frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 01 August 2023
DOI 10.3389/fnins.2023.1224752

OPEN ACCESS

EDITED BY
Arindam Basu,
City University of Hong Kong,
Hong Kong SAR, China

REVIEWED BY
Shaista Hussain,
Agency for Science, Technology and Research (A*STAR), Singapore
Sam Neymotin,
Nathan Kline Institute for Psychiatric Research, United States

*CORRESPONDENCE
Shiyan Yang
✉ yangsy@zju.edu.cn

RECEIVED 18 May 2023
ACCEPTED 13 July 2023
PUBLISHED 01 August 2023

CITATION
Yuan Y, Zhu Y, Wang J, Li R, Xu X, Fang T, Huo H, Wan L, Li Q, Liu N and Yang S (2023) Incorporating structural plasticity into self-organization recurrent networks for sequence learning.
Front. Neurosci. 17:1224752.
doi: 10.3389/fnins.2023.1224752

Incorporating structural plasticity into self-organization recurrent networks for sequence learning

Ye Yuan[1], Yongtong Zhu[1], Jiaqi Wang[1], Ruoshi Li[1], Xin Xu[1], Tao Fang[2], Hong Huo[2], Lihong Wan[3], Qingdu Li[1], Na Liu[1] and Shiyan Yang[4]*

[1]School of Health Science and Engineering, Institute of Machine Intelligence, University of Shanghai for Science and Technology, Shanghai, China, [2]Automation of Department, Shanghai Jiao Tong University, Shanghai, China, [3]Origin Dynamics Intelligent Robot Co., Ltd., Zhengzhou, China, [4]Eco-Environmental Protection Institution, Shanghai Academy of Agricultural Sciences, Shanghai, China

Introduction: Spiking neural networks (SNNs), inspired by biological neural networks, have received a surge of interest due to its temporal encoding. Biological neural networks are driven by multiple plasticities, including spike timing-dependent plasticity (STDP), structural plasticity, and homeostatic plasticity, making network connection patterns and weights to change continuously during the lifecycle. However, it is unclear how these plasticities interact to shape neural networks and affect neural signal processing.

Method: Here, we propose a reward-modulated self-organization recurrent network with structural plasticity (RSRN-SP) to investigate this issue. Specifically, RSRN-SP uses spikes to encode information, and incorporate multiple plasticities including reward-modulated spike timing-dependent plasticity (R-STDP), homeostatic plasticity, and structural plasticity. On the one hand, combined with homeostatic plasticity, R-STDP is presented to guide the updating of synaptic weights. On the other hand, structural plasticity is utilized to simulate the growth and pruning of synaptic connections.

Results and discussion: Extensive experiments for sequential learning tasks are conducted to demonstrate the representational ability of the RSRN-SP, including counting task, motion prediction, and motion generation. Furthermore, the simulations also indicate that the characteristics arose from the RSRN-SP are consistent with biological observations.

KEYWORDS

spiking neural network, self-organization, reward-modulated spike timing-dependent plasticity, homeostatic plasticity, structural plasticity

1. Introduction

Spiking neural networks, inspired by biological neural networks, are deemed to possess strong information processing abilities due to temporal encoding (as shown in Figure 1) and variable connection pattern (Zhang et al., 2018b, 2021; Bellec et al., 2020), which are driven by multiple neural plasticities, such as STDP (Frémaux and Gerstner, 2016; Brzosko et al., 2019), structural plasticity (Caroni et al., 2012; Milano et al., 2020), and homeostatic plasticity (Delvendahl and Müller, 2019; Haşegan et al., 2022). STDP enables the network to modulate its connection weight based on spike timing, while homeostatic plasticity can regulate the excitability of neurons within an appropriate range. Structural plasticity can endow the network with robust adaptability by fine-tuning its mesoscopic connection pattern during the lifecycle. However, it is non-trivial to achieve a stable training procedure for spiking neural networks incorporating multiple neural plasticity mechanisms. The temporal encoding of biological neural networks is a sophisticated information encoding method, which needs to cooperate with

various neural plasticities to exert strong information processing ability. Although many sophisticated spiking neuron models have been designed, there is a lack of research on neural plasticities. Many current spiking neural networks are simple abstraction of biological neural networks, which makes it not an easy task to train the network. For example, changes in input may occasionally cause a sharp increase or decrease in the firing rate of neurons, resulting in probabilistic non-convergence (Pfeiffer and Pfeil, 2018; Xing et al., 2019) and a lack of adaptation to input (Wang et al., 2020). To address these questions, it is crucial to understand how these plasticities interact to shape neural networks and affect neural signal processing. But, it is difficult and expensive to directly observe the biological neural network at the mesoscopic level, since it consists of a large number of neurons that are dynamically connected through synapses.

To handle the problem mentioned above, it is a potential way to establish a biologically reasonable spiking neural network model that incorporates multiple neural plasticity mechanisms. For example, Lazar et al. proposed a self-organization recurrent network (SORN) driven by multiple neural plasticities (Lazar et al., 2009), which only consists of a recurrent layer and an output layer. The recurrent layer is mainly adapted by STDP and homeostatic plasticity. STDP is used to adjust synaptic weights based on postsynaptic spike activity. In detail, synaptic weight is strengthened when pre-synaptic spike activity is followed by post-synaptic spike activity, while the reverse pattern makes synaptic weight weak. Homeostatic plasticity induces a competition among synaptic connections and maintains spike firing. The simulation results of SORN show that STDP and homeostatic plasticity lead to some non-statistical characteristics of spiking neural networks, such as lognormal-like distribution of synaptic weights, long-term persistence of strong synaptic connections, and power-law distribution of synaptic lifetimes. Inspired by SORN, Aswolinskiy et al. proposed a reward-modulated self-organization recurrent network (RM-SORN) (Aswolinskiy and Pipa, 2015; Dora et al., 2016), in which synaptic weights are adjusted by R-STDP and homeostatic plasticity. R-STDP refers that the outcome of STDP, induced by pre-synaptic and post-synaptic spike activity, is gated by external reward, and the resulting learning rules are no longer unsupervised (Izhikevich, 2007; Anwar et al., 2022).

Despite the fact that the SORN and RM-SORN models are self-organization networks, their connection patterns do not alter continually during the training phase. It means that these spiking neural network models do not really incorporate structural plasticity, and structural plasticity remains under-explored for existing SNNs. Therefore, in this work, we propose a novel reward-modulated self-organization recurrent network with structural plasticity, in which the connection pattern is continuously adjusted along with the lifecycle. In detail, R-STDP is utilized to generate effective representations for inputs in the recurrent layer, which also helps to achieve efficient mapping in the output layer. Besides, homeostatic plasticity is used to stabilize the excitability of neurons. In particular, structural plasticity is further introduced to simulate the growth and pruning of connections in the recurrent layer, which could well explore the characteristics of structural plasticity for training SNNs. The representational ability of the RSRN-SP is evaluated on three sequence learning tasks, including counting task, motion prediction task, and motion generation task.

