RESEARCH ARTICLE

An Efficient Supervised Training Algorithm for Multilayer Spiking Neural Networks

Xiurui Xie[1], **Hong Qu**[1,2,3]*, **Guisong Liu**[1], **Malu Zhang**[1], **Jürgen Kurths**[2,3]

1 Department of Computer Science and Engineering, University of Electronic Science and Technology of China, 611731, Chengdu, Sichuan, China, **2** Department of Physics, Humboldt University, 12489, Berlin, Berlin, Germany, **3** Potsdam Institute for Climate Impact Research(PIK), 14473 Potsdam, Germany

* hongqu@uestc.edu.cn

OPEN ACCESS

Citation: Xie X, Qu H, Liu G, Zhang M, Kurths J (2016) An Efficient Supervised Training Algorithm for Multilayer Spiking Neural Networks. PLoS ONE 11(4): e0150329. doi:10.1371/journal.pone.0150329

Editor: Gennady Cymbalyuk, Georgia State University, UNITED STATES

Received: November 8, 2015

Accepted: February 11, 2016

Published: April 4, 2016

Data Availability Statement: All relevant data not included in the paper are available from the UCI mechanism repository. The Iris data set and its description is available from http://archive.ics.uci.edu/ml/datasets/Iris. The Breast Cancer Wisconsin (Original) Data Set is available from http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29.

Funding: This work was partially supported by the National Science Foundation of China and the China Scholarship Council under Grants 61273308 and 61573081 (http://www.nsfc.gov.cn/). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

The spiking neural networks (SNNs) are the third generation of neural networks and perform remarkably well in cognitive tasks such as pattern recognition. The spike emitting and information processing mechanisms found in biological cognitive systems motivate the application of the hierarchical structure and temporal encoding mechanism in spiking neural networks, which have exhibited strong computational capability. However, the hierarchical structure and temporal encoding approach require neurons to process information serially in space and time respectively, which reduce the training efficiency significantly. For training the hierarchical SNNs, most existing methods are based on the traditional back-propagation algorithm, inheriting its drawbacks of the gradient diffusion and the sensitivity on parameters. To keep the powerful computation capability of the hierarchical structure and temporal encoding mechanism, but to overcome the low efficiency of the existing algorithms, a new training algorithm, the Normalized Spiking Error Back Propagation (NSEBP) is proposed in this paper. In the feedforward calculation, the output spike times are calculated by solving the quadratic function in the spike response model instead of detecting postsynaptic voltage states at all time points in traditional algorithms. Besides, in the feedback weight modification, the computational error is propagated to previous layers by the presynaptic spike jitter instead of the gradient decent rule, which realizes the layer-wised training. Furthermore, our algorithm investigates the mathematical relation between the weight variation and voltage error change, which makes the normalization in the weight modification applicable. Adopting these strategies, our algorithm outperforms the traditional SNN multi-layer algorithms in terms of learning efficiency and parameter sensitivity, that are also demonstrated by the comprehensive experimental results in this paper.

Introduction

Increasing the level of realism in a neural simulation and improving the computational capability of artificial neural networks [1][2], the spiking neural networks employing temporal coding mechanism is introduced as the third generation of neural networks and has achieved great

Competing Interests: The authors have declared that no competing interests exist.

success in various artificial intelligence tasks [3]–[6]. Most traditional neural networks represent real-valued analog data by the firing rate of neurons, like the first generation neural networks pioneered by the McCulloch-Pitts model [7] and the second generation by the the perceptron model [8]. However, there are substantial evidences that in biological neural systems there exist fast computations that are very likely based on spike firing events [1][2][9]. To simulate these firing events, the third generation of neural networks, the SNNs transmit information by spike times instead of the firing rate, and have been proven computationally more powerful than networks with rate coding [10]–[13].

In the learning of SNNs with temporal encoding mechanism, the supervised training is an important biomimetic concept which could potentially improve the learning speed with the help of an instructor signal. Various supervised training algorithms of SNNs have been proposed by now, which can broadly be subdivided into two classes: training algorithms for single layer SNNs, and for multilayers.

The single layer training algorithms are introduced based on the gradient decent rule or learning windows. Regarding to the gradient decent rule, the Tempotron [14] is a classical algorithm employing the distance between the output neuron's voltage and the firing threshold as the cost function and can complete training efficiently, however, it can only complete binary classification tasks. The Chronotron [15] and spike pattern association neuron algorithm [16] try to minimize the distance between the desired and actual output spike trains by the gradient descent rule, with the distance defined by the Victor and Purpura metric [17] and the van Rossum metric [18] respectively.

A lot of algorithms based on learning windows have been proposed [19] for single layer networks. Among which, the remote supervised learning method is a classical one employing both the Spike-Timing-Dependent Plasticity (STDP) window, and the anti-STDP learning window to complete training [20]. The perceptron-based spiking neuron learning rule adopts a learning window based on the postsynaptic voltage function to instruct training [21]. The Synaptic Weight Association Training (SWAT) utilizes the STDP learning window and the Bienenstock-Cooper-Munro learning rule [22] to drive learning and achieves convergence. The precise spike driven synaptic plasticity learning rule [23] combines the Windrow-Hoff rule and the learning window of postsynaptic potential. Further algorithms adopting learning windows are introduced in [24]. These training algorithms employing learning windows are often more efficient than those with the gradient descent rule. But these single layer algorithms cannot complete training when the network structure contains hidden layers. However, electrophysiology experiments on cat's visual system and monkey striate cortex reveal that the information in biological neurons is processed hierarchically rather than by a single layer [25]–[27]. Then, training a hierarchical spiking neural network is by far the closest way to the biological system.

For multilayers learning of the SNNs, the Spike Propagation (SpikeProp) [28] is the pioneer method that defines the computational error by the distance between the actual and target firing time, and minimizes the error by the gradient descent rule. It achieves training accurately but inefficiently, and only the first spike of a neuron can be trained. Different variations of the SpikeProp, the Quick Propagation, Resilient Propagation [29] and the Multiple SpikeProp [30] [31] are proposed to improve the SpikeProp's learning performance. The Multi-layer Remote Supervised Learning Method(Multi-ReSuMe) [32] extends the ReSuMe [20] to multiple layers by the gradient decent rule, assuming that the relation between the input and output firing rates is linear. All of these existing algorithms can achieve learning, while the efficiency of them is much lower than that of the biological system [33][34], and does not meet the requirements of the real-time applications.

