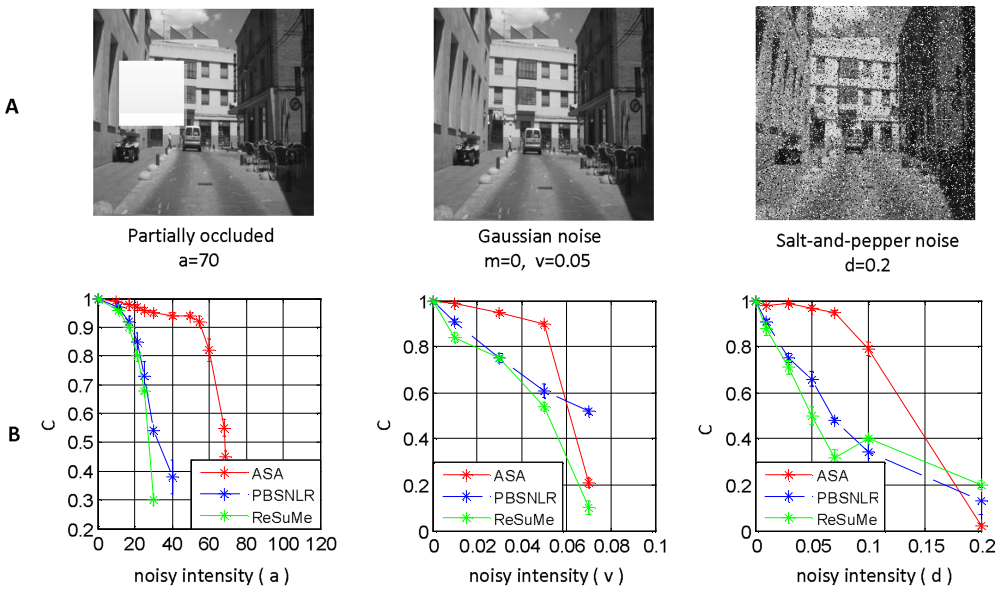

Fig. 7. Test results with different types of noise. (a) Images with different noises. Left: partially occluded image. Middle: image with Gaussian noise. Right: image with salt-and-pepper noise. (b) Recognition results of three learning algorithms on these three different noises, respectively.

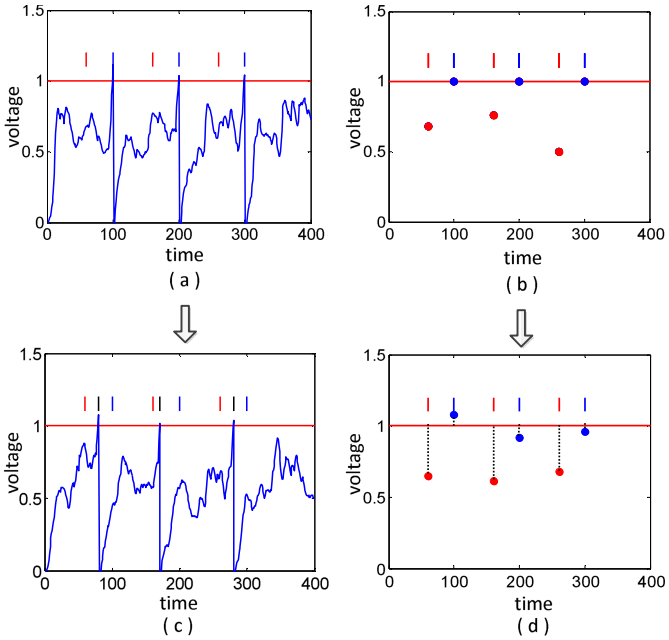

Fig. 8. Learning results with noise. Learning results of the traditional algorithms (a) with and (c) without noises, respectively. Learning results of our algorithm (b) with and (d) without noises, respectively.

is num $= r * t$, where t has the unit of seconds. To evaluate the learning accuracy quantitatively, the correlation-based measure C defined in the Appendix is applied to denote the training accuracy. The simulation results shown in Fig. 9(a) indicate that the ASA can complete learning at most 20 epochs and achieve $C = 1$. While the PBSNLR and the ReSuMe require hundreds of epochs at time lengths exceeding 800 ms, and obtain lower accuracies than the ASA. These differences increase with the growth of time length. Besides, it reveals that the PBSNLR has a higher accuracy than ReSuMe but a lower one than ASA. Although the PBSNLR shows a better performance than the ReSuMe, it has a higher standard deviation than other algorithms in both learning accuracy and epoch, which indicates its instability in learning errors.