In summary, our contributions are as follows: (1) We propose a novel reward-modulated self-organization recurrent network with structural plasticity (RSRN-SP), in which structural plasticity is introduced from neurophysiology to enhance the variability of connection patterns; (2) We experimentally find that structural plasticity could improve the adaptability of the network and reduce the training difficulty; (3) We empirically reveal some characteristics arose from the RSRN-SP are consistent with biological observations, i.e., lognormal-like distribution of connection weight, power-law distribution of connection lifecycle, and a stable tendency for stronger connections; (4) Experiments on three sequence learning tasks show that our method achieve better representation ability than the same type of spiking neural networks such as SORN and RM-SORN. Further analyzes are utilized to demonstrate the effectiveness of structural plasticity.

2. Related works

There have been many researches on spiking neuron models and learning rules (Yu et al., 2014; Zhang et al., 2018a; Ju et al., 2020; Xu et al., 2021), where neuron models, learning rules, and network architectures are three essential factors for designing spiking neural networks.

2.1. Neuron models

The human brain contains billions of neurons, which form structurally complex and computationally efficient networks through dynamic synaptic connections (Bassett and Sporns, 2017). There are various spiking neural models to simulate the temporal coding of neurons in the brain, i.e., Hodgkin-Huxley (HH) model (Izhikevich, 2004), leaky integrate-and-fire (LIF) model (Yu et al., 2014), and binary neuron model (Dayan and Abbott, 2001). The HH model focuses on the microscopic mechanism of spikes, while the LIF model focuses on the computational complexity of spikes. The binary neuron model is developed from the LIF model, has lower computational complexity, and is suitable for building large-scale networks. Therefore, in this work, the binary neuron model is used to construct a spiking neural network.

2.2. Learning rules

Inspired by biological observations, there are mainly two brain-inspired learning rules suitable for SNNs (Caporale and Dan, 2008; Frémaux and Gerstner, 2016): Hebb learning rule and STDP learning rule. The former suggests that neurons activated at the same time should have a closer relationship. The latter indicates that synaptic weight is adjusted based on the spike timing of pre-synaptic and post-synaptic neurons. STDP learning rule can be described as follows,

$$\Delta w_{ij} = \begin{cases} A_+ \exp\left(-\Delta t/\tau_+\right), & \text{if } \Delta t \geq 0 \\ -A_- \exp\left(\Delta t/\tau_-\right), & \text{if } \Delta t < 0 \end{cases} \tag{1}$$

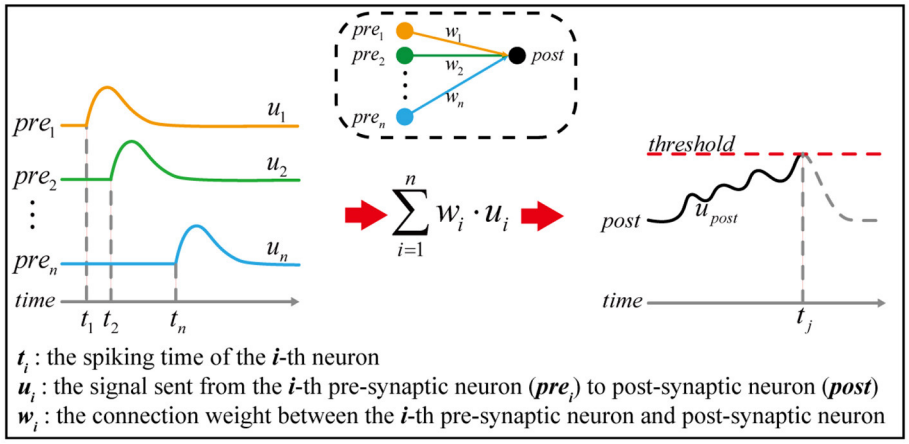

FIGURE 1
The principle of temporal encoding. Given a simple neural network in the dashed box, n pre-synaptic neurons are connected to one post-synaptic neuron. The pre-synaptic neuron pre_i generates a spike at time t_i, which causes a signal u_i continuously sent to the post-synaptic neuron. Once the signal received by the post-synaptic neuron exceeds the threshold, a spike is generated, and the corresponding spiking time is marked as t_j. According to neuroscience, information is thought to be encoded in the spiking time sequence, such as $t_1, t_2, ..., t_n, t_j$.

where Δw_{ij} represents the weight change of connection from pre-synaptic neuron j to post-synaptic neuron i. A_+, A_-, τ_+, and τ_- are dimensionless constants, which are obtained by fitting neurophysiological data. $\Delta t = t_i^f - t_j^f$ represents the error between the last spike timing of post-synaptic neuron i and the last spike timing of pre-synaptic neuron j.

2.3. Spiking neural networks

Spiking neural networks are known as the third-generation neural networks. Due to the brain-like temporal encoding and multiple neural mechanisms, they are considered to have a strong information processing capability. However, this also makes the training of spiking neural networks difficult (Wang et al., 2020). The widely used gradient descent algorithm is difficult to apply to spiking neural networks (Taherkhani et al., 2020). Therefore, many researches combine various mathematical optimization techniques with spiking neural networks, trying to propose a new learning paradigm suitable for SNNs. For example, Xing et al. adopted the ANN-to-SNN strategy to migrate the parameters of the trained ANNs to the SNNs of the same architecture (Xing et al., 2019; Gao et al., 2023). Anwar et al. applied reinforcement learning to spiking neural networks to perform specific tasks, such as Pong and Cartpole game playing (Bellec et al., 2020; Anwar et al., 2022; Haşegan et al., 2022). These spiking neural networks combine biologically reasonable neural mechanisms with reinforcement learning and mathematical optimization to complete sophisticated tasks. Some studies also employ spiking neural networks containing biologically reasonable neural mechanisms to explore the principle of biological neural network information processing. For example, the SORN and RM-SORN models are proposed to explore the coordination of neural mechanisms such as R-STDP, structural plasticity, and homeostatic plasticity (Lazar et al., 2009; Aswolinskiy and Pipa, 2015; Dora et al., 2016).

3. The proposed method

3.1. Problem formulation

3.1.1. Preliminary

Consider a binary neuron model, the neuron state (0 and 1) changes over the inputs. The binary neuron state s_t at discrete time t is updated as follows:

$$s_t = \begin{cases} 1, \text{ if } \psi_t \geq \theta \\ 0, \text{ if } \psi_t < \theta \end{cases} \tag{2}$$

where θ is the threshold and ψ is the sum of inputs. Once ψ reaches the threshold, the neuron state will be activated as 1, otherwise $x = 0$. Besides, ψ is calculated as follows:

$$\psi_i = u_i + \sum_{j \in S} w_{ij} s_i \tag{3}$$

where w_{ij} is the weight between neuron i and neuron j. u_i is the external signal received by neuron i. The notations used in this work are explained in Table 1.