To solve the low efficiency problem in the multilayer training of SNNs, the Normalized Spiking Error Back-Propagation (NSEBP) is proposed in this paper, which is motivated by the selective attention mechanism of the primate visual system [35, 36] and its layer-wise feature representation method in hierarchical structures [25, 26]. Different from traditional algorithms, our algorithm only selects target spike times as attention areas and ignores the states of other times. Besides, the voltage difference is employed to evaluate training errors, and the relation between the weight variation and voltage error change is uncovered, which enables the NSEBP to adjust each synaptic efficacy accurately. Moreover, the computational error is back propagated to previous layers by presynaptic spike jitter instead of the traditional gradient decent rule, which realizes layer-wise training in our algorithm. In the feedforward calculation, the analytic solutions of the spike time are calculated in the spiking response model, instead of detecting the postsynaptic voltage states at all time points. Employing these strategies, our algorithm achieves a significant improvement in training efficiency compared with the traditional training methods.

Learning Algorithm

In this section, a new algorithm for feed-froward multilayer spiking neural networks, the Normalized Spiking Error Back Propagation (NSEBP) is presented.

Neuron Model

In our study, the simplified Spike Response Model (SRM$_0$) is employed because of its simplicity and effectiveness. In the SRM$_0$ [37], once the jth spike is emitted, a fundamental voltage ϵ_j is inspired and transmitted to its postsynaptic neuron. Each postsynaptic neuron integrates the weighted sum of all presynaptic influence ϵ_j at time t as its voltage $u(t)$, and emits a spike if its voltage $u(t)$ reaches the threshold. The postsynaptic voltage $u(t)$ is described in the following equations:

$$u(t) \quad = \quad \eta(t - \hat{t}_{out}) + \sum_{j \in \Gamma_j} w_j \epsilon_j (t - t^j_{in}) + u^{ext}, \tag{1}$$

where

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

with the Heaviside step function

$$H(s_j) \quad = \quad \begin{cases} 1, & \text{if} \quad s_j \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Specifically, $\hat{t}_{out}$ denotes the last recent output spike of the postsynaptic neuron, w_j is the weight of the presynaptic neuron emitting the jth input spike, and $\eta(t - \hat{t}_{out})$ is the refractory function to simulate the biological refractory period. Γ_j is a set containing the spike time emitted by all the presynaptic neurons, u^{ext} is the external voltage, $s_j = t - t^j_{in}$, with t^j_{in} denoting the jth firing time of the input spike train. τ_1 and τ_2 are constant parameters.

In our algorithm, the Post-Synaptic Potential (PSP) learning window is employed, which is represented in Eq (4), providing a relation of the weight modification and spike time deviation. Obviously, it only directs weight modification if the presynaptic neuron fires before the postsynaptic one.

$$W_{ind}(s_j) \quad = \quad \begin{cases} A_1 \epsilon_j(s_j), & \text{if} \quad s_j \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

where A_1 is a constant set to be 1 in our study, and $s_j = t - t_{in}^j$ denotes the time distance between the current time t and the input time t_{in}^j.

The NSEBP Algorithm

In this section, the Normalized Spiking Error Back Propagation (NSEBP) is proposed. The feedforward calculation process is derived in the Theorem 1, and the feedback training process of this learning rule is described here for one postsynaptic neuron with several presynaptic neurons.

In the network with n layers employing the SRM_0 model, an arbitrary postsynaptic neuron o has an input spike train $T_{in} = \{t_{in}^1, t_{in}^2, t_{in}^3, \dots t_{in}^P\}$ denoting the ordered input spikes, and a target spike train $T_d = \{t_d^1, t_d^2, \dots, t_d^D\}$. Inspired by the selective attention mechanism of the primate visual system, the NSEBP only detects and trains the voltage states at target time points for neuron o, and ignore states on other non-target time.

For each postsynaptic neuron o, instead of the traditional time error, the voltage distance between the threshold ϑ and the postsynaptic voltage $u(t_d)$ at the target time t_d is employed as the network error in our algorithm described in Eq (5), which is trained to become zero in our algorithm:

$$\text{err} = \vartheta - u(t_d) \tag{5}$$

To train the postsynaptic voltage to ϑ, two steps are applied to our algorithm, that are the presynaptic spike jitter to back propagate error and the weight modification to complete training of the current layer.

Presynaptic spike jitter. The presynaptic spike jitter is employed to back propagate error instead of the traditional gradient descent rule. It can influence the postsynaptic neuron voltage $u(t_d)$ and realize layer-wised training. To achieve the back-propagated layer-wised learning, neurons in hidden layers also require the target spike time and training error. Then, the error in Eq (5) is allocated to n layers by the normalized parameter r and back propagated by the presynaptic spike jitter. The error assigned to the current layer err_w^n is

$$err_w^n = r\text{err}, \tag{6}$$

and to the previous $n - 1$ layers err_t^n is

$$err_t^n = (1 - r)\text{err}, \tag{7}$$

where r is set to $1/n$ in our algorithm. This error assignment is shown in Fig 1.

The error err_t^n is back propagated to previous $n - 1$ layers by shifting each influential presynaptic spikes (presynaptic spikes which have influence to the postsynaptic neuron o at the current target time t_d). The error assigned to the jth influential presynaptic spike is Δu_j, which is calculated by

$$\Delta u_j = \gamma_t^{jn} err_t^n. \tag{8}$$

In which γ_t^{jn} is an assign variable and calculated by

$$\gamma_t^{jn} = \begin{cases} (A_1 - W_{ind}(s_j)) / \sum_{k=m_1}^{m_2} (A_1 - W_{ind}(s_k)), & \text{if} \quad \text{err} > 0, \\ W_{ind}(s_j) / \sum_{k=m_1}^{m_2} W_{ind}(s_k), & \text{if} \quad \text{err} < 0, \end{cases} \tag{9}$$

with A_1 and $W_{ind}(s_j)$ defined in Eq (4). m_1 and m_2 are the first and last indexes of the influential presynaptic spikes respectively. To achieve training of this Δu_j, the time variation of the jth

