Temporal coding in recurrent spiking neural networks with synaptic delay-weight plasticity

Guoyong Shi*, Jungang Liang, Yong Cui
School of Information, University of Southwest Petroleum, Nanchong 637001, Sichuan, China

ABSTRACT

As a brain-inspired artificial neural network computational model, a recurrent spiking neural network is composed of biologically plausible spiking neurons, which has taken on increasing importance in this study mainly include complex network structure and implicit nonlinear mechanism. This paper presents a learning algorithm with synaptic delay-weight plasticity for recurrent spiking neural network, which may enable real-time communication of complex spatiotemporal spike trains simulating organisms. First, by copying the hidden layer between the input and output layers, a context layer can be created. A total error for spatiotemporal pattern is then introduced, along with the learning rule for synaptic weights with synaptic delay plasticity. Moreover, by using synaptic delays of spike train from postsynaptic neuron, the proposed algorithm defines the learning rules for synaptic weights. In addition, the performance of the learning algorithm has been successfully tested to achieve high-precision learning with limited iterative resources. Finally, the main factors have been evaluated qualitatively and quantitatively, such as different lengths and frequencies for desired output spike trains.

Keywords: Recurrent spiking neural networks, context layer, synaptic delay plasticity, supervised learning rules

1. INTRODUCTION

Spiking neural networks (SNNs) establish interoperability properties at the artificial neural network level by simulating the communication mechanism of firing spike trains in real time[1,2]. Biological neuron signals can be converted into continuous functions by SNNs[2,3], which are widely used for spatial-temporal data analysis in research fields. Several categories of SNNs architecture have been developed in recent years based on different applicability, including single-layer SNNs, multi-layer feed-forward SNNs, and recurrent SNNs. Compared with the other two network structures, recurrent SNNs (RSNNs) are more biologically realistic, and it has the ability to simulate the actual biological activities of biological pulse issuance, transmission, processing and feedback in real time. At the same time, RSNNs are the artificially constructed structures that are closest to the real biological complex network node links. An artificial neural network with recurrent connections is a type of machine learning model with a feedback loop structure that can be used to assist in solving parallel processes, such as machine automation[4], intelligent recognition[5], and textual analysis[6].

A key component of learning and memory is synaptic plasticity. As part of his initial synaptic plasticity hypothesis, Hebb presented that synergistic activity between presynaptic and postsynaptic neurons played a vital role in increasing synaptic potentiation[7]. With the in-depth exploration of researchers, excellent results have emerged, including strengthening synaptic weights using spike-timing dependent plasticity (STDP)[8], weakening them using anti-STDP[9], threshold driven plasticity rules (TDP)[10], precise-spike-driven synaptic plasticity[11] and somato-dendritic synaptic plasticity[12]. Consequently, the supervised synaptic plasticity learning mechanism, such as STDP and TDP, missing the characteristic of the delay in the propagation of impulse signals in biological synapses, cannot be directly used to express the biological communication mechanism of RSNNs. From this point of view, it focuses more on the exploration of neural dynamics rather than just analyzing the macro variables of neural networks[13]. On the basis of the comparative study, scientists do not emphasize the importance of developing synaptic weights between neuronal connections, which calls for developing sophisticated algorithms for RSNNs.

The aim of this paper is to propose a supervised learning rule based on synaptic delay plasticity with a spike train weighting adjustment operation in all layers, which can improve the learning ability for complex spatiotemporal patterns. The update method of synaptic weights is defined in combination with the synaptic delay, and then a supervised learning rule is constructed to realize the spatiotemporal communication of spike train. In Section 2 we mainly introduce the

* shiguoyong@swpu.edu.cn

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 125060D

network structure of RSNNs and the simplified neural information processing flow by introducing the context layer. In Section 3 we give the construction process of learning rules for each layer of the network. In Section 4 we show the learning performance of the spike train and the effect of different element variations. A summary of the work in paper is presented in Section 5.

2. RELATED WORK

This section mainly describes the linear network structure and conceptual network introduction of RSNNs in detail. Besides, spike train function is introduced through mathematical expressions.