3.1.2. Counting task

The task objective is to predict the subsequent element by modeling the structure of a recurrent sequence. Consider a m times recurrent sequence as follows:

$$\overbrace{a\underbrace{bb\cdots b}_{n}c}^{1}\overbrace{a\underbrace{bb\cdots b}_{n}c}^{2}\cdots\overbrace{a\underbrace{bb\cdots b}_{n}c}^{m} \tag{4}$$

where each subsequence contains a start flag, an end flag and a fixed number of a repeated element (i.e., a, c, and b). Taking an element as the input, the model aims to accurately predict the next element. For example, given a as the input, and the ground-truth is b. This task is designed to test the memory property of the model.

TABLE 1 Notations used in this work. We demonstrate the commonly used notations in this work.

Notation	Interpretation
I/E	Inhibitory/Excitatory neuron
N^I/N^E	Number of inhibitory/excitatory neuron
$s_i^e/s_i^{in}/s_i^o$	State of i-th excitatory/inhibitory/output neuron
$\theta_i^e/\theta_i^{in}/\theta_i^o$	Threshold of i-th excitatory/inhibitory/output neuron
w_{ij}^{ee}	Weight between i-th excitatory and j-th excitatory neuron
w_{ij}^{ei}	Weight between i-th excitatory and j-th inhibitory neuron
w_{ij}^{ie}	Weight between i-th inhibitory and j-th excitatory neuron
w_{ij}^{oe}	Weight between i-th output and j-th excitatory neuron

3.1.3. Motion prediction task

The task objective is to predict the subsequent element by modeling the structure of a recurrent sequence, in which all elements are associated with different spatial positions. Consider a m times recurrent sequence as follows:

$$\overbrace{12\cdots n}^{1}\overbrace{12\cdots n}^{2}\cdots\overbrace{12\cdots n}^{m} \tag{5}$$

where each subsequence contains n integers from 1 to n. Taking an element as the input, the model aims to accurately predict the next element. Because the elements are associated with different spatial positions, this task can be interpreted as the left to right motion of an object along an axis.

3.1.4. Motion generation task

The task objective is to generate the m times recurrent sequence same as (5), with the output serving as the input and no external teaching signals. For example, if the current output of the model is equal to 1 and serves as the next input, the next output should be 2.

3.2. Overview architecture

The overall architecture of RSRN-SP is shown as in Figure 2, which is composed of two general modules, i.e., a recurrent layer, and an output layer.

3.2.1. Recurrent layer

The recurrent layer extracts features of inputs and stores them in its variable connection patterns and weights. The neurons in the layer are divided into two groups: excitatory and inhibitory neurons. N^E and N^I are utilized to denote the numbers of them, $N^I = 0.2 \times N^E$. They are connected through weighted connections, which is denoted as a weight matrix W. The element w_{ij} in the weight matrix is the connection weight from neuron j to neuron i. The connections among excitatory neurons are sparse, while full connections exist between excitatory and inhibitory neurons. The initial connection density among excitatory neurons is controlled by the average connection fraction p_c of neurons. It should be noted that there exist no self-connections and connections among inhibitory neurons.

3.2.2. Output layer

The output layer maps the features stored in the recurrent layer into interpretable and specific-task outputs. It only contains excitatory neurons without connections among each other. There is a feedback connection from the output layer to the recurrent layer.

3.3. Evolution

The input of RSRN-SP is a time sequence that contains different symbols (letters or digits). Each symbol corresponds to a subset of neurons in the recurrent layer, which and only which can receive the corresponding input symbol. When a certain symbol in the sequence is input to the model, only neurons in the corresponding subset are activated, while other neurons remain silent. The state updating for different types of neurons are as follows:

$$\begin{aligned} &s_k^e(t+1) \\ =&\Theta\Big(\sum_{i=1}^{N^E} w_{ki}^{ee} s_i^e(t) - \sum_{j=1}^{N^I} w_{kj}^{ei} y_k(t) + u_k(t) - \theta_k^e(t)\Big), \end{aligned} \tag{6}$$

$$s_k^{in}(t+1) = \Theta\Big(\sum_j^{N^E} w_{kj}^{ie} s_j^e(t) - \theta_k^{in}(t)\Big). \tag{7}$$

$$s_k^o(t+1) = \Theta(\sum_j^{N^E} w_{kj}^{oe} s_j^e(t) - \theta_k^o(t)) \tag{8}$$

where Θ is the Heaviside step function, and $u_k(t)$ is the external signal of neuron k at time t. The weights are uniformly drawn from $[0, 1]$ and initially normalized as $\sum_j^N w_{ij} = 1$.

3.4. Learning rule of R-STDP

According to the learning rule of R-STDP (Frémaux and Gerstner, 2016; Yuan et al., 2018), the change of connection weight is not only controlled by spikes, but also regulated by reward signal, as illustrated in Figure 2B, which can be described as follows:

$$W(t+1) = W(t) + \eta \cdot M \times e. \tag{9}$$

Where η is the learning rate of the weight. e denotes a synaptic eligibility trace to temporarily store the outcome of R-STDP, which can be still available for a delayed reward signal. The eligibility trace is computed as:

$$\begin{aligned} \frac{de_{ij}}{dt} = &-\frac{e_{ij}}{\tau_e} + \big[s_i(t) \cdot s_j(t-1) \\ &-f \cdot s_i(t-1) \cdot s_j(t)\big] \end{aligned} \tag{10}$$