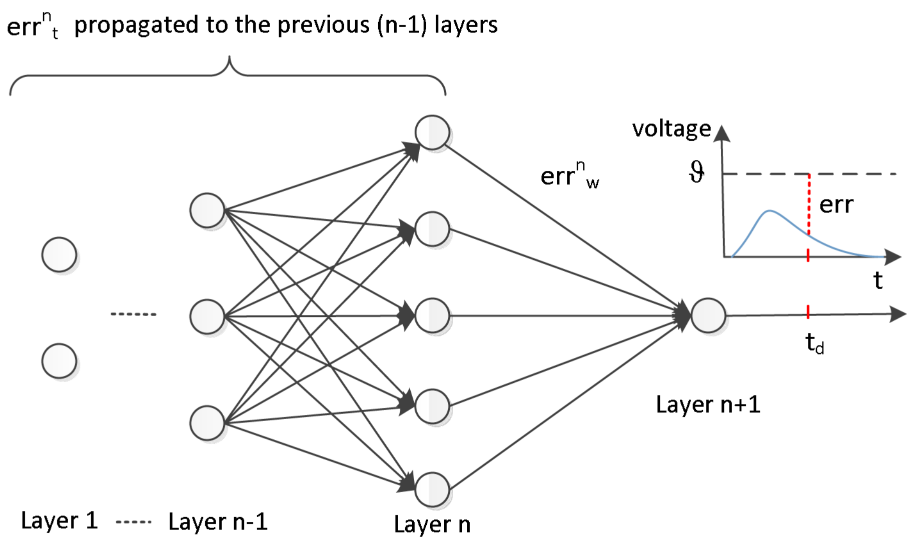

Fig 1. The error and its assignment in our algorithm. The error *err* is assigned to two parts, among which err^n_w is assigned to the current layer for weight modification, and err^n_t is propagated to previous (n – 1) layers.

doi:10.1371/journal.pone.0150329.g001

input spike Δt^j_{pre} is calculated by

$$\Delta t^j_{pre} = \tau_1 \ln\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right), \tag{10}$$

with

$$a = -w_j \exp\left((t^j_{pre} - t_d)/\tau_2\right), \tag{11}$$

$$b = w_j \exp\left((t^j_{pre} - t_d)/\tau_1\right), \tag{12}$$

$$c = w_j \exp\left((t^j_{pre} - t_d)/\tau_2\right) - w_j \exp\left((t^j_{pre} - t_d)/\tau_1\right) - \Delta u_j, \tag{13}$$

where w_j is the corresponding weight of the *j*th presynaptic spike, t_d is the target spike time, τ_1 and τ_2 are model parameters defined in Eq (2), t^j_{pre} denotes the *j*th presynaptic spike time. If $a \neq 0$ and $b^2 - 4ac \geq 0$ hold, Δt^j_{pre} is calculated by Eq (10) and the solution with the minimum absolute value is applied to our algorithm. The calculation is derived in the following theoretical derivation section.

Weight modification. To train err^n_w to 0, the weight modification in our algorithm for an arbitrary influential input spike *j* is calculated by

$$\Delta w_j = \frac{\gamma^n_j err^n_w}{\epsilon_j(s_j)}, \tag{14}$$

where γ^n_j is a parameter defined by the normalized learning window

$$\gamma^n_j = \frac{W_{ind}(s_j)}{\sum_{k=m_1}^{m_2} W_{ind}(s_k)}, \tag{15}$$

with $\epsilon_j(s_j)$ calculated by Eq (2), and $W_{ind}(s_j)$ by Eq (4).

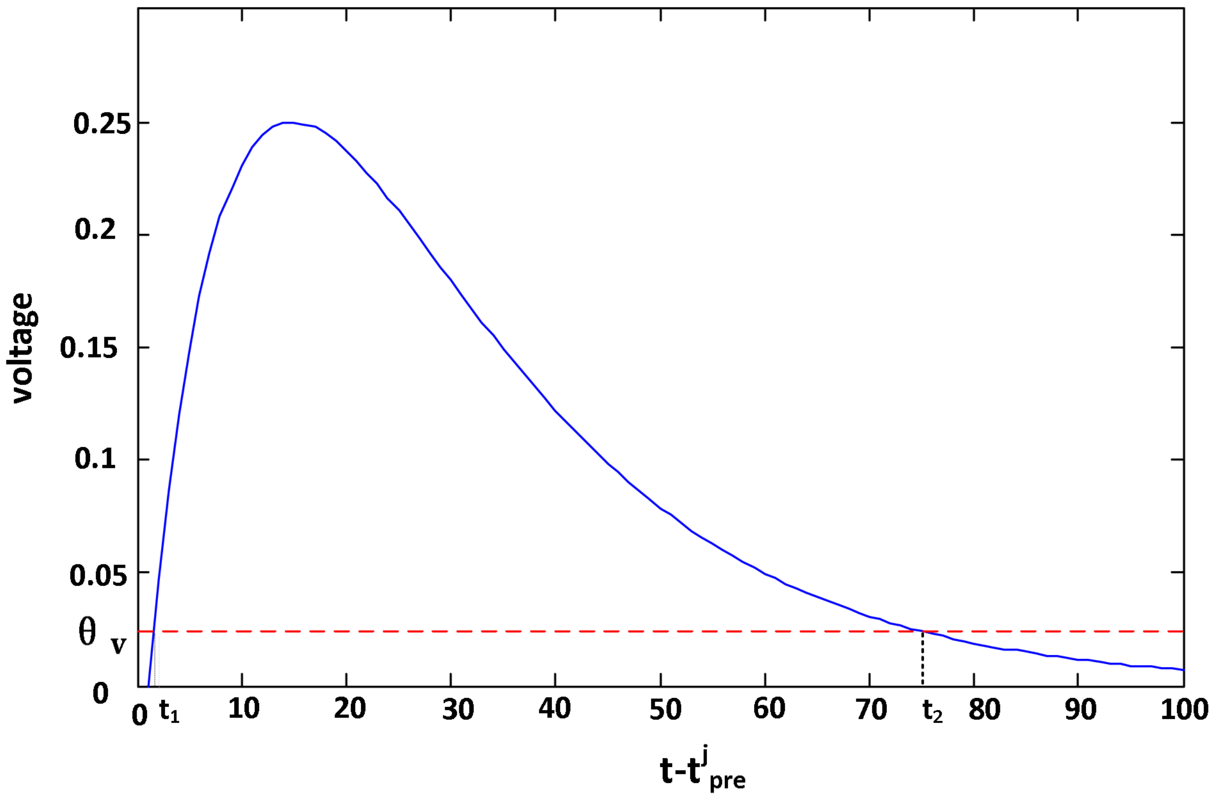

Fig 2. The voltage in the time scope $t - t^j_{pre} \in [t_1, t_2]$. The voltage $\epsilon_j(s_j)$ caused by the input spike t^j_{pre} is above ϑ_v when $t - t^j_{pre} \in [t_1, t_2]$. The voltage of the input t^j_{pre} is set to 0 at time t if this $t - t^j_{pre}$ is not in the interval $[t_1, t_2]$.

doi:10.1371/journal.pone.0150329.g002

To avoid the $\epsilon_j(s_j)$ in Eq (14) going to infinitesimal when s_j is too large, not all presynaptic spikes but only the spikes with voltage $\epsilon_j(s_j) > \vartheta_v$ at t_d are trained, where ϑ_v is the voltage threshold. Solving the same mathematical equations as Theorem 2 in the following section, the time boundaries are $t_1 = t_d - t^j_{pre} + \tau_1 \ln((1 - \sqrt{1 - 4\vartheta_v})/2)$ and $t_2 = t_d - t^j_{pre} + \tau_1 \ln((1 + \sqrt{1 - 4\vartheta_v})/2)$, as shown in Fig 2. Its corresponding spike index range is denoted by $[m_1, m_2]$.