2.1 The network structure

In this study, a multi-layer spiking neural network is recursively constructed using random coupling and intrinsic connectivity. As shown in Figure 1, the network structure consists of three main layers, namely input, hidden and output, and each layer consists of neuronal connections. It is obvious that the processing process between the layers for spike train is a sequential recursive feed-forward structure, while the randomly coupled real-time communication of the neurons within the hidden layers forms recurrent connections. To achieve a recurrent connection approach as shown in Figure 1 by the dotted line, a connectivity degree (C_d) is applied to indicate the tightness of the coupling within the hidden neurons. In fact, the hidden neurons are connected in an intricate way, which is more reflected by random coupling, so we just artificially abstract to facilitate the construction of network model.

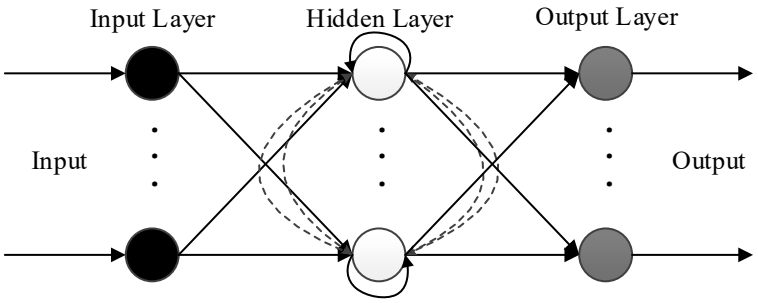

Figure 1. The constructed network structure.

As mentioned above, RSNNs are complex in structure, so it is difficult to develop an algorithm for spike train learning in terms of synaptic delay-weight plasticity. In order to follow the structural principle and facilitate program design, a copy of the recurrent layer (called the context layer) is placed at the bottom with the aim of converting RSNNs into an equivalent feed-forward structure, as shown in Figure 2.

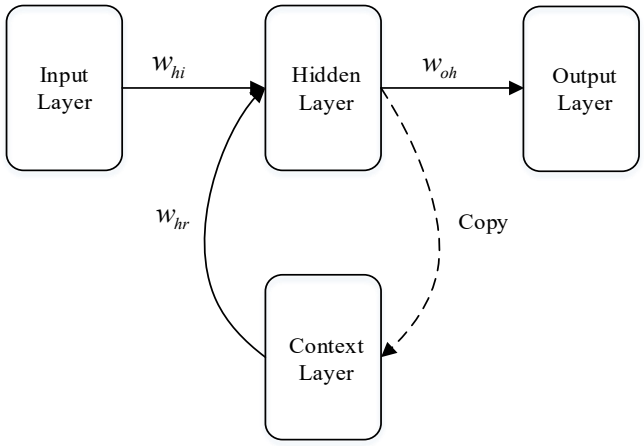

Figure 2. The network structure conversion with context layer.

2.2 The spike train function

After the biological neuron is stimulated by the presynaptic a spike train s_i, it will generate the stress response output spike train s_o. A continuous-time linear Poisson neuron model is used to establish the relationship between input and output spike trains, which can be shown as:

$$f_{s_o}(t) = \sum_{i=1}^{A} w_{oi} f_{s_i}(t) \tag{1}$$

where the number of presynaptic neurons is denoted by A, and an important part of the spike train processing between neuron i and another neuron o is the synaptic weight w_{oi}. Based on the results in References[14,15], learning rules can be constructed using a simplified linear summation mechanism of smooth spike trains.

In addition, a smoothing function h is chosen for mathematical convenience, which allows discrete spike trains to be converted into continuous functions. On $R_2(\varpi)$ space, s_i and s_j are defined as follows[16]:

$$F(s_i, s_j) = \left\langle f_{s_i}, f_{s_j} \right\rangle_{R_2(\varpi)} = \sum_{m=1}^{F_i} \sum_{n=1}^{F_j} \int_{\varpi} u(t - t_i^m)\, u(t - t_j^n)\, \mathrm{d}t = \sum_{m=1}^{F_i} \sum_{n=1}^{F_j} \Theta(t_i^m, t_j^n) \tag{2}$$

where an autocorrelation between smoothing function u and the kernel function Θ, and the kernel can be defined as $\Theta(t^m, t^n) = \int_{\Gamma} u(t - t^m) u(t - t^n)\, dt$.