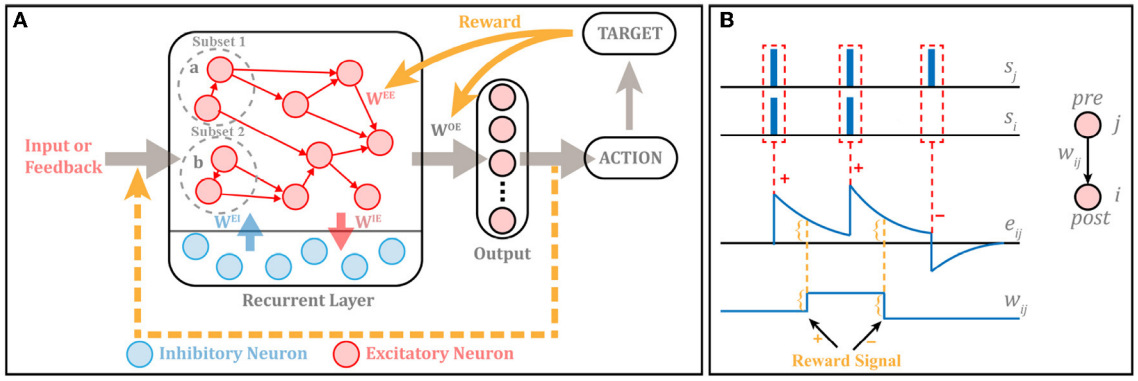

FIGURE 2
(A) The structure of RSRN-SP. Excitatory-to-Excitatory (E-E) connections W^{EE} are sparse. Excitatory-to-Inhibitory (E-I) connections W^{IE}, Inhibitory-to-Excitatory (I-E) connections W^{EI} and connections from the recurrent to output layer W^{OE} are full. W^{EE} and W^{OE} follow R-STDP, while others keep fixed. Suppose that the input is a m times recurrent sequence (i.e., *abab...ab*), in which *a* and *b* correspond to subset 1 and 2, respectively. When *a* is input into the model, only the neurons of the subset 1 are activated, likewise *b* is input into the model, only the neurons of the subset 2 are activated. **(B)** Illustration of R-STDP. The outcome of R-STDP is stored in eligibility trace. The updating of the weights occurs only at the time of reward.

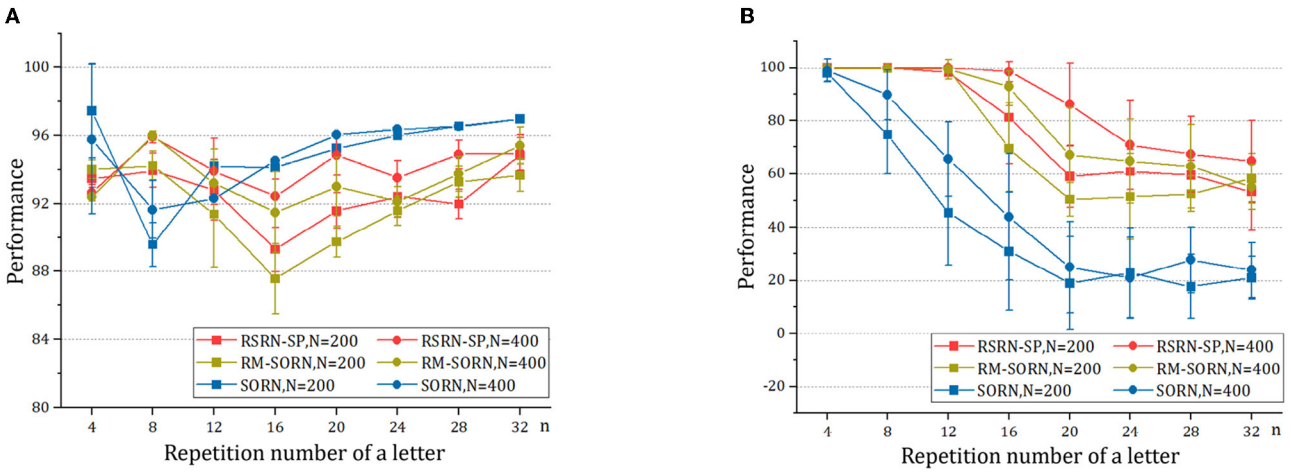

FIGURE 3
The results of the counting task. **(A)** The overall performance. **(B)** The counting performance.

where s_i and s_j are the activation of pre- and post-synaptic neurons, respectively. τ_e is a time constant. f is a dimensionless parameter ($f = 1$ for W^{EE} and $f = 0.01$ for W^{OE}). According to Equation (9), when neuromodulation factor $M = R - b$ is not equal to 0, i.e., reward R deviates from baseline b, the connection weight will be updated.

In our model, for the counting task and motion prediction task, the reward R is set to 1 for correct output, while either 0 or -1 for incorrect output. The baseline b is set to the moving average of R. For the generation task, when the target sequence is correctly generated, the reward R that is proportional to the length of the correctly generated sequence will be given.

3.5. Structural plasticity

Structural plasticity is a fundamental neural mechanism of the biological neural network in the brain, which is demonstrated to have a critical role in regulating circuit connection during learning (Caroni et al., 2012). Structural plasticity refers that old synaptic connections may be pruned and new synaptic connections formed during the self-organization of neural networks (Lamprecht and LeDoux, 2004). In our model, we apply structural plasticity to the connections among excitatory neurons. New connections will be added between two unconnected neurons with a probability $p_{sp} \in (0, 1)$, and their weights are initialized as 0.001. p_{sp} is fine-tuned as a hyper-parameter to stabilize the recurrent layer. Old

connections will be pruned if their weights are less than a near-zero threshold $w^{th} \in (0, 1)$.

Structural plasticity can be formulated as follows:

$$\begin{cases} c_{ij} = 0, \text{ if } c_{ij} = 1 \text{ and } w_{ij} < w^{th} \\ c_{ij} = 1, w_{ij} = 0.001, \text{ if } c_{ij} = 0 \text{ and } rnd > 1 - p_{sp} \end{cases} \tag{11}$$

where $c_{ij} \in \{0, 1\}$ indicates whether a connection from neuron j to neuron i exists or not. If $c_{ij} = 1$, the connection exists; if $c_{ij} = 0$, the connection does not exist. $rnd \in (0, 1)$ is a uniformly distributed random number. w_{ij} denotes the weight of connection from neuron j to neuron i.

3.6. Homeostatic plasticity

Homeostatic plasticity is critical to alleviate the instability of neural networks. Two common homeostatic mechanisms are utilized in our method: synaptic normalization and intrinsic plasticity. The synaptic normalization is formulated as:

$$w_{ij}(t) = \frac{w_{ij}(t)}{\sum_j w_{ij}(t)} \longrightarrow \sum_j w_{ij}(t) = 1. \tag{12}$$

Where the weights of all afferent connections to a neuron are proportionally scaled to make their sum equal to 1. Synaptic normalization can promote healthy competition among connections that connect to the same neuron.

The intrinsic plasticity enables to adjust the thresholds of excitatory neurons by an average firing rate μ_{ip}, which can be formulated as follows:

$$\Delta\theta_i^e(t) = \eta(s_i(t) - \mu_{ip}). \tag{13}$$

Where η_{ip} is the learning rate of the threshold. For excitatory neurons in the recurrent layer, μ_{ip} is fine-tuned in a range of $[0.05, 0.25]$. In the output layer, μ_{ip} is uniquely set for each neuron, corresponding to the expected occurrence probability of the symbol represented by the neuron. Due to the intrinsic plasticity, the threshold of a neuron in our model will increase if it is too active; otherwise, the threshold will decrease.