2) The second simulation is conducted to investigate the learning efficiency under different firing rates of spike trains. Similar to the previous simulations, there are 400 input neurons and one output neuron. The input and output spike trains share the same time length of 800 ms, and the same firing rates which are generated by a homogeneous Poisson process ranging from 20 to 300 Hz. Simulation results shown in Fig. 9(b) reveal that the ASA can complete learning tasks at most 40 epochs and only requires five epochs at 20 Hz. Whereas, the PBSNLR and the ReSuMe require hundreds of epochs to be convergent when the firing rate exceeds 50 Hz. Similar to the previous results, our method can achieve higher accuracy than the PBSNLR and the ReSuMe. Fig. 9 shows that the ASA reduces computational epochs dramatically, and obtains a higher accuracy compared with the two popular algorithms in various learning situations of multispike sequence learning.
3) The third group of simulations explores the running time on one epoch instead of the number of epochs. In these simulations, the running time to complete one learning epoch with both different firing rates and various time lengths is detected. For each algorithm, learning time of 50 learning epochs are tested and the average is obtained. First, the running time under different firing rates is investigated. There are 500 input presynaptic neurons with the time length of 800 ms, and both the input and output trains have the same firing rates, ranging from 10 to 200 Hz. The running time of these three algorithms under different firing rates

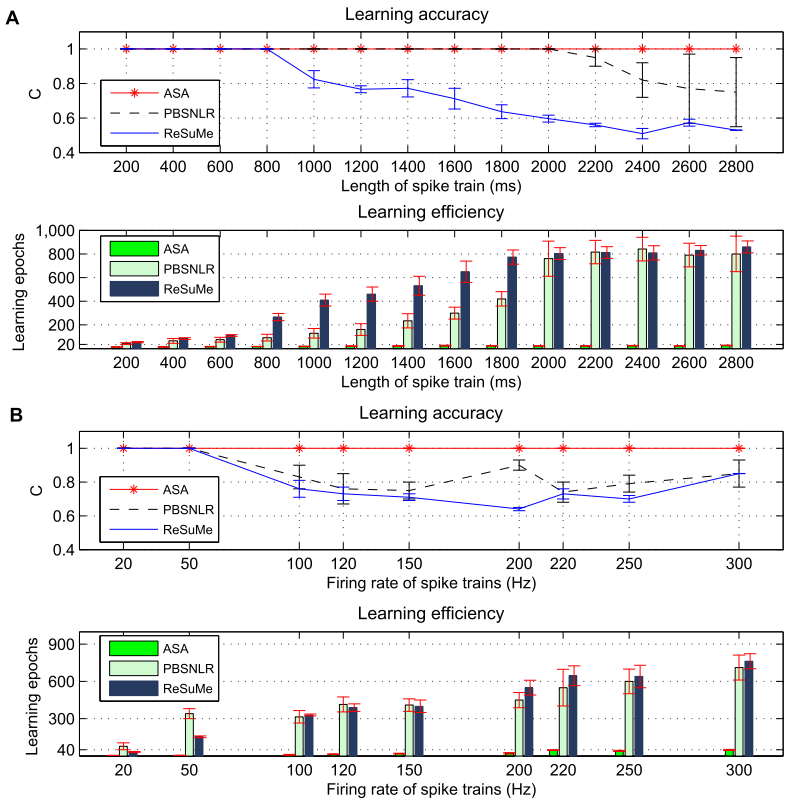

Fig. 9. Learning results on accuracy and efficiency. (a) Learning results at different time lengths. (b) Learning results under different firing rates.

TABLE I
RUNNING TIME OF ONE EPOCH FOR VARIOUS FIRING RATES

Firing rate (Hz)	Running time of the ASA(s)	Running time of the PBSNLR(s)	Running time of the ReSuMe(s)
10	0.003	0.061	0.542
20	0.007	0.112	0.831
50	0.044	0.307	1.551
100	0.135	0.639	2.882
120	0.181	0.857	3.288
150	0.286	1.021	3.987
200	0.401	1.432	5.125

TABLE II
RUNNING TIME OF ONE EPOCH FOR VARIOUS TIME LENGTHS

Time length (ms)	Running time of the ASA(s)	Running time of the PBSNLR(s)	Running time of the ReSuMe(s)
100	0.008	0.021	0.218
200	0.008	0.032	0.319
300	0.008	0.039	0.379
400	0.007	0.044	0.401
500	0.009	0.051	0.462
600	0.008	0.062	0.523
700	0.009	0.068	0.594