If there is no input spike in the range $[t_1, t_2]$, S spikes in $[t_1, t_2]$ are added randomly to the presynaptic hidden neurons with a probability. The probability p_i assigned spikes to neuron i is calculated by

$$p_i = \frac{\frac{1}{n_i}}{\sum_{k=1}^{m} \frac{1}{n_k}}, \tag{16}$$

in which, m is the number of presynaptic neurons, n_i is the number of spikes emitted by neuron i, and $n_i = 0.5$ if there is no spike emitted. Consequently, the fewer spikes emitted by neuron i, the higher probability p_i it possesses. This allocation approach not only solves the none input problem in the training, but also balances the spike distribution. These added spikes are regarded as the target time of previous hidden layers and trained in the previous layer.

Mathematical Analysis

Theoretical Derivation

In this section, the theoretical derivations in the feedforward and back propagation of our algorithm are presented in Theorem 1 and Theorem 2 respectively.

In traditional methods employing the temporal encoded SNNs, the output spikes of a neuron are detected serially in time, leading to an inefficient feed forward process. In our study, the output spike times are obtained by solving the voltage function instead of traversing all time points, which are derived in the following theorem.

Supposing that for a postsynaptic neuron o, the refractory period $\eta(t - \hat{t}_{out})$ is set to $-A_2 \exp(-(t - \hat{t}_{out})/\tau_1)$ with the constant $A_2 > 0$, the external interference voltage $u^{ext} = 0$, and $T_{pre} = \{t^1_{pre}, t^2_{pre}, t^3_{pre}, \ldots t^{P_1}_{pre}\}$ is the ordered presynaptic spike train of the P_1 presynaptic input spikes, where t^j_{pre} denotes the jth presynaptic spikes with w_j representing its response synapse weight. m_o is the index of the first influencing spike to the postsynaptic neuron o, ϑ is the firing threshold, τ_1 and τ_2 are model parameters defined in Eq (2). With these definitions, the relation between the pre and postsynaptic spikes is obtained by solving the quadratic function, which is proved in the following theorem:

Theorem 1 *In the SRM*$_0$ *model, for each range* $[t^j_{pre}, t^{j+1}_{pre})$ *in* T_{pre} *with* $1 \le j \le P_1 - 1$, *assuming that*

$$a = \sum_{m=m_0}^{j} w_m \exp(t^m_{pre}/\tau_2), \tag{17}$$

$$b = \sum_{m=m_0}^{j} w_m \exp(t^m_{pre}/\tau_1) - A_2 \exp(\hat{t}_{out}/\tau_1) \tag{18}$$

If the following conditions hold:

$$\begin{aligned} (I) &\qquad a \neq 0 \text{ and } b^2 - 4a\vartheta \ge 0, \\ (II) &\qquad \tau_1 = 2\tau_2, \end{aligned}$$

the postsynaptic output spike t_{out} *in the range* $[t^j_{pre}, t^{j+1}_{pre})$ *is solved by*

$$t_{out} = -\tau_1 \ln\left(\frac{b \pm \sqrt{b^2 - 4a\vartheta}}{2a}\right). \tag{19}$$

Proof: According to the SRM$_0$ model described in Eqs (1)–(3), for $t \in [t^j_{pre}, t^{j+1}_{pre})$, the voltage of a postsynaptic neuron $u(t)$ is

$$u(t) = \eta(t - \hat{t}_{out}) + \sum_{m=m_0}^{j} w_m \epsilon_m(t - t^m_{pre}) + u^{ext}. \tag{20}$$

Since

$$\begin{cases} \eta(t - \hat{t}_{out}) = -A_2 \exp(-(t - \hat{t}_{out})/\tau_1), \\ u^{ext} = 0, \end{cases} \tag{21}$$

at time t_{out}, we have

$$u(t_{out}) = \sum_{m=m_0}^{j} w_m \epsilon_m (t_{out} - t_{pre}^m) - A_2 \exp\left(-\frac{t_{out} - \hat{t}_{out}}{\tau_1}\right). \tag{22}$$

The postsynaptic neuron will fire once its voltage reaches the threshold ϑ, then the postsynaptic output spike time t_{out} follows

$$\sum_{m=m_0}^{j} w_m \epsilon_m (t_{out} - t_{pre}^m) - A_2 \exp\left(-\frac{t_{out} - \hat{t}_{out}}{\tau_1}\right) = \vartheta. \tag{23}$$

According to Eq (2),

$$\sum_{m=m_0}^{j} w_m \left[\exp\left(-\frac{t_{out} - t_{pre}^m}{\tau_1}\right) - \exp\left(-\frac{t_{out} - t_{pre}^m}{\tau_2}\right) \right] - A_2 \exp\left(-\frac{t_{out} - \hat{t}_{out}}{\tau_1}\right) = \vartheta. \tag{24}$$

Thus, we have

$$\sum_{m=m_0}^{j} w_m \left\{ \exp\left(\frac{-t_{out}}{\tau_1}\right) \exp\left(\frac{t_{pre}^m}{\tau_1}\right) - \exp\left(\frac{-t_{out}}{\tau_2}\right) \exp\left(\frac{t_{pre}^m}{\tau_2}\right) \right\} - A_2 \exp\left(\frac{-t_{out}}{\tau_1}\right) \exp\left(\frac{\hat{t}_{out}}{\tau_1}\right) \\ = \vartheta. \tag{25}$$

Suppose

$$z = \exp\left(-\frac{t_{out}}{\tau_1}\right),$$

and refer to Eqs (17), (18), (25) and (*II*),

$$az^2 - bz + \vartheta = 0. \tag{26}$$

By (*I*), the solutions of Eq (26) is

$$z = \frac{b \pm \sqrt{b^2 - 4a\vartheta}}{2a}, \tag{27}$$

and for all presynaptic time, we have $t_i > 0$, $z > 0$, then

$$t_{out} = -\tau_1 \ln\left(\frac{b \pm \sqrt{b^2 - 4a\vartheta}}{2a}\right). \tag{28}$$

The result follows.