3.7. Two-stage training

A two-stage training scheme is proposed for RSRN-SP. In the first stage, the model is trained with R-STDP, homeostatic plasticity, and structural plasticity. In the second stage, the connection pattern and weight of the recurrent layer are fixed. The connection weight of the output layer is fine-tuned for specific tasks. The first stage is alternated with the second stage. In each alternation, the model takes about 100 steps at the first stage, and then proceeds to the second stage to take about 20,000 steps. During training, the alternation will be repeated about 200 times. During inference, the model will be evaluated on testing data. For the motion generation task, the output of the model is fed back as its input during training, and the model is used to generate desired output during inference. The algorithm for the first stage is listed here. The second-stage algorithm is similar, which only applies R-STDP and homeostatic plasticity at the output layer.

Require: Sequence that meets the counting task, motion prediction task, and motion generation task.
Ensure: Letter or integer predicted or generated based on the sequence.
Initialize the network parameters: neuron number N, connection weight W, neuron state S, state threshold Θ, etc.
for t = 0 to 100 **do**
  for k = 1 to N **do**
    compute $s_k^e(t+1)$, $s_k^i(t+1)$, and $s_k^o(t+1)$
  end for
  for i = 1 to N^E **do**
    for j = 1 to N^E **do**
      if connection from j^{th} to i^{th} neurons exists **then**
        compute e_{ij} based on Equation (10)
        $w_{ij}^{ee}(t+1) \leftarrow w_{ij}^{ee}(t) + \eta * M * e_{ij}$
      end if
    end for
  end for
  for i = 1 to N^E **do**
    $w_{ij}(t+1) \leftarrow w_{ij}(t+1) / \sum_j w_{ij}(t+1)$
    $\theta_i^e(t+1) \leftarrow \theta_i^e(t) + \eta\left(s_i^e(t) - \mu_{ip}\right)$
  end for
  for i = 1 to N^E **do**
    for j = 1 to N^E **do**
      if $c_{ij} == 1$ **and** $w_{ij}^{ee}(t+1) < w^{th}$ **then**
        $c_{ij} \leftarrow 0$
      end if
      if $c_{ij} == 0$ **and** $rnd > 1 - p_{sp}$ **then**
        $c_{ij} \leftarrow 1$
        $w_{ij}^{ee}(t+1) \leftarrow 0.001$
      end if
    end for
  end for
end for

Algorithm 1. The first-stage algorithm of RSRN-SP.

4. Experiments

4.1. Experimental settings

Our model is evaluated on three tasks: counting task (Lazar et al., 2009), motion prediction (Aswolinskiy and Pipa, 2015), and motion generation (Aswolinskiy and Pipa, 2015). The comparison approaches mainly involve SORN (Lazar et al., 2009) and RM-SORN (Aswolinskiy and Pipa, 2015). Notably, in SORN and RM-SORN, the weights to the output layer are trained with linear

regression, and there is no structural plasticity applied in the recurrent layer. The model is implemented in Python programming on a Windows 10 computer with NVIDIA RTX 1080Ti. Source code and parameters are released at Github.

4.2. Experiments for counting task

4.2.1. Evaluation protocol

Two kinds of evaluation protocols are used: (1) overall performance is to evaluate the matching of all letters in a sequence; (2) counting performance is to evaluate the prediction accuracy for all subsequences in a sequence.

TABLE 2 The results of the counting task.

Overall performance	n				
$N = 200$	4	8	12	16	20
RM-SORN	94.02	94.23	91.39	87.56	89.73
RSRN-SP (Ours)	93.48	93.96	**92.82**	**89.28**	**91.59**
$N = 400$	4	8	12	16	20
SORN	95.78	91.66	92.31	94.52	96.04
RSRN-SP (Ours)	92.64	**95.92**	**93.91**	92.43	94.58
Counting performance	n				
$N = 200$	4	8	12	16	20
RM-SORN	99.70	99.50	99.31	69.50	50.44
RSRN-SP (Ours)	**100.00**	**100.00**	**98.18**	**81.17**	**58.88**
$N = 400$	4	8	12	16	20
SORN	98.87	89.68	65.44	43.89	24.95
RSRN-SP (Ours)	**100.00**	**100.00**	**100.00**	**98.38**	**86.04**

The bold values indicate that the method outperforms other models under the same conditions.

4.2.2. Results

The results for the counting task are shown in Figure 3 and Table 2. It can be observed that as the number n of the repetition of a letter in a subsequence increases, the overall performance fluctuates around 90% within a relatively narrow range, while the counting performance declines. This is because the value of n is proportional to the number of input patterns that the recurrent layer needs to learn. Larger n increases the difficulty of predicting the last letter of a subsequence, but reduces the difficulty of predicting the other letters of this subsequence. The counting performance gap among our model, SORN, and RM-SORN can be explained by the difference between reward-modulated learning and offline linear regression. SORN and RM-SORN try to learn all separable input patterns and minimize global mapping errors, whereas the reward of our model is computed as moving average within a time window, in which each individual input pattern is more effectively learned.

4.3. Experiments for motion prediction task

4.3.1. Results

As shown in Figure 4A and Table 3, when n is small, our model, SORN and RM-SORN have high prediction accuracy. As

TABLE 3 The results of the motion prediction task.

$N = 200$	n				
	4	8	12	16	20
RM-SORN	98.79	98.04	95.42	90.46	86.80
RSRN-SP (Ours)	**99.38**	**98.63**	**97.87**	**93.51**	**89.32**
$N = 400$	4	8	12	16	20
RM-SORN	99.46	99.08	98.57	96.96	93.92
RSRN-SP (Ours)	**99.83**	**99.28**	**99.07**	**97.99**	**95.57**

The bold values indicate that the method outperforms other models under the same conditions.

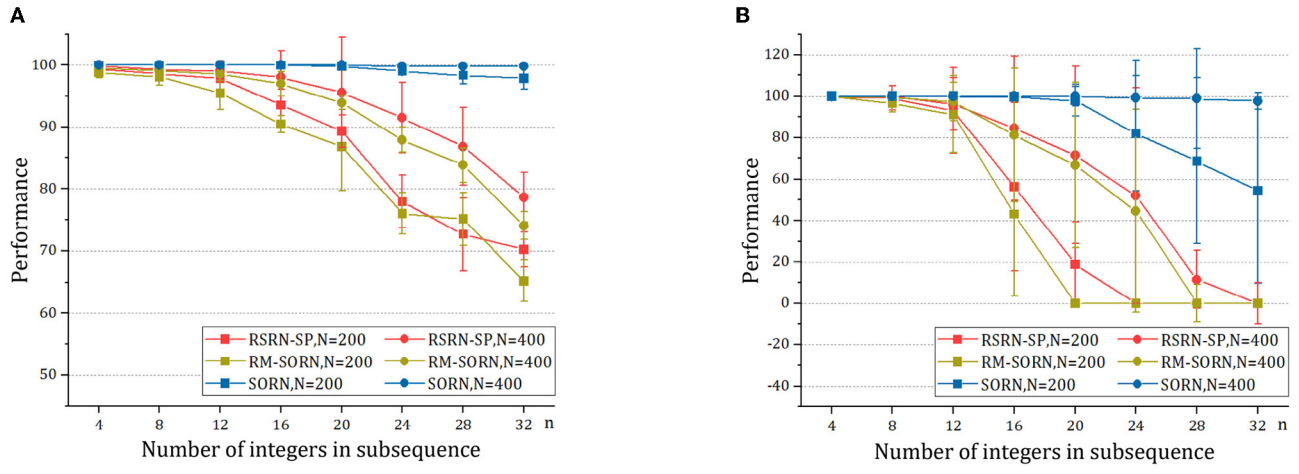

FIGURE 4
The results of the motion prediction and generation tasks. **(A)** The motion prediction task. **(B)** The motion generation task.

n increases, the accuracy of the three models gradually decreases, and the accuracy of our model becomes higher than RM-SORN and lower than SORN. It is worth noting that in our model, increasing the number of neurons can greatly increase the performance of the model. Considering that our model has much less training difficulty than SORN, it can achieve an accuracy similar to SORN by increasing the number of neurons when n is very big.