3. SUPERVISED LEARNING ALGORITHM WITH SYNAPTIC DELAY-WEIGHT PLASTICITY

In this case, we can define s_o^a, as the desired spike train, and s_o^d, as the actual spike train, respectively. By combining equations (1) and (2), we can interpret the total error of RSNNs at time t as the error train in output spikes:

$$\begin{aligned} E &= \int_{\Gamma} E(t)\mathrm{d}t = \frac{1}{2} \sum_{o=1}^{N_o} \int_{\varpi} \left[f_{s_o^a}(t) - f_{s_o^d}(t) \right]^2 \mathrm{d}t = \frac{1}{2} \sum_{o=1}^{N_o} \left\langle f_{s_o^a}(t) - f_{s_o^d}(t), f_{s_o^a}(t) - f_{s_o^d}(t) \right\rangle \\ &= \frac{1}{2} \sum_{o=1}^{N_o} \left[F(s_o^a, s_o^a) - 2F(s_o^a, s_o^d) + F(s_o^d, s_o^d) \right] \end{aligned} \tag{3}$$

For all output neurons, the error function can be transformed into the inner product of spike trains, facilitating the adjustment of real-time spike train for an efficient learning process.

By backpropaguing network errors to synaptic weights through error backpropagation, a generalized delta update rule is used to modify synaptic weights. As a result of synaptic connections, the synaptic weight w is calculated as follows:

$$\Delta w = -\eta \nabla E + d_i \tag{4}$$

where η is the learning rate, ∇E is the error function for adjusting the learning efficiency of spike train, and d_i is synaptic delay, which can be expressed as:

$$d_i = d_\phi - D_i^f \tag{5}$$

where d_ϕ is the synapse ϕ that need to be adjusted, and D_i^f denotes the distance between the time interval when the maximum postsynaptic potential occurs between the input and the desired after the generation of spike trains, which can be represented as:

$$D_i^f = \left| t_i^f + d_i + \psi - t \right| \quad if \quad t \geq t_i^f + \psi \tag{6}$$

If $t > t_i^f + d_i + \psi$, the synaptic delay d_i is assigned as $d_i = d_\phi + D_i^f$. Otherwise, d_i is assigned as $d_i = d_\phi - D_i^f$. The mathematical expression for the gradient ∇E is:

$$\nabla E = \int_{\Gamma} \frac{\partial E(t)}{\partial w} \mathrm{d}t \tag{7}$$

As demonstrated in Figure 2, synaptic weight learning rules for the network layer can be separated into three layers: output, context, and input.

3.1 Learning rule in output layer (N_O)

Using equation (7) and the corresponding partial derivative transformation operation, the gradient ∇E_{oh} is associated with the recurrent neurons at time t, which can be shown:

$$\begin{aligned} \nabla E_{oh} &= \int_{\varpi} f_{s_h}(t)\left[f_{s_o^a}(t) - f_{s_o^d}(t) \right] \mathrm{d}t = \left\langle f_{s_o^a}(t), f_{s_h}(t) \right\rangle - \left\langle f_{s_o^d}(t), f_{s_h}(t) \right\rangle \\ &= F(s_o^a, s_h) - F(s_o^d, s_h) \end{aligned} \tag{8}$$

Accordingly, the rule for updating the synaptic weight Δw_{oh} based on the inner products of adjacent layer relationships and spike train conversion transitions is:

$$\Delta w_{oh} = -\eta\left[F(s_o^a, s_h) - F(s_o^d, s_h) \right] = \eta\left[\sum_{m=1}^{F_o^d} \sum_{n=1}^{F_h} \Theta(t_d^m, t_h^n) - \sum_{m=1}^{F_o^a} \sum_{n=1}^{F_h} \Theta(t_a^m, t_h^n) \right] + d_{io} \tag{9}$$

where the spike train counts of the actual and the expected are denoted by F_o^a and F_o^d, both as continuous values.