The Theorem 1 proves the relation between the pre and postsynaptic spikes, which is applied to our algorithm to improve the feedforward computation efficiency.

In the feedback process of our algorithm, the error is back propagated by the presynaptic spike jitter instead of the traditional gradient decent rule, by which, the layer-wised training is applicable to our algorithm and improves the learning efficiency of our algorithm significantly. The relation of the presynaptic spike jitter and the voltage change is investigated in the following theorem.

Supposing that Δu_j is the voltage variation of the postsynaptic neuron o generated by the jth presynaptic spike, t_d is the current target spike time, and other variables are the same as that in Theorem 1, then the relation between the time jitter Δt_{pre}^j and Δu_j is obtained by solving the quadratic function, which is proved in the following theorem:

Theorem 2 *In the SRM$_0$ model with $\tau_1 = 2\tau_2$, if the voltage change Δu_j follows*

$$\begin{array}{ll}(I) & w_j \neq 0, \\ (II) & -w_j\epsilon_j(t_d - t^j_{pre}) < \Delta u_j \leq \left(\frac{1}{4} - \epsilon_j(t_d - t^j_{pre})\right) w_j, \ \text{if} \ w_j > 0 \\ (III) & \left(\frac{1}{4} - \epsilon_j(t_d - t^j_{pre})\right) w_j \leq \Delta u_j < -w_j\epsilon_j(t_d - t^j_{pre}), \ \text{if} \ w_j < 0\end{array}$$

and assuming that

$$a = -w_j \exp\left((t^j_{pre} - t_d)/\tau_2\right), \tag{29}$$

$$b = w_j \exp\left((t^j_{pre} - t_d)/\tau_1\right), \tag{30}$$

$$c = w_j \exp\left((t^j_{pre} - t_d)/\tau_2\right) - w_j \exp\left((t^j_{pre} - t_d)/\tau_1\right) - \Delta u_j, \tag{31}$$

the voltage variation Δu_j of the postsynaptic neuron can be achieved by the presynaptic spike time jitter

$$\Delta t^j_{pre} = \tau_1 \ln\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right). \tag{32}$$

Proof: Supposing that u_j is the postsynaptic voltage stimulated by the input spike t^j_{pre} at the target time t_d, and the presynaptic spike jitter $\triangle t^j_{pre}$ makes the voltage change $\triangle u_j$. By Eq (1), we have

$$w_j\left(\exp\left(\frac{t^j_{pre} + \Delta t^j_{pre} - t_d}{\tau_1}\right) - \exp\left(\frac{t^j_{pre} + \Delta t^j_{pre} - t_d}{\tau_2}\right)\right) = u_j + \Delta u_j, \tag{33}$$

and then

$$\begin{aligned}\Delta u_j = \ & w_j[\exp\left(\frac{t^j_{pre} - t_d}{\tau_1}\right)\exp\left(\frac{\Delta t^j_{pre}}{\tau_1}\right) - \exp\left(\frac{t^j_{pre} - t_d}{\tau_1}\right) \\ & + \exp\left(\frac{t^j_{pre} - t_d}{\tau_2}\right) - \exp\left(\frac{t^j_{pre} - t_d}{\tau_2}\right)\exp\left(\frac{\Delta t^j_{pre}}{\tau_2}\right)].\end{aligned} \tag{34}$$

Let

$$z = \exp\left(\Delta t^j_{pre}/\tau_1\right), \tag{35}$$

and by Eqs (29)–(31) and (34) can be expressed by

$$az^2 + bz + c = 0. \tag{36}$$

Under the condition (I), we have $w_j \neq 0 \Rightarrow a \neq 0$, and if $w_j > 0$, for condition (II),

$$\Delta u_j \leq \left(\frac{1}{4} - \epsilon_j(t_d - t^j_{pre})\right) w_j, \tag{37}$$

$$\epsilon_j(t_d - t^j_{pre}) + \frac{\Delta u_j}{w_j} \leq \frac{1}{4}. \tag{38}$$

Then the discriminant of Eq (36) is

$$\begin{aligned}\Delta = b^2 - 4ac &= w_j^2 \exp^2\left(\frac{t_{pre}^j - t_d}{\tau_1}\right) - 4w_j^2 \exp\left(\frac{t_{pre}^j - t_d}{\tau_2}\right)\left[\epsilon_j(t_d - t_{pre}^j) + \frac{\Delta u_j}{w_j}\right] \\ &\geq w_j^2 \exp^2\left(\frac{t_{pre}^j - t_d}{\tau_1}\right) - w_j^2 \exp\left(\frac{t_{pre}^j - t_d}{\tau_2}\right) \\ &= w_j^2 \exp\left(\frac{t^j - t_d}{\tau_2}\right) - w_j^2 \exp\left(\frac{t_{pre}^j - t_d}{\tau_2}\right) = 0.\end{aligned} \tag{39}$$

Analogously, when $w_j < 0$, for condition (*III*),

$$\Delta u_j \geq \left(\frac{1}{4} - \epsilon_j(t_d - t_{pre}^j)\right) w_j, \tag{40}$$

$$\epsilon_j(t_d - t_{pre}^j) + \frac{\Delta u_j}{w_j} \leq \frac{1}{4}, \tag{41}$$

$$\Delta = b^2 - 4ac \geq 0. \tag{42}$$

Then, under these conditions, Eq (36) has solutions

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{43}$$

By the property of the logarithmic function, the spike jitter Δt_{pre}^j can be obtained by Eq (35) only if $z > 0$. For $w_j > 0$, we have $a < 0$, $b > 0$, then

$$z = \frac{-b - \sqrt{b^2 - 4ac}}{2a} > 0. \tag{44}$$

Under condition (*II*), for $w_j > 0$, $\Delta u_j > -w_j \epsilon_j(t_d - t_{pre}^j)$, we have $\epsilon_j(t_d - t_{pre}^j) + \Delta u_j / w_j > 0$, and then

$$4ac = 4w_j^2 \exp\left(\frac{t_{pre}^j - t_d}{\tau_2}\right)\left[\epsilon_j(t_d - t_{pre}^j) + \frac{\Delta u_j}{w_j}\right] > 0, \tag{45}$$

$$b^2 - 4ac < b^2, \tag{46}$$

$$z = \frac{-b + \sqrt{b^2 - 4ac}}{2a} > 0. \tag{47}$$

Analogously, by condition (*III*), $w_j < 0$,

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} > 0. \tag{48}$$

Then, under these conditions, Δt_{pre}^j is solved by Eqs (35) and (43) with

$$\Delta t_{pre}^j = \tau_1 \ln\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right). \tag{49}$$

The result follows.