4.4. Experiments for motion generation task

4.4.1. Evaluation protocol

The performance is calculated as the percentage of the symbols belonging to the target sequence to the total number of symbols. For example, assuming that the desired sequence is 1234. If the desired sequence is generated, the model receives the full reward of unit 1. Otherwise, it receives the reward of $\frac{3}{4}$ for the sequence $x123$, $\frac{2}{4}$ for the sequence $xx12$ and $\frac{1}{4}$ for the sequence $xxx1$.

4.4.2. Results

As shown in Figure 4B and Table 4, without external teaching signals, our model can still generate the desired sequence accurately. Success in this task shows that the model can generate an arbitrary sequence with the same symbol distribution as in the motion sequence.

As a result of the self-organization driven by multiple plasticity, the recurrent layer can an create effective representation of inputs. The model containing 400 neurons outperforms that containing 200 neurons, suggesting that the memory capacity of the model is closely related to the number of neurons.

TABLE 4 The results of the motion generation task.

$N = 200$	n				
	4	8	12	16	20
RM-SORN	99.79	96.66	91.24	43.20	0.00
RSRN-SP (Ours)	**99.90**	**98.98**	**93.13**	**56.40**	**18.78**
$N = 400$	4	8	12	16	20
RM-SORN	99.92	99.66	97.24	81.20	66.70
RSRN-SP (Ours)	**100.00**	**99.98**	**96.13**	**84.64**	**71.54**

The bold values indicate that the method outperforms other models under the same conditions.

4.5. Influence of structural plasticity

To study structural plasticity, the models with different N and p_c are constructed in the counting task. Figure 5 and Table 5 suggest that the recurrent layer with structural plasticity outperforms that without structural plasticity. The performance improvement is larger when the recurrent layer is initialized with sparse connectivity. The recurrent layer with structural plasticity has great advantage, which is prominent when the connection patterns are not reasonably initialized. In the case of initial $p_c = 0.002$, the

TABLE 5 The changes of E-E connection fractions of RSRN-SP on the counting tasks at four initial connection fractions.

$p_c^{(0)}$	$N = 200$		$N = 400$	
	$\Delta p_c/p_c^{(0)}$	$p_c^{(1)}$	$\Delta p_c/p_c^{(0)}$	$p_c^{(1)}$
0.002	+(459 ± 242)%	0.011	+(305 ± 223)%	0.008
0.01	+(53 ± 29)%	0.015	+(53 ± 20)%	0.015
0.1	−(37 ± 11)%	0.063	−(41 ± 13)%	0.059
0.2	−(46 ± 10)%	0.108	−(50 ± 11)%	0.099

$p_c^{(0)}$ is the initial connection fraction.
$p_c^{(1)}$ is the connection fraction after the training, which is computed as the average of the model on the counting tasks ($n = [4, 8, .., 32]$).
The relative change of $\Delta p_c/p_c^{(0)}$ is denoted as "(mean ± std)".

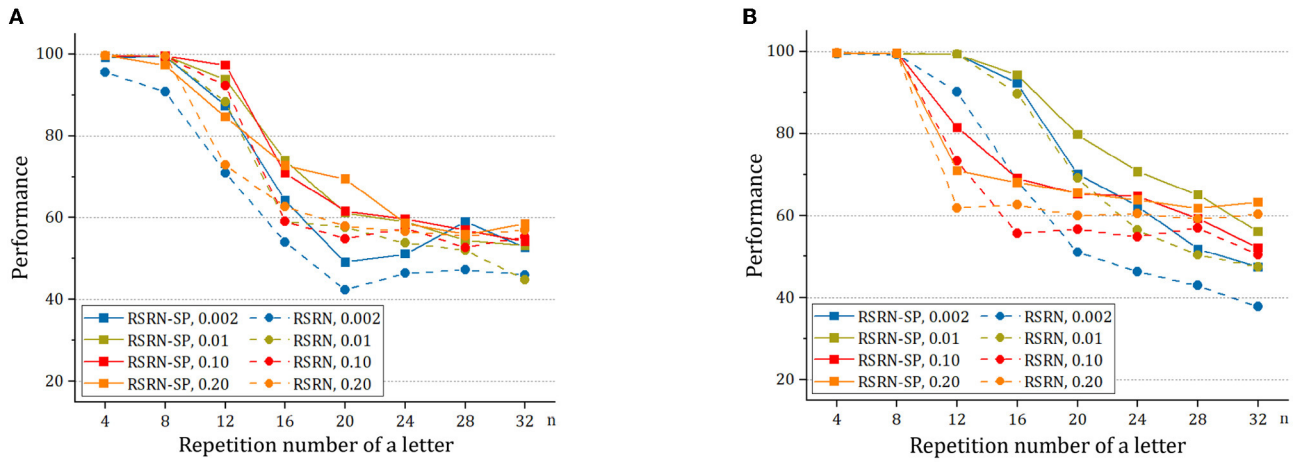

FIGURE 5
The performance of RSRN with and without structural plasticity (SP), with initial $p_c = 0.002, 0.01, 0.1, 0.2$, on the counting task. **(A)** The network consisting of $N = 200$ neurons. **(B)** The network consisting of $N = 400$ neurons.