3.2 Learning rule in context layer (N_C)

From equation (7) and the partial derivative operation, the gradient ∇E_{hr} is associated with the recurrent neurons at time t, which can be represented as:

$$\nabla E_{hr} = \int_{\varpi} \frac{\partial E(t)}{\partial f_{s_h}(t)} \frac{\partial f_{s_h}(t)}{\partial w_{hr}} \mathrm{d}t = \sum_{o=1}^{N_o} \int_{\varpi} \left[f_{s_o^a}(t) - f_{s_o^d}(t) \right] \left(w_{oh} + \sum_{r=1}^{N_R} w_{hr} w_{or} \right) f_{s_r}(t) \mathrm{d}t \tag{10}$$

The rule for updating the synaptic weight Δw_{hr} of adjacent layer relationships and spike train conversion transitions is:

$$\begin{aligned} \Delta w_{hr} &= -\eta \sum_{o=1}^{N_o} \left[F(s_o^a, s_r) - F(s_o^d, s_r) \right] \left(w_{oh} + \sum_{r=1}^{N_R} w_{hr} w_{or} \right) + d_{ic} \\ &= \eta \sum_{o=1}^{N_o} \left[\sum_{m=1}^{F_o^d} \sum_{n=1}^{F_r} \Theta(t_d^m, t_r^n) - \sum_{m=1}^{F_o^a} \sum_{n=1}^{F_r} \Theta(t_a^m, t_r^n) \right] \left(w_{oh} + \sum_{r=1}^{N_R} w_{hr} w_{or} \right) + d_{ic} \end{aligned} \tag{11}$$

3.3 Learning rule in input layer (N_I)

In the same way, the gradient ∇E_{oh} is associated with the recurrent neurons at time t, which can be show as:

$$\nabla E_{hi} = \int_{\varpi} \frac{\partial E(t)}{\partial f_{s_h}(t)} \frac{\partial f_{s_h}(t)}{\partial w_{hi}} \mathrm{d}t = \sum_{o=1}^{N_o} \int_{\varpi} \left[f_{s_o^a}(t) - f_{s_o^d}(t) \right] \left(w_{oh} + \sum_{r=1}^{N_R} w_{hr} w_{or} \right) f_{s_i}(t) \mathrm{d}t \tag{12}$$

Similarly, the rule of the synaptic weight Δw_{hi} updated is:

$$\begin{aligned} \Delta w_{hi} &= -\eta \sum_{o=1}^{N_o} \left[F(s_o^a, s_i) - F(s_o^d, s_i) \right] \left(w_{oh} + \sum_{r=1}^{N_R} w_{hr} w_{or} \right) + d_{iI} \\ &= \eta \sum_{o=1}^{N_o} \left[\sum_{m=1}^{F_o^d} \sum_{n=1}^{F_i} \Theta(t_d^m, t_i^n) - \sum_{m=1}^{F_o^a} \sum_{n=1}^{F_i} \Theta(t_a^m, t_i^n) \right] \left(w_{oh} + \sum_{r=1}^{N_R} w_{hr} w_{or} \right) + d_{iI} \end{aligned} \tag{13}$$

Finally, a summary of the learning rules for synaptic weights is given in equations (9), (11), and (13) corresponding to each layer of the RSNNs.

4. EXPERIMENTS

This section shows the learning capabilities of RSNNs when trained with a supervised learning algorithm with spike weights. The learning algorithm is set up by defining the initial parameters to facilitate testing and verification, and then the emulating performance of spike train based on learning rules is analyzed.

4.1 Experimental setup

The short-term memory spike response model (SRM)[17] is chosen in the experiment with the aim of constructing the network as a neuronal model, which has the ability to solve the spatiotemporal spike pattern problems of RSNNs after training.

Table 1. Related parameter settings.