is shown in Table I, which reveals that our method requires the least amount of running time compared with the PBSNLR and the ReSuMe in all situations. Besides, more time is demanded for each algorithm with increasing firing rate. Second, to study the factors affecting the time consuming of these algorithms, one more simulation is conducted at different time lengths. In this simulation, the running time of one epoch is tested. Similar to the previous simulations, there are 500 input neurons, with the time step of 1 ms. In particular, the input and target spike trains are generated by homogeneous Poisson processes with the number of spikes fixed to 10. Simulation results shown in Table II indicate that our ASA takes dramatically less running time than the PBSNLR and the ReSuMe. Besides, the running time of the ASA algorithm has no prominent change with time length ranging from 100 to 700 ms, whereas the running time of the PBSNLR and ReSuMe increases obviously. This reveals the advantage of the selective attention mechanism employed in ASA, which enables our ASA to concentrate attention on target contents at target time and does not have to scan all time points as traditional algorithms do. Consequently, the running time of ASA is only relative to the number of target spikes, and has no obvious relation to the time length. Our algorithm relaxes the restrictions of SNN's applications, and extends temporal encoding's range to arbitrary large.

4) To investigate the influence factor on the learning efficiency of our algorithm, the unnormalized ASA is trained with $\Delta w_j = \eta \gamma_j (\vartheta - u_{\text{td}}^{\text{out}})$, which adopts the traditional unnormalized STDP learning window with $\gamma_j = W_{\text{ind}}(s_j)$ instead of the normalized one in (5). The ASA and its unnormalized version are conducted with different lengths of target spike trains on one image in the LabelMe data set.

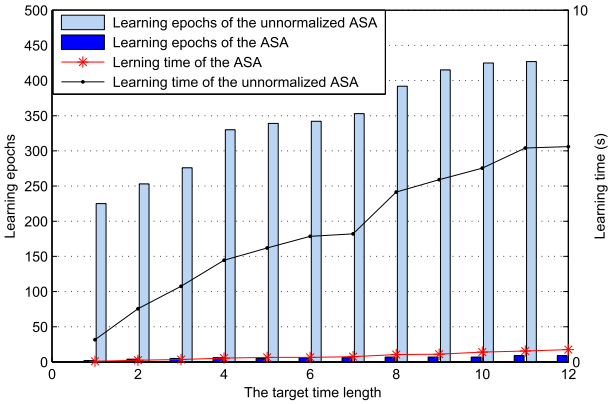

Fig. 10. Learning epochs and learning time of the ASA compared with the unnormalized ASA. All these simulations achieve $C = 1$.

TABLE III
RUNNING TIME FOR VARIOUS LENGTHS OF TARGET TRAINS

Length of the target spike train	Running time of the ASA (s)	Running time of the unnormalized ASA (s)
1	0.006	0.005
2	0.007	0.006
3	0.008	0.007
4	0.009	0.009
5	0.012	0.010
6	0.013	0.013
7	0.015	0.015
8	0.017	0.017
9	0.018	0.020
10	0.021	0.022
11	0.023	0.023
12	0.026	0.026

Fig. 10 shows the comparison results on the learning epochs and learning time of the ASA compared with the unnormalized ASA, which indicates that our ASA with normalization operating requires less learning epochs and learning time than that of the unnormalized ASA. This suggests that the normalized STDP learning window in our algorithm plays an important role in the effective learning. Table III shows the learning time of one epoch of the ASA and the unnormalized ASA, indicating that with the selective attention mechanism, these two algorithms require the same learning time to complete one epoch.

Simulations in this section demonstrate that our algorithm not only achieves a good performance on memory capability and robustness, but also reduces computational time dramatically. The accurate weight expression mechanism with the normalized STDP rule makes our algorithm to require less epochs for convergence, and the selective attention mechanism enables our algorithm to consume less time in each epoch than other SNN algorithms.

V. CLASSIFICATION ON THE UCI DATA SETS

In this section, the data sets of the Iris, Breast Cancer Wisconsin (BCW), Glass Identification, Piama Diabetes, and Liver Disorders from the UCI machine learning repository [33] are employed to investigate the capability of our algorithm over classification tasks. The number of features M and classes N in each data set are shown in Table IV.