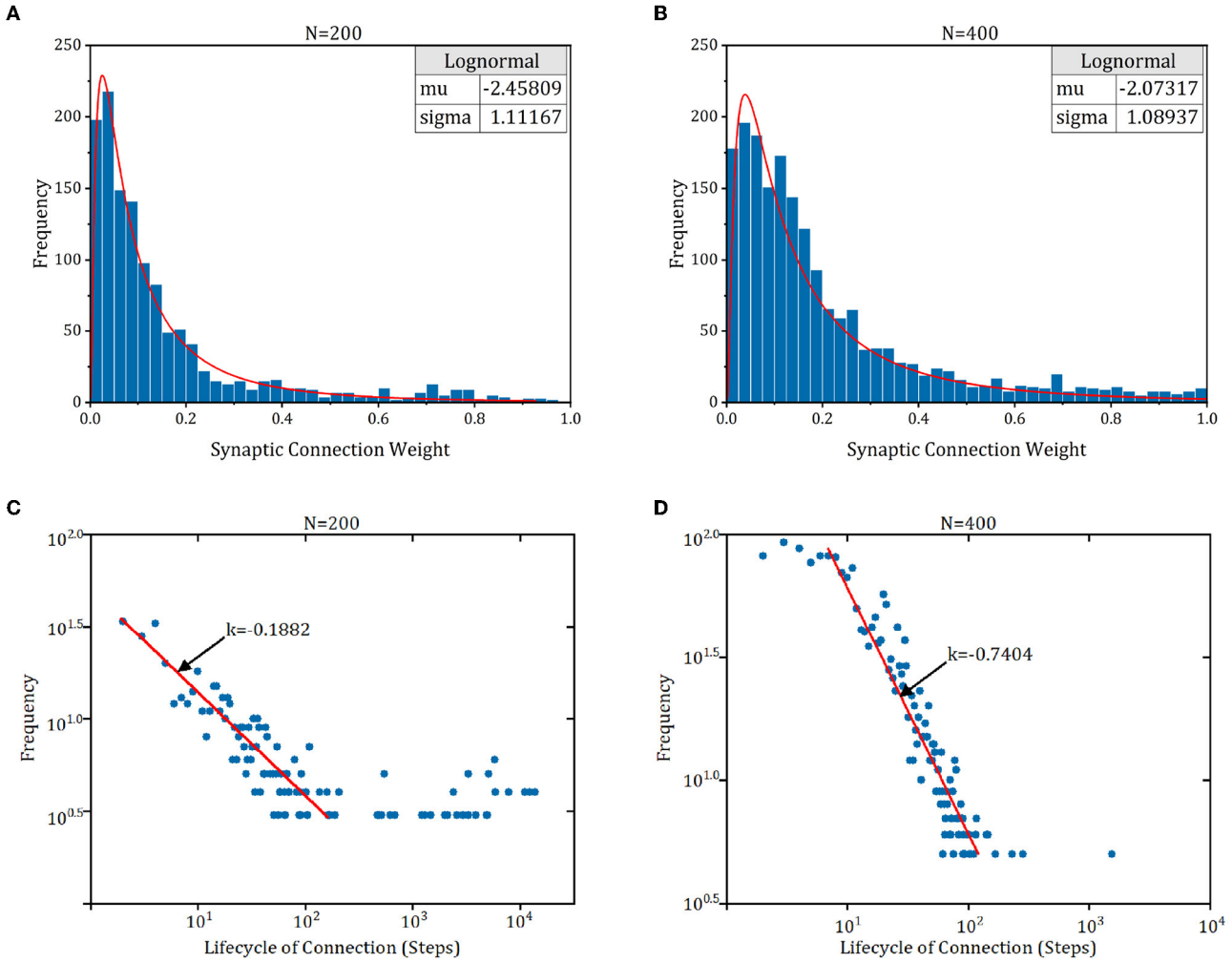

FIGURE 6
The synaptic connection characteristics of RSRN-SP. **(A, B)** Synaptic weight distribution. Blue bars are synaptic weight distribution histogram. Red line is lognormal distribution fit curve. **(C, D)** Synaptic lifecycle distribution. Red line is distribution fit curve.

connection fractions change into 0.011 and 0.008 after training, for the model of $N = 200$ and $N = 400$, respectively. When the initial p_c is set to 0.01, the increment is much smaller. The decrease is much smaller for $p_c = 0.1$, compared to the case of $p_c = 0.2$. It is testified that the model of $N = 200$ and $N = 400$ achieved the best performance with $p_c = 0.05$ and $p_c = 0.0125$, respectively.

4.6. Synaptic connection characteristics of RSRN-SP

The connection pattern of cortex exhibits some fundamental characteristics (Zheng et al., 2013), e.g., lognormal-like distribution of synaptic weight, power-law distribution of synaptic lifecycle, and a tendency for stronger connections to be more stable. To study whether these characteristics exist in RSRN-SP, we simulated a model of 200 and 400 on the counting task of $n = 8$. As shown in Figures 6A, B, the synaptic weights exhibit lognormal-like distribution, which is consistent with biological observations (Song et al., 2005; Loewenstein et al., 2011). Figures 6C, D demonstrated that the distribution of lifecycle of newly formed connections can be roughly fitted by a power law. Most of newly formed connections tend to disappear and only a few of them can persist and become strong.

5. Discussion and conclusion

To understand how multiple plasticities interact to shape biological neural networks and affect neural signal processing, we proposed a novel spiking neural network incorporating with multiple neural plasticity from neurophysiology, e.g., reward-modulated spike timing-dependent plasticity, homeostatic plasticity, and structural plasticity. In particular, homeostatic plasticity and reward-modulated spike timing-dependent plasticity are used to promote the consistency between the network updating and brain learning, which help to guide the updating of connection weight during training SNNs. Specially, structural plasticity is introduced to simulate the growth and pruning of connections in

the network, which could guarantee the consistency between the network structure and brain structure.

Here, our work attempts to combine R-STDP with other plasticity mechanisms to achieve better training results. The simulations demonstrated that (1) reward-modulated spike timing-dependent plasticity, structural plasticity, and homeostatic plasticity can work in coordination to empower neural networks to learn; (2) structural plasticity weakens the network connection stability but enhances its ability to adapt to the input; (3) RSRN-SP could effectively learn the representation of the input, and achieves better performance on sequence learning tasks than the same type of spiking neural network including SORN and RM-SORN. Furthermore, the simulations also indicate that the characteristics arose from RSRN-SP are consistent with biological observations.

Compared to the widely used artificial neural networks, our spiking neural network is not easy to train due to complex temporal encoding, variable connection pattern, and diverse plasticity mechanisms. One challenge stems from the temporal encoding, which allows information to be processed in the form of spikes in SNNs. However, spikes are not mathematically differentiable, making it difficult to apply traditional gradient-based optimization algorithms. The generation of new connections and the disappearance of old connections also increase the difficulty of network training. To address these challenges, some studies have explored R-STDP (Frémaux and Gerstner, 2016), which is considered a biologically plausible learning algorithm suitable for SNNs. Nevertheless, how efficient learning of SNNs can be achieved by R-STDP, while maintaining sustained balanced network activity remains an open question.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YY: software, formal analysis, and writing—original draft. RL, TF, QL, and HH: writing—reviewing and editing. XX and LW: data curation. NL: supervision and funding acquisition. All authors contributed to the article and approved the submitted version.