Specially, when Δu_j exceeds the boundary of Theorem 2 (*II*) or (*III*) in our algorithm, it is set to the corresponding feasible boundary in the same direction of the condition.

Convergence Analysis

In this section, the convergence of our algorithm is investigated. To guarantee the convergence of the traditional and our algorithms employing the SRM$_0$ model, some conditions need to be met to select target time points. These conditions for traditional algorithms and our algorithm are studied in Theorem 3 (1) and Theorem 3 (2) respectively by analyzing the voltage function and the spiking firing conditions.

Theorem 3 *In the network under n layers employing the SRM$_0$ model described in Eqs (1)–(3) with $\tau_1 = 2\tau_2$, we have:*

(1) To guarantee the convergence of the traditional algorithms based on the precise spike time mechanism, a time point t_d^m is available as target time only if there exist input spikes in $[t_d^m - (n-1)\tau_1 \ln 2, t_d^m)$.

(2) To guarantee the convergence of our algorithm, a time points t_d^m is available as target time only if there exist input spikes in $[0, t_d^m)$. When the strategy of $[t_1, t_2]$ described in Fig 2 is applied to our algorithm, this scope is $[t_d^m - nt_1, t_d^m - nt_2]$.

Proof: (1) For an arbitrary *j*th input spike t_{in}^j, by Eq (2),

$$\begin{aligned} \epsilon_{pre}(t - t_{in}^j) &= \exp\left(-\frac{t - t_{in}^j}{\tau_1}\right) - \exp\left(-\frac{t - t_{in}^j}{\tau_2}\right) \\ &= \exp\left(\frac{t_{in}^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) - \exp\left(\frac{t_{in}^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right). \end{aligned} \tag{50}$$

Taking the partial derivatives,

$$\frac{\partial \epsilon_{pre}(t - t_{in}^j)}{\partial t} = -\frac{1}{\tau_1}\exp\left(\frac{t_{in}^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) + \frac{1}{\tau_2}\exp\left(\frac{t_{in}^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right). \tag{51}$$

In the traditional precise time mechanism, it is $\frac{\partial \epsilon_{pre}(t - t_{in}^j)}{\partial t} \geq 0$ when emitting spikes, then we have

$$-\frac{1}{\tau_1}\exp\left(\frac{t_{in}^j}{\tau_1}\right)\exp\left(-\frac{t}{\tau_1}\right) + \frac{1}{\tau_2}\exp\left(\frac{t_{in}^j}{\tau_2}\right)\exp\left(-\frac{t}{\tau_2}\right) \geq 0, \tag{52}$$

$$\exp\left(\frac{t}{\tau_1} - \frac{t}{\tau_2}\right) \geq \frac{\tau_2}{\tau_1}\left(\frac{t_{in}^j}{\tau_1} - \frac{t_{in}^j}{\tau_2}\right), \tag{53}$$

by $\tau_1 = 2\tau_2$,

$$\left(\frac{\tau_2 - \tau_1}{\tau_1\tau_2}\right)t \geq -\ln 2 + \left(\frac{\tau_2 - \tau_1}{\tau_1\tau_2}\right)t_{in}^j, \tag{54}$$

$$t \leq t_{in}^j + \tau_1 \ln 2. \tag{55}$$

Then in traditional algorithms, the input spike t_{in}^j can only inspire output spikes in the scope $(t_{in}^j, t_{in}^j + \tau_1 \ln 2]$ for its postsynaptic neurons. Analogously, the output spikes caused by t_{in}^j are in $(t_{in}^j, t_{in}^j + (n-1)\tau_1 \ln 2]$ after *n* layers. Consequently, traditional algorithms cannot get convergent at t_d^m if there is no input spike in $[t_d^m - (n-1)\tau_1 \ln 2, t_d^m)$. Then, to guarantee the convergence of the traditional algorithms based on the precise spike time mechanism, a time point t_d^m is available as target time only if there exist input spikes in $[t_d^m - (n-1)\tau_1 \ln 2, t_d^m)$.

(2) Our algorithm employs the primate selective attention mechanism instead of the precise spike time rule, then there is no requirement of $\partial\epsilon_{pre}(t - t_{in}^j)/\partial t \geq 0$. When the local influence shown in Fig 2 is not applied to our algorithm, all presynaptic spikes which have an influence on t_d^m can be trained to complete learning. Consequently, the time scope for input spikes is $[0, t_d^m)$.

If the local influence shown in Fig 2 is applied to our algorithm, the time scope of output spikes generated by the input t_{in}^j after several layers is shown in Fig 3. It is obvious that in layer 1, the time scope is $[t_1, t_2]$, and in layer 2, the earliest time to fire is $t_1' = t_1 + t_1$, and the latest firing time is $t_2' = t_2 + t_2$. Then, for layer $n - 1$, we have t_d satisfying Eq (56) to complete training:

$$(n-1)t_1 < t_d^m < (n-1)t_2 \tag{56}$$

Obviously, if there is no input in $[t_d^m - nt_1, t_d^m - nt_2]$, this t_d^m can not be trained convergently. Then, in this condition, the time points t_d^m is available as target time only if there exist input spikes in $[t_d^m - nt_1, t_d^m - nt_2]$.

The results follow.

Theorem 3 provides conditions for encoding target times, which guarantees the convergence of different algorithms. It also indicates that the convergent condition of our algorithm is relaxed compared with traditional algorithms. When target spikes are selected following these conditions, different algorithms have different convergence properties and speed, which depend on their training mechanisms. In our algorithm, all qualified target times can be trained successfully and efficiently.

At the *m*th target time t_d^m, the convergence of our algorithm is proved in the Theorem 4 by analyzing the postsynaptic voltage. Since the interference from training other target spikes or patterns varies with different network status, the following theorem proves the convergence of our algorithm ignoring this interference. The convergent situations with various interference are investigated in the following simulations sections.