TABLE IV
FIVE UCI DATA SETS ADOPTED FOR CLASSIFICATION

Data set	# Features (M)	# Classes (N)	# Samples	
			Training Set	Testing Set
Iris	4	3	135	15
BCW	9	2	614	69
Glass Identification	9	6	192	22
Pima Diabetes	8	2	691	77
Liver Disorders	6	2	310	35

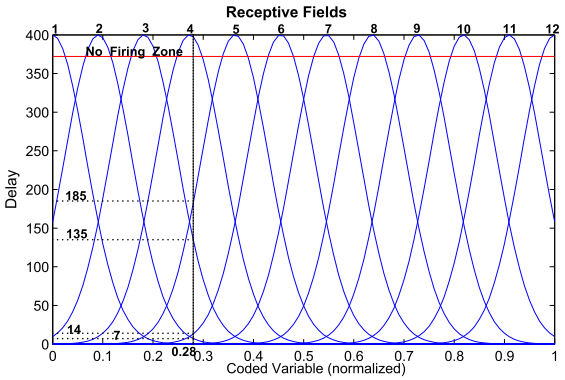

Fig. 11. Continuous input variable encoded by local RFs. The input variable is normalized to [0, 1], and a nonfiring zone is defined to avoid spikes in later time. Every no firing neuron has code -1. For instance, 0.28 is encoded to a spike train of 12 neurons: $-1, 7, 135, -1, 185, 14, -1, -1, -1, -1, -1, -1$.

In our classification, ten cross-validations are employed, and the average accuracies of training and testing sets are obtained. By the ten cross-validations, the number of samples in the training and testing set is shown in Table IV.

For better distinguishing the data, each feature value is mapped into a high-dimensional space using the population encoding method [34]. Several Gaussian RFs are defined to express the excitation, which is proportional to input values. These Gaussian RFs and their width σ and centers c_i are defined by the following equations:

$$f(x_i) = A_2 \exp\left(-\frac{(x_i - c_i)^2}{2\sigma^2}\right) \tag{36}$$

$$\sigma = \frac{1}{\gamma\,(m+1)} \tag{37}$$

$$c_i = \frac{i-1}{m-1} \tag{38}$$

where A_2 defines the encoded maximum value, m is the number of RFs, and γ controls the width σ. In our simulation, to generate 12 uniforming distributed Gaussian RFs in [0, 1], we set $\gamma = 1.5$, $m = 12$, and $A_2 = 400$. As shown in Fig. 11, these RFs can distribute an input variable over 12 input neurons, with each input neuron emits only one spike. Consequently, one feature is expressed by 12 neurons, and the encoding results of one sample with four features in Iris are shown in Fig. 12.

The network structure devised for this classification is shown in Fig. 13, and the input layer has $12 * M$ neurons with each 12 neurons representing one feature. For expressing the characteristics of the four features comprehensively, there are M hidden neurons possessing local connections to each

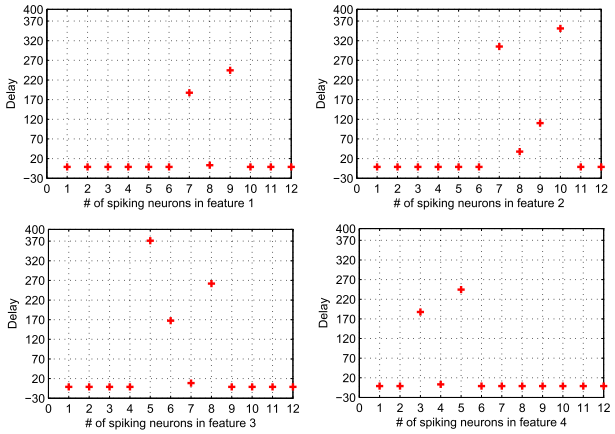

Fig. 12. Encoding result of one sample from Iris. Each feature is represented by the spikes of 12 neurons.

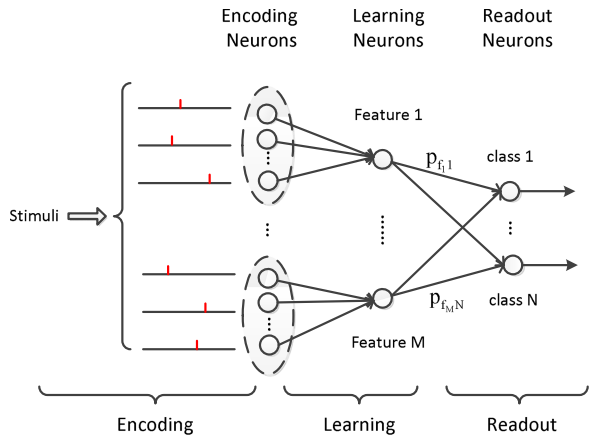

Fig. 13. Network structure consisting of input neurons, hidden neurons, and output neurons. Every 12 input neurons connect to one hidden neuron.