Funding

This study was supported by the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 62206175), the Pujiang Talents Plan of Shanghai (Grant No. 2019PJD035), and the Artificial Intelligence Innovation and Development Special Fund of Shanghai (Grant No. 2019RGZN01041).

Conflict of interest

LW was employed by Origin Dynamics Intelligent Robot Co., Ltd.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Anwar, H., Caby, S., Dura-Bernal, S., D'Onofrio, D., Hasegan, D., Deible, M., et al. (2022). Training a spiking neuronal network model of visual-motor cortex to play a virtual racket-ball game using reinforcement learning. *PLoS ONE* 17, e0265808. doi: 10.1371/journal.pone.0265808

Aswolinskiy, W., and Pipa, G. (2015). RM-SORN: a reward-modulated self-organizing recurrent neural network. *Front. Comput. Neurosci.* 9, 36. doi: 10.3389/fncom.2015.00036

Bassett, D. S., and Sporns, O. (2017). Network neuroscience. *Nat. Neurosci.* 20, 353–364. doi: 10.1038/nn.4502

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., et al. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* 11, 3625. doi: 10.1038/s41467-020-17236-y

Brzosko, Z., Mierau, S. B., and Paulsen, O. (2019). Neuromodulation of spike-timing-dependent plasticity: past, present, and future. *Neuron* 103, 563–581. doi: 10.1016/j.neuron.2019.05.041

Caporale, N., and Dan, Y. (2008). Spike timing–dependent plasticity: a Hebbian learning rule. *Annu. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Caroni, P., Donato, F., and Muller, D. (2012). Structural plasticity upon learning: regulation and functions. *Nat. Rev. Neurosci.* 13, 478–490. doi: 10.1038/nrn3258

Dayan, P., and Abbott, L. F. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press.

Delvendahl, I., and Müller, M. (2019). Homeostatic plasticity - a presynaptic perspective. *Curr. Opin. Neurobiol.* 54, 155–162. doi: 10.1016/j.conb.2018.10.003

Dora, S., Subramanian, K., Suresh, S., and Sundararajan, N. (2016). Development of a self-regulating evolving spiking neural network for classification problem. *Neurocomputing* 171, 1216–1229. doi: 10.1016/j.neucom.2015.07.086

Frémaux, N., and Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circ.* 9, 85. doi: 10.3389/fncir.2015.00085

Gao, H., He, J., Wang, H., Wang, T., Zhong, Z., Yu, J., et al. (2023). High-accuracy deep ANN-to-SNN conversion using quantization-aware training framework and calcium-gated bipolar leaky integrate and fire neuron. *Front. Neurosci.* 17, 1141701. doi: 10.3389/fnins.2023.1141701

Haşegan, D., Deible, M., Earl, C., D'Onofrio, D., Hazan, H., Anwar, H., et al. (2022). Training spiking neuronal networks to perform motor control using reinforcement and evolutionary learning. *Front. Comput. Neurosci.* 16, 1017284. doi: 10.3389/fncom.2022.1017284

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* 15, 1063–1070. doi: 10.1109/TNN.2004.832719

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cereb. Cortex* 17, 2443–2452. doi: 10.1093/cercor/bhl152

Ju, X., Fang, B., Yan, R., Xu, X., and Tang, H. (2020). An fpga implementation of deep spiking neural networks for low-power and fast classification. *Neural Comput.* 32, 182–204. doi: 10.1162/neco_a_01245

Lamprecht, R., and LeDoux, J. (2004). Structural plasticity and memory. *Nat. Rev. Neurosci.* 5, 45–54. doi: 10.1038/nrn1301

Lazar, A., Pipa, G., and Triesch, J. (2009). SORN: a self-organizing recurrent neural network. *Front. Comput. Neurosci.* 3, 23. doi: 10.3389/neuro.10.023.2009

Loewenstein, Y., Kuras, A., and Rumpel, S. (2011). Multiplicative dynamics underlie the emergence of the log-normal distribution of spine sizes in the neocortex *in vivo*. *J. Neurosci.* 31, 9481–9488. doi: 10.1002/aisy.202000096

Milano, G., Pedretti, G., Fretto, M., Boarino, L., Benfenati, F., Ielmini, D., et al. (2020). Brain-inspired structural plasticity through reweighting and rewiring in multi-terminal self-organizing memristive nanowire networks. *Adv. Intell. Syst.* 2, 2000096. doi: 10.3389/fnins.2018.00774

Pfeiffer, M., and Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci.* 12, 774.

Song, S., Sjöström, P. J., Reigl, M., Nelson, S., and Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol.* 3, e68. doi: 10.1371/journal.pbio.0030068

Taherkhani, A., Belatreche, A., Li, Y., Cosma, G., Maguire, L. P., and McGinnity, T. M. (2020). A review of learning in biologically plausible spiking neural networks. *Neural Netw.* 122, 253–272. doi: 10.1016/j.neunet.2019.09.036

Wang, X., Lin, X., and Dang, X. (2020). Supervised learning in spiking neural networks: a review of algorithms and evaluations. *Neural Netw.* 125, 258–280. doi: 10.1016/j.neunet.2020.02.011

Xing, F., Yuan, Y., Huo, H., and Fang, T. (2019). "Homeostasis-based CNN-to-SNN conversion of inception and residual architectures," in *International Conference on Neural Information Processing* (Sydney, NSW: Springer), 173–184.

Xu, Q., Shen, J., Ran, X., Tang, H., Pan, G., and Liu, J. K. (2021). Robust transcoding sensory information with neural spikes. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 1935–1946. doi: 10.1109/TNNLS.2021.3107449

Yu, Q., Tang, H., Tan, K. C., and Yu, H. (2014). A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing* 138, 3–13. doi: 10.1016/j.neucom.2013.06.052

Yuan, Y., Huo, H., Zhao, P., Liu, J., Liu, J., Xing, F., et al. (2018). Constraints of metabolic energy on the number of synaptic connections of neurons and the density of neuronal networks. *Front. Comput. Neurosci.* 12, 91. doi: 10.3389/fncom.2018.00091

Zhang, T., Jia, S., Cheng, X., and Xu, B. (2021). Tuning convolutional spiking neural network with biologically plausible reward propagation. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 7621–7631. doi: 10.1109/TNNLS.2021.3085966

Zhang, T., Zeng, Y., Zhao, D., and Shi, M. (2018a). "A plasticity-centric approach to train the non-differential spiking neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, T., Zeng, Y., Zhao, D., and Xu, B. (2018b). "Brain-inspired balanced tuning for spiking neural networks," in *IJCAI* (Stockholm), 1653–1659.

Zheng, P., Dimitrakakis, C., and Triesch, J. (2013). Network self-organization explains the statistics and dynamics of synaptic connection strengths in cortex. *PLoS Comput. Biol.* 9, e1002848. doi: 10.1371/journal.pcbi.1002848