Theorem 4 *In the SRM*$_0$ *model described in Eqs* (1)–(3) *with* $\tau_1 = 2\tau_2$, *the postsynaptic voltage of our algorithm at an arbitrary target time* t_d^m *is convergent to the threshold* ϑ *if the condition in Theorem 3 (2) holds, ignoring the interference from training other target spikes or patterns.*

Proof: There are two cases in the training: (1) the presynaptic hidden neurons emit qualified spikes.

Since our algorithm are trained layer-wisely, each layer shares the same training process and becomes convergent in the same way. Then, the convergence of our algorithm is proved in one layer with a postsynaptic neuron and several presynaptic neurons. Suppose that the voltage of the postsynaptic neuron at t_d^m is $u(t_d^m)$ calculated by Eq (1), and the voltage variations generated by the presynaptic weight modification and spike jitter are Δu_w and Δu_t respectively. For an arbitrary postsynaptic neuron *o*, and to the *p*th presynaptic spike, the voltage variation generated by the weight modification of this spike is Δu_w^p, which is calculated by

$$\Delta u_w^p = \frac{\partial u(t_d^m)}{\partial w_p} \Delta w_p, \tag{57}$$

where Δw_p is the variation of its weight w_p. Since $\partial u(t_d^m)/\partial w_p = \epsilon_p(t_d^m - t_{in}^p)$,

$$\Delta u_w^p = \epsilon_p(t_d^m - t_{in}^p)\Delta w_p. \tag{58}$$

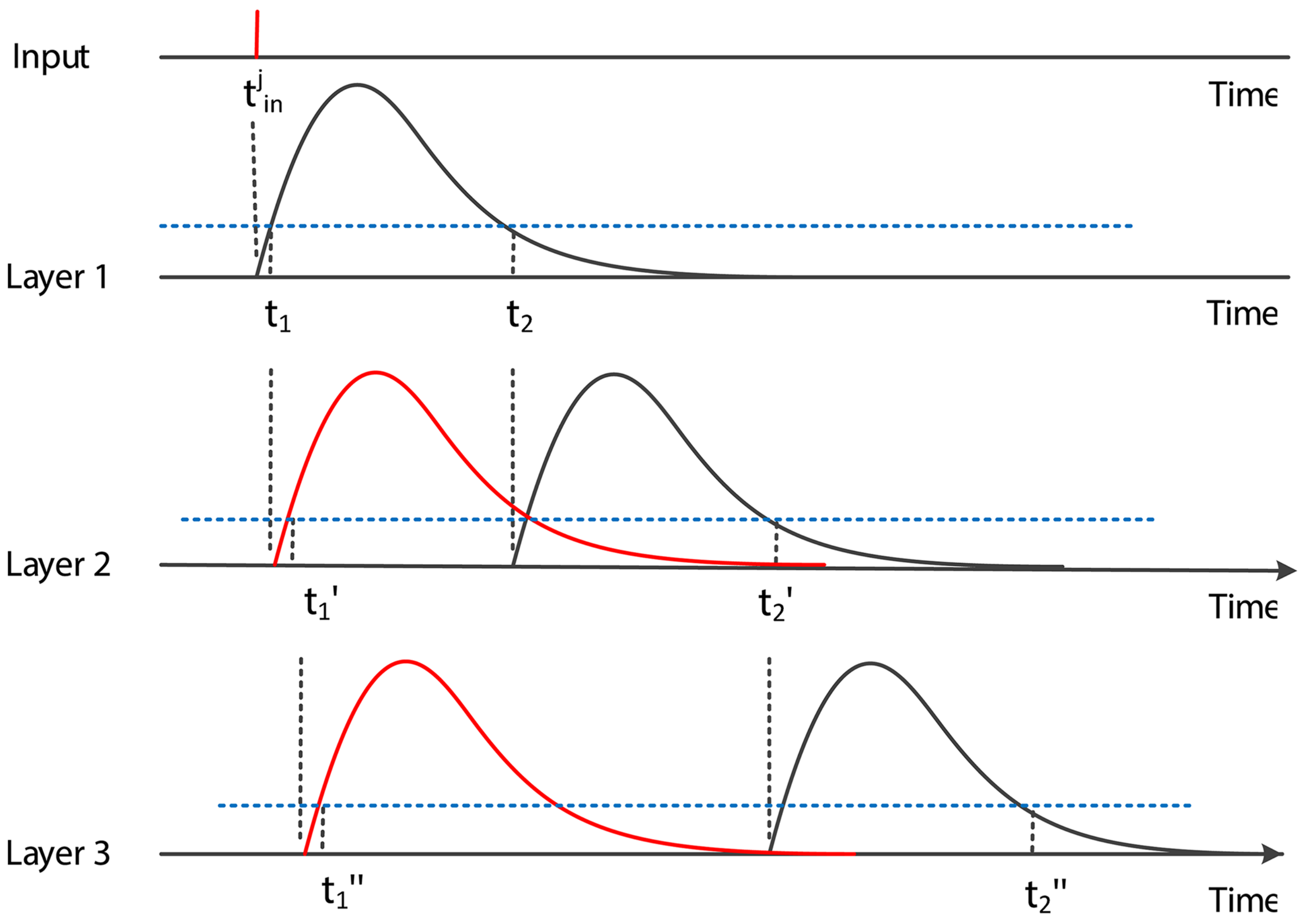

Fig 3. The output spike scopes. The output spike scopes generated by the input spike t^j_{in} to each layer.

doi:10.1371/journal.pone.0150329.g003

For all presynaptic inputs, the voltage variation generated by the weight modification is

$$\Delta u_w = \sum_{p=m_1}^{m_2} \epsilon_p (t^m_d - t^p_{pre}) \Delta w_p \tag{59}$$

in which t^p_{pre} is the pth presynaptic spike time, and m_1 and m_2 are the first and last indexes of these presynaptic spikes. By Eqs (5), (6) and (14),

$$\Delta u_w = \sum_{p=m_1}^{m_2} \epsilon_p (t^m_d - t^p_{pre}) \frac{r\gamma^n_p (\vartheta - u(t^m_d))}{\epsilon_p (t^m_d - t^p_{pre})} = \sum_{p=m_1}^{m_2} r\gamma^n_p (\vartheta - u(t^m_d)). \tag{60}$$