Sections	Parameters	Settings
SRM	The absolute refractory period	1 ms
	The neuron threshold	1.0
	The number of neurons in each layer	$N_i = N_r = 150$ $N_o = 1$
	Initialized synaptic weight distribution interval	(0, 0.5)
	The connectivity degree	C_h=0.35
Other variables	The rate at which Poisson generates spikes	r_i=20 Hz r_d=50 Hz
	The Gaussian kernel	$\Theta(x, y) = \exp(-\lvert x - y \rvert^2 / 2\sigma^2)$
	The learning rate	0.001
	The time step	0.1

The settings of related parameters such as SRM model and other variables are shown in Table 1. Finally, at least one test path is satisfied based on the averaged experimental results over 100 trials, specifically applying the learning epoch for up to 500 steps or until no network error exists.

4.2 Elemental analysis

Figure 3 shows the simulation threads of spike train based learning rules with synaptic delay. Figure 3a illustrates an overview of the learning process of the pulse sequence, which is divided into three phases: initial, desired and actual. Through supervised learning over about 250 epochs, the learning algorithm produces the desired output spike train in 150 ms intervals based on the initial output spike train. As shown in Figure 3b, the total network error E evolves over time. Following 250 learning epochs, the error function drops to 0 as the network is iteratively learned. This means that the spike train achieves the actual-to-expected identicality.

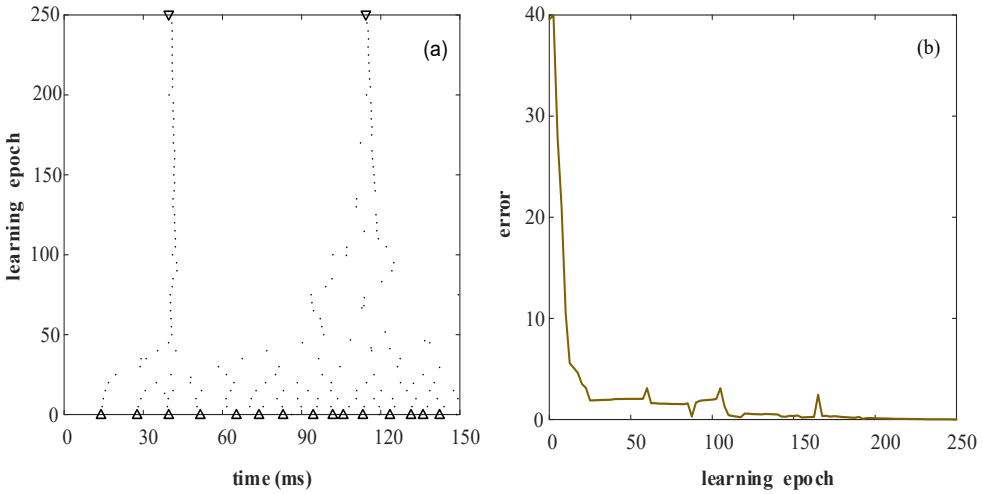

Figure 3. The simulation threads of spike train based learning rules with synaptic delay for RSNNs. (a): The efficient iterative process. Δ, ∇ and $\cdot$ all denote spike trains, each specific to the initial, desired and actual; (b): The errors on the network evolve over time.