12 input ones. In the training period, only synaptic weights from input to hidden neurons are adjusted. The synaptic efficiencies in the output layer denote the weight of a feature f_i in the decision making, and they are fixed to the emergence probability p_{f_ic} with $p_{f_ic} = N_{f_ic}/N_c$, where N_{f_ic} is the number of class c's training samples that contain f_i in the ith feature, and N_c is the number of training samples in class c. In the Glass Identification, since the high similarity of data between different classes, the mean value of each feature is adopted to calculate this probability: $p_{f_ic} = \lfloor 100/|f_i - m_{f_ic}| + N_{f_ic}/N_c \rfloor$, where m_{f_ic} is the mean value of the ith feature in class c. In the decision making, a feature f_i votes to class c when the ith hidden neurons fire at the target times of c. To collect the vote for class c, the cth readout neuron adds a normalized p_{f_ic} when feature f_i votes to class c. A sample belongs to the class possessing the maximum vote value.

In this application, the sth sample of class c has the input spike train $\boldsymbol{t}_i^{sc}$ for four features and a target spike train $\boldsymbol{t}_d^{sc}$, which is obtained by $\boldsymbol{t}_d^{sc} = \boldsymbol{t}_i^{sc} + \boldsymbol{D}$, with the parameter $\boldsymbol{D}$ a constant vector which is set to 3 in our simulations. The local connections in our network enable us to train each independent subnetwork concurrently, since the output layer does not require training.

TABLE V
COMPARED WITH DIFFERENT ALGORITHMS FOR THE UCI DATA SET

Data set	Algorithm	Training accuracy	Testing accuracy	Training Time
Iris	SpikeProp	0.90	0.88	24.82s
	ReSuMe	0.91	0.88	12.86s
	PBSNLR	0.92	0.87	3.21s
	ASA	**0.96**	**0.95**	**0.015s**
	LIBSVM	0.98	0.99	0.008s
BCW	SpikeProp	0.92	0.91	25.82s
	ReSuMe	0.91	0.91	15.62s
	PBSNLR	0.92	0.92	1.73s
	ASA	**0.96**	**0.95**	**0.09s**
	LIBSVM	0.99	0.96	0.02s
Glass Identification	SpikeProp	0.79	0.73	11.72s
	ReSuMe	0.80	0.72	8.05s
	PBSNLR	0.81	0.73	5.17s
	ASA	**0.85**	**0.76**	**0.021s**
	LIBSVM	0.78	0.75	0.006s
Pima Diabetes	SpikeProp	0.77	0.70	73.3s
	ReSuMe	0.76	0.69	53.4s
	PBSNLR	0.77	0.71	16.8s
	ASA	**0.77**	**0.72**	**0.139s**
	LIBSVM	0.98	0.71	0.068s
Liver Disorders	SpikeProp	0.74	0.57	19.5s
	ReSuMe	0.76	0.58	16.3s
	PBSNLR	0.75	0.56	10.5s
	ASA	**0.76**	**0.60**	**0.063s**
	LIBSVM	0.99	0.61	0.0098s

1) Compared With Algorithms: We compare the convergent accuracy and learning efficiency of the ASA with some traditional algorithms, the SpikeProp [17], the ReSuMe [21], the PBSNLR [22], and the support vector machines (SVM). These SNN algorithms are conducted with the same network settings as our algorithm. The standard LIBSVM toolbox [35] with a radial-basis-function (RBF) kernel is applied to this classification. Different features are trained synchronously in our simulations of this section, and the training of a sample is stopped when $C > 0.95$ or it is not changed on 60 successive cycles.

Comparison results shown in Table V indicate that our algorithm has a better performance than the SpikeProp, ReSuMe, and PBSNLR, and has comparable accuracy with the SVM. In learning efficiency, our algorithm consumes less than 1 s for training, which is comparable with the SVM, and outperforms the SNN algorithms significantly. Like the LIBSVM, the training efficiency of our algorithm can meet the requirement of real-world applications.

2) Compared With SNN Classifiers: To further explore the learning performance of our method, we compare it with the existing SNN classifiers in terms of number of neurons, parameters, and training performance. The pioneered classifier based on the SpikeProp [17] and a classical classifier employing the SWAT [23] is compared. Since the recently proposed classifiers based on the self-regulating methods [36]–[39] are the current most efficient ones, their two newest methods, the SRESN classifier [25] and the growing-pruning SNN (GPSNN) [39] are compared in our simulation.

The comparative results are shown in Table VI, which indicate that the SRESN and the GPSNN require less neurons than other classifiers, since the self-regulating method enables them to add neurons dynamically according to the similarity

TABLE VI
COMPARED WITH DIFFERENT CLASSIFIERS FOR THE UCI DATA SET

Classifier	# Neurons	# Parameters	Testing accuracy	# Epochs
Iris				
SpikeProp [17]	63	24,000	0.96	1000
SWAT [23]	227	624	0.95	500
SRESN [25]	25	100	0.97	297
GPSNN [39]	29	138	0.96	97
ASA	**55**	**60**	**0.95**	**2**
BCW				
SpikeProp [17]	81	30,720	0.97	1500
SWAT [23]	128	243	0.96	500
SRESN [25]	47	90	0.96	285
GPSNN [39]	49	94	0.96	123
ASA	**119**	**126**	**0.95**	**2**
Liver Disorders				
SRESN [25]	6	216	0.59	715
GPSNN [39]	6	228	0.59	105
ASA	**80**	**84**	**0.60**	**3**
Pima Diabetes				
SRESN [25]	9	486	0.69	254
GPSNN [39]	7	392	0.71	51
ASA	**119**	**124**	**0.72**	**2**

of the data. Since neurons in the SRESN and the GPSNN are fully connected, and in our method are locally connected, the number of parameters for each neuron in our method is less than them. Then, when the samples have less features or large gap between data, the ASA has less parameters than the SRESN and the GPSNN, and conversely has more parameters.

Table VI shows that even with more neurons or parameters, our algorithm is still the most efficient one. The results reveal that the classification accuracy of the ASA is comparable with that of the other classifiers. Besides, our algorithm can be applied to various spiking network structures to achieve higher accuracy and reduce redundancy neurons.

VI. CONCLUSION

In this paper, an efficient supervised learning algorithm, the ASA is presented for SNNs. The accurate weight modification method using the normalized STDP learning rule enables our algorithm to achieve a rapid convergence. Besides, motivated by the selective attention mechanism of the primate visual system, our algorithm only focuses on the main contents in the target spike trains and ignores neuron states at the untarget ones, which makes our algorithm to achieve a significant improvement in efficiency for learning one epoch. Simulation results indicate that our algorithm outperforms the existing learning algorithms in learning efficiency, noise immunity, and learning capacity. Besides, ASA achieves a comparable learning performance with the SVM.

Our algorithm is derived from the SRM_0 model with Γ_j containing all input spikes, but the same derivation process is feasible to all models whose voltage u can be expressed by an equation of time t specifically. Besides, with this training efficiency, the SNN can be applied to various applications with arbitrary real-value analog inputs,and it is feasible to investigate a deep spiking model combining the strong computation capability of spiking neurons and the good feature representation ability of deep learning.

TABLE VII
LEARNING RATES IN EXPERIMENTS

Experiment position	PBSNLR	ReSuMe
Figure 6	0.0001–0.05	0.001–0.5
Figure 7	0.03	0.1
Figure 9A	0.4(200ms)-0.01(2800ms)	0.09(200ms)-0.01(800ms)
Figure 9B	0.04(20Hz)-0.01(300Hz)	0.08(20Hz)-0.03(300Hz)
Table I	0.1(10Hz)-0.08(200Hz)	0.2(10Hz)-0.06(200Hz)
Table II	0.1(100ms)-0.08(700ms)	0.2(100ms)-0.06(700ms)
Table V	0.01–0.1	0.01–0.5

TABLE VIII
VALUE OF τ_1 (ms) AND ϑ (mv) IN EXPERIMENTS

Experiment position	τ_1	ϑ
Figure 6	1	30
Figure 7	0.5	15
Figure 9A	6(200ms)–3(800ms)	10
Figure 9B	4(20Hz)–0.5(300Hz)	8(20Hz)–10(300Hz)
Figure 10	1	30
Table I	3(10Hz)–0.5(200Hz)	8(10Hz–50Hz),10(100Hz–200Hz)
Table II	3(100ms)–0.5(700ms)	6
Table III	1	1
Table V	4–8	1–30

APPENDIX

A. Experimental Details

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, which is chosen and shown in Tables VII and VIII. They enable us to achieve the comparable learning performance with their original references. The learning rates of the PBSNLR and the ReSuMe are shown in Table VII. τ_1 and ϑ in various simulations are shown in Table VIII.

In the simulations of Section IV, the training of one pattern is stopped when its training accuracy $C > 0.95$ or when it achieves the maximum learning epochs 5000. The maximum learning accuracy and its corresponding training epoch of these 5000 epochs are obtained. In the simulations of Section V, the training time is only allowed in 10 000 s.

The maximum time length $t_{\max}$ is 600 ms in the simulations of Figs. 5–7 and 10, and Table III, and 400 ms in Table V. The time step in all simulations is 1 ms. The ASA-based PBSNLR has a learning rate 0.01, and the ASA-based ReSuMe and the unnormalized ASA have a learning rate 0.05. The learning rates of the SpikeProp in Table V are set to 0.0005–0.001. The LIBSVM has RBF kernal with $c = 1$ and $g = 0.25$ for Iris, $c = 1$ and $g = 0.12$ for BCW, $c = 1.2$ and $g = 0.4$ for Glass Identification, $c = 1$ and $g = 0.17$ for Liver Disorders, and $c = 1.2$ and $g = 0.125$ for Piama Diabetes.

B. Error Measures

To evaluate the learning accuracy, we use the correlation-based measure method of C [32] to denote the similarity

between the target and actual output spike trains by

$$C = \frac{\boldsymbol{v_d} \cdot \boldsymbol{v_o}}{|\boldsymbol{v_d}||\boldsymbol{v_o}|} \tag{39}$$

where $\boldsymbol{v_d}$ and $\boldsymbol{v_o}$ are the vectors obtained by the convolution of the target and actual output spike trains using a Gaussian filter

$$g_i(t) = \sum_{m=1}^{N_i} G_{\sigma/\sqrt{2}}\left(t - t_m^i\right) \tag{40}$$

in which N_i is the number of spikes, t_m^i is the mth spike time, and $G_{\sigma/\sqrt{2}}(t) = \exp[-t^2/\sigma^2]$. $\boldsymbol{v_d} \cdot \boldsymbol{v_o}$ in (39) is the inner product, and $|\boldsymbol{v_d}|$, $|\boldsymbol{v_o}|$ are the Euclidean norms of $\boldsymbol{v_d}$ and $\boldsymbol{v_o}$, respectively. The standard deviations at the target and actual output time are set to $\sigma_d = \sigma_o = 1$.

For the ASA, when the classification accuracy of several patterns is evaluated, such as in Sections IV-C, IV-D, and V, the C is calculated to measure the similarity between the output and target spike trains. In this situation, the vector $\boldsymbol{v_o}$ in our method is obtained by target intervals instead of traditional all time intervals. When the training accuracy of one pattern is evaluated, the C is obtained by the voltage of the output neuron at target times with

$$g_i(u) = \sum_{m=1}^{N_i} G_{\sigma/\sqrt{2}}\left(u - u_m^i\right) \tag{41}$$

where N_i is the number of spikes, and u_m^i is the voltage at the mth spike time. In this situation, $g_i(u)$ is employed instead of (40) to test the similarity of the neuron voltage and the threshold, such as in Section IV-E.

REFERENCES

[1] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia, "Spatiotemporal firing patterns in the frontal cortex of behaving monkeys," *J. Neurophysiol.*, vol. 70, no. 4, pp. 1629–1638, Oct. 1993.

[2] T. J. Sejnowski, "Time for a new neural code?" *Nature*, vol. 376, no. 6535, pp. 21–22, 1995.

[3] R. Pfeifer, Z. Schreter, F. Fogelman-Soulie, and L. Steels, *Connectionism in Perspective*. New York, NY, USA: Elsevier, 1989.

[4] M. R. Mehta, A. K. Lee, and M. A. Wilson, "Role of experience and oscillations in transforming a rate code into a temporal code," *Nature*, vol. 417, no. 6890, pp. 741–746, 2002.

[5] R. Van Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex," *Neural Comput.*, vol. 13, no. 6, pp. 1255–1283, Jun. 2001.

[6] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[7] N. K. Kasabov, "NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Netw.*, vol. 52, pp. 62–76, Apr. 2014.

[8] F. Naveros, N. R. Luque, J. A. Garrido, R. R. Carrillo, M. Anguita, and E. Ros, "A spiking neural simulator integrating event-driven and time-driven computation schemes using parallel CPU-GPU co-processing: A case study," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1567–1574, Jul. 2015.

[9] K. Minkovich, C. M. Thibeault, M. J. O'Brien, A. Nogin, Y. Cho, and N. Srinivasa, "HRLSim: A high performance spiking neural network simulator for GPGPU clusters," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 316–331, Feb. 2014.

[10] B. A. Kaplan, M. A. Khoei, A. Lansner, and L. U. Perrinet, "Signature of an anticipatory response in area VI as modeled by a probabilistic model and a spiking neural network," in *Proc. Int. Joint Conf. Neural Netw.*, 2014, pp. 3205–3212.

[11] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward categorization on AER motion events using cortex-like features in a spiking neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 1963–1978, Sep. 2014.

[12] Z. Zhang, Q. Wu, Z. Zhuo, X. Wang, and L. Huang, "Wavelet transform and texture recognition based on spiking neural network for visual images," *Neurocomputing*, vol. 151, pp. 985–995, Mar. 2015.

[13] F. Ponulak and A. Kasiński, "Introduction to spiking neural networks: Information processing, learning and applications," *Acta Neurobiol. Experim.*, vol. 71, no. 4, pp. 409–433, 2011.