By Eq (15), $\sum_{p=m_1}^{m_2} \gamma_p^n = 1$, then

$$\Delta u_w = r(\vartheta - u(t_d^m)). \tag{61}$$

If the conditions in Theorem 2 hold, according to Eq (8), the Δt_{pre}^p calculated by Eq (32) is

$$\Delta t_{pre}^p = \tau_1 \ln\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right), \tag{62}$$

which makes the postsynaptic voltage variation generated by the presynaptic spike jitter Δu_t^p become $\gamma_t^{pn} err_t^n$, with err_t^n and γ_t^{pn} defined in Eqs (7) and (9) respectively. Under these conditions, for all presynaptic spikes, the voltage variation inspired by the presynaptic spike jitter Δu_t is

$$\Delta u_t = \sum_{p=m_1}^{m_2} \gamma_t^{pn} err_t^n. \tag{63}$$

By Eq (9), $\sum_{p=m_1}^{m_2} \gamma_t^{pn} = 1$, then according to Eqs (5) and (7),

$$\Delta u_t = (1 - r)(\vartheta - u(t_d^m)), \tag{64}$$

where r is a parameter defined in Eq (7). Consequently, the whole postsynaptic voltage variation Δu generated by both presynaptic spike jitter and weight modification is

$$\begin{aligned} \Delta u &= \Delta u_t + \Delta u_m = (1 - r)(\vartheta - u(t_d^m)) + r(\vartheta - u(t_d^m)) \\ &= \vartheta - u(t_d^m), \end{aligned} \tag{65}$$

and then

$$u(t_d^m) + \Delta u = \vartheta. \tag{66}$$

If the value of Δu_t^p exceeds the boundary in Theorem 2, the corresponding feasible boundary value is set to Δu_t^p, and the solution is obtained according to Eq (32). It is obvious that the boundary value has the same training direction with Δu, which can make err close to 0. Then our algorithm will be convergent after several leaning epochs at t_d^m.

(2) If there is no qualified input spike in the presynaptic hidden layer, our algorithm adds spikes randomly with probability calculated by Eq (16), after which all weight modifications and spike jitters are the same as case (1), and our algorithm can get convergence.

The layer-wise training is employed in our algorithm, by which each layer shares the same training process and becomes convergent in the same way. In this analysis, the interference of other target spike trains or patterns are ignored. With the influence, our algorithm requires several more epochs to offset this interference and complete training.

The results follow.

Computational Complexity

In this section, the computational time complexity of our algorithm is studied and compared with two traditional algorithms, the SpikeProp [28] and Multi-ReSuMe [32]. Before this, the detailed pseudo-codes of the feedforward and feedback processes of our method are listed.

Feedforward calculation. According to the previous description, the pseudo-code of the feedforward calculation of our method is listed below. Since in the multilayer networks, each layer shares the same feedforward calculation process, only the calculation of one layer is described in this pseudo-code.

The Feedforward Calculation of Our Algorithm

```
Definition:
$T_{pre}$: the set of presynaptic spikes, which contains spikes emitted by all pre-
synaptic neurons $\{t_{pre}^1, t_{pre}^2, t_{pre}^3, \ldots, t_{pre}^P\}$. $T_{pre}$ is sorted and has no duplicate
numbers.
Initialization:
The weight matrix $W$ is initialized randomly.
Feedforward calculation:
  For each postsynaptic neuron:
    For each presynaptic spike interval $t_{pre}^j$ to $t_{pre}^{j+1}$ with $1 < j < P-1$:
      For all presynaptic spikes before $t_{pre}^j$, calculate parameters $a$ and $b$ by
      Eqs (17) and (18).
      If $a$ and $b$ meet the conditions in Theorem 1:
        Calculate the output spike time in this scope by Eq (19) and add it to
        the output spike train of the postsynaptic neuron.
      End If
    End For
  End For
```

Supposing that there are M presynaptic neurons, N postsynaptic neurons, P input spikes of all these presynaptic neurons, and the time length is T. As described in the pseudo-code, our algorithm detects each input spike scope and calculates parameters a and b by all of these P input spikes in the worst case, then the time complexity of our algorithm for one layer is $O(NP^2)$, which also reveals the number of operations in our method.

For traditional feedforward calculation method, all discrete time points in T are detected instead of the input spike scopes of P, then the second loop in the pseudo-code above is replaced by the time scope in T (supposing that the time interval is 1ms). For each time scope, it calculates the postsynaptic voltage by these P input spikes and determines whether the voltage is greater than the threshold. In this way, the time complexity of the traditional method in one layer is $O(NTP)$. Since a neuron can only emits one spike in a time points, we have $P \leq T$. Consequently, the time complexity of our method in the feedforward calculation is less than that of the traditional approach.

Feedback modification. Similar to the feedforward calculation, the feedback weight modifications in each layer and each output neuron of our algorithm have the same training process. Then in this part, only the training process of one layer and one output neuron is listed in the following pseudo-code.

The Feedback Modification of Our Algorithm

```
Definition:
$T_{pre}$: the set of presynaptic spikes, which contains spikes emitted by all pre-
synaptic neurons $\{t_{pre}^1, t_{pre}^2, t_{pre}^3, \ldots, t_{pre}^P\}$.
$T_d$: the set of target output spikes, which contains all target spikes of the
postsynaptic neuron $\{t_d^1, t_d^2, t_d^3, \ldots, t_d^{D_1}\}$.
Initialization:
The weight matrix $W$ is initialized randomly.
Feedback modification:
  For each target spike time in $T_d$:
    Calculate the weighted sum of all input spikes as the postsynaptic volt-
    age $u(t_d)$, and calculate the error $err$ by Eq (5).
    If $err \neq 0$
      Step1: Assign $err$ to each layer by Eqs (6) and (7).
```

```
        Step2: Calculate the presynaptic spike variation in the current layer
        by Eq (10).
        Step3: Adjust all presynaptic weights in this layer by Eq (14).
    End If
End For
```

Assuming that there are M presynaptic neurons, N postsynaptic neurons, and the number of target spikes for all postsynaptic neurons is D, which is equal to $D_1 + D_2 + \ldots + D_N$, with D_i represents the number of target spikes of the ith postsynaptic neuron. The number of input spikes is P, and the time length is T. According to the pseudo-code for the feedback modification of our algorithm, the time complexity for one layer is $O(DP)$, which also reveals the number of operations in our algorithm.

In traditional algorithms, like the SpikeProp and Multi-ReSuMe, the postsynaptic states at all time points of T instead of target intervals T_d are detected and their corresponding weights are modified by a given condition. Then the time complexity for most traditional algorithms like the SpikeProp and