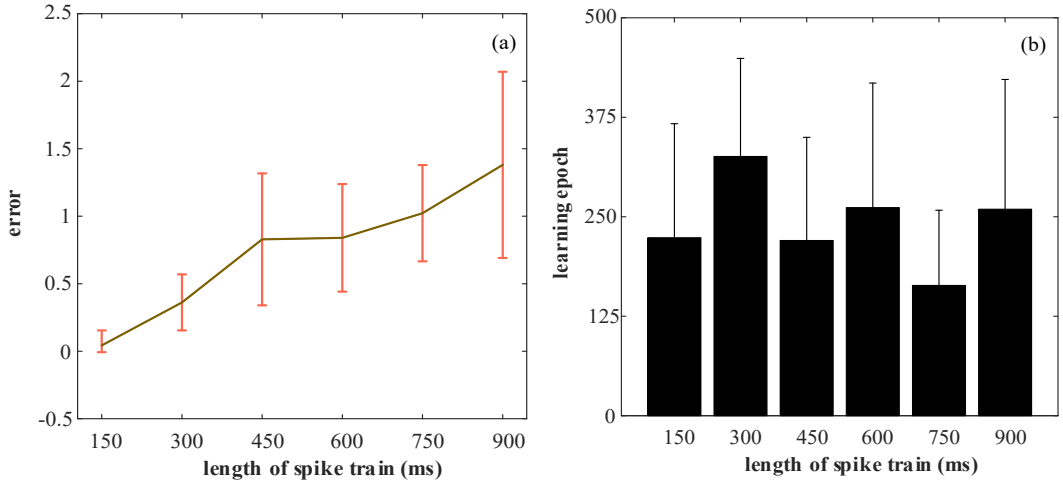

Figure 4. The spike trains with varying lengths when iterating through 500 learning epochs. (a): The differences in network errors due to spike train lengths; (b): The effect of spike train lengths on errors and learning epochs.

Figure 4 shows the variation of network errors based on desired output spike train of different lengths. Figure 4a illustrates that the optimal expected output spike train length of the algorithm is 150 ms, then high-precision autonomous learning can be performed, and the error increases significantly with the length of spike train. It is intuitively evident from Figure 4b that the learning epoch number for the algorithm with synaptic delay incorporated is about 250 epochs. The main reason is that the occurrence of negative feedback regarding learning accuracy and spike train length, where the error rises significantly when the length increases.

Figure 5 shows the variation of network errors based on desired output spike train of different frequencies. Figure 5a illustrates that the network error grows significantly as the frequency of desired output spike train increases, and the algorithm can perform high-precision autonomous learning when the output frequency is set to 50 Hz. From Figure 5b, the learning epoch of constructed rules is approximately in equilibrium as output spike frequency varies.

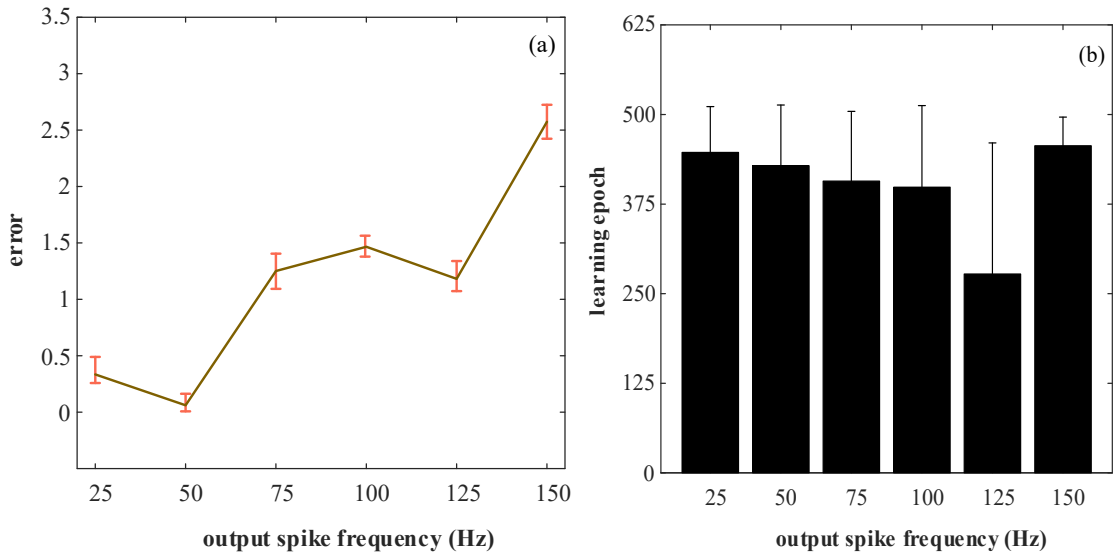

Figure 5. The spike trains with varying frequencies when iterating through 500 learning epochs. (a): The differences in network errors due to spike train frequencies; (b): The effect of spike train frequencies on errors and learning epochs.