[14] G. J. Stuart and B. Sakmann, "Active propagation of somatic action potentials into neocortical pyramidal cell dendrites," *Nature*, vol. 367, no. 6458, pp. 69–72, 1994.

[15] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Competitive STDP-based spike pattern learning," *Neural Comput.*, vol. 21, no. 5, pp. 1259–1276, 2009.

[16] A. Mohemmed and S. Schliebs, "Training spiking neural networks to associate spatio-temporal input-output spike patterns," *Neurocomputing*, vol. 107, pp. 3–10, May 2013.

[17] 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.

[18] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural Netw.*, vol. 22, no. 10, pp. 1419–1431, 2009.

[19] 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.

[20] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neurosci.*, vol. 9, no. 3, pp. 420–428, 2006.

[21] 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.

[22] 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.

[23] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A spiking neural network training algorithm for classification problems," *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.

[24] 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.

[25] S. Dora, K. Subramanian, S. Suresh, and N. Sundararajan, "Development of a self-regulating evolving spiking neural network for classification problem," *Neurocomputing*, vol. 171, pp. 1216–1229, Jan. 2016.

[26] 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.

[27] R. Desimone and L. G. Ungerleider, "Neural mechanisms of visual processing in monkeys," *Handbook Neuropsychol.*, vol. 2, pp. 267–299, Jan. 1989.

[28] S. Kastner and L. G. Ungerleider, "Mechanisms of visual attention in the human cortex," *Annu. Rev. Neurosci.*, vol. 23, no. 1, pp. 315–341, 2000.

[29] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A database and Web-based tool for image annotation," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 157–173, 2008.

[30] Z. Nadasdy, "Information encoding and reconstruction from the phase of action potentials," *Frontiers Syst. Neurosci.*, vol. 3, p. 6, Jul. 2009.

[31] 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.

[32] 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.

[33] K. Bache and M. Lichman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[34] H. P. Snippe, "Parameter extraction from population codes: A critical assessment," *Neural Comput.*, vol. 8, no. 3, pp. 511–529, 1996.

[35] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[36] S. Dora, R. Savitha, and S. Suresh, "A basis coupled evolving spiking neural network with afferent input neurons," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2013, pp. 1–8.
[37] S. Dora, S. Suresh, and N. Sundararajan, "A sequential learning algorithm for a minimal spiking neural network (MSNN) classifier," in *Proc. Int. Joint Conf. Neural Netw.*, 2014, pp. 2415–2421.
[38] S. Dora, S. Suresh, and N. Sundararajan, "A sequential learning algorithm for a spiking neural classifier," *Appl. Soft Comput.*, vol. 36, pp. 255–268, Nov. 2015.
[39] S. Dora, S. Suresh, and N. Sundararajan, "A two stage learning algorithm for a growing-pruning spiking neural network for pattern classification problems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2015, pp. 1–7.

Xiurui Xie 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.

Her current research interests include neural networks, intelligent computation, 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.

He was a Post-Doctoral Fellow with the Advanced Robotics and Intelligent Systems Laboratory, School of Engineering, University of Guelph, Guelph, ON, Canada, from 2007 to 2008. 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.

Zhang Yi (F'16) received the Ph.D. degree in mathematics from the Institute of Mathematics, Chinese Academy of Sciences, 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 entitled *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 has been an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS since 2014.

Jürgen Kurths received the B.S. degree in mathematics from the University of Rostock, Rostock, Germany, the Ph.D. degree from the Academy of Sciences of the German Democratic Republic, Berlin, Germany, in 1983, the Honorary degree from the N. I. Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia, in 2008, and the Honorary degree from Saratow State University, Saratov, Russia, in 2012.

He was a Full Professor with the University of Potsdam, Potsdam, Germany, from 1994 to 2008. Since 2008, he has been a Professor of Nonlinear Dynamics with the Humboldt University of Berlin, Berlin, Germany, and the Chair of the Research Domain Transdisciplinary Concepts with the Potsdam Institute for Climate Impact Research, Potsdam. Since 2009, he has been the Sixth-Century Chair of Aberdeen University, Aberdeen, U.K. He has authored over 480 papers, which are cited more than 25 000 times (H-factor: 71). His current research interests include synchronization, complex networks, time series analysis, and their applications.

Dr. Kurths is a fellow of the American Physical Society and a member of the Academia Europea and the Macedonian Academy of Sciences and Arts. He received the Alexander von Humboldt Research Award from the Council of Scientific and International Research in India. He is an Editor of *PLos ONE*, *Philosophical Transactions of the Royal Society A*, and *CHAOS*.