5. CONCLUSIONS

In this work, we propose a multi-spike learning rule with synaptic delay-weight plasticity based on weighting adjustment operation of spike train for RSNNs. The introduction of synaptic delay makes the adjustment of synaptic weights more similar to the processing process of biological neurons for spike train. The synaptic weight update rules combined with synaptic delay are given mathematical representations on N_O, N_C, and N_I, respectively. The high-precision experimental results are obtained in the learning process by integrating the learning algorithm with synaptic delay, including the analysis of the optimal learning performance for spike train, the influence of different spike train lengths, and the change of the different output frequency for desired spike train. Therefore, the updating of synaptic weights based on the delay characteristic may be utilized in various neuron learning rules, neuron models and complex network structures.

ACKNOWLEDGMENTS

The research is supported by the City-School Science and Technology Strategic Cooperation Project of Nanchong City under Grants no. SXQHJH014, and Philosophy and Social Sciences Research Planning Project of Nanchong City under Grants no. NC22C361.

REFERENCES

[1] Demin, V. A., Nekhaev, D. V. and Surazhevsky, I. A., “Necessary conditions for STDP-based pattern recognition learning in a memristive spiking neural network,” Neural Networks 134, 64-75 (2021).
[2] Wang, X., Lin, X. and Dang, X., “Supervised learning in spiking neural networks: A review of algorithms and evaluations,” Neural Networks 125, 258-280 (2020).
[3] Tan, C., Šarlija, M. and Kasabov, N., “NeuroSense: Short-term emotion recognition and understanding based on spiking neural network modelling of spatio-temporal EEG patterns,” Neurocomputing 434, 137-148 (2021).
[4] Shaban, A., Bezugam, S. S. and Suri, M., “An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation,” Nature Communications 12, 1-11 (2021).
[5] Shen, J., Liu, J. K. and Wang, Y., “Dynamic spatiotemporal pattern recognition with recurrent spiking neural network,” Neural Computation 33, 2971-299 (2021).
[6] Shen, J., Lin, K. and Wang, Y., “Character recognition from trajectory by recurrent spiking neural networks,” 39th Annual Inter. Conf. of the IEEE Engineering in Medicine and Biology Society, 2900-2903 (2017).

[7] Hebb, D. O., “Perception of a complex: The phase sequence,” [The Organization of Behavior: A Neuropsychological Theory], Psychology Press, chapter 5, 79-106 (2005).
[8] Clopath, C., Büsing, L. and Vasilaki, E., “Connectivity reflects coding: a model of voltage-based STDP with homeostasis,” Nature Neuroscience 13, 344-352 (2010).
[9] Rumsey, C. and Abbott, L., “Synaptic democracy in active dendrites,” Journal of Neurophysiology 96, 2307-2318 (2006).
[10] Yu, Q., Li, H. and Tan, K. C., “Spike timing or rate neurons learn to make decisions for both through threshold-driven plasticity,” IEEE Transactions on Cybernetics 49, 2178-2189 (2018).
[11] Yu, Q., Tang, H. and Tan, K. C., “Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns,” Plos One 8, e78318 (2013).
[12] Schiess, M. and Urbanczik, R., “Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites,” PLoS computational biology 12, e1004638 (2016).
[13] Pani, D., Meloni, P. and Tuveri, G., “An FPGA platform for real-time simulation of spiking neuronal networks,” Frontiers in Neuroscience 11, 90 (2017).
[14] Carnell, A. and Richardson, D., “Linear algebra for time series of spikes,” ESANN, 363-368 (2005).
[15] Sporea, I. and Grüning, A., “A supervised learning in multilayer spiking neural networks,” Neural computation 25, 473-509 (2013).
[16] Paiva, A. R. C., Park, I. and Principe, J. C., “A reproducing kernel Hilbert space framework for spike train signal processing,” Neural Computation 21, 424-449 (2009).
[17] Wulfram, G. and Werner, M., [Spiking Neuron Models: Single Neurons, Populations, Plasticity], Cambridge University Press, (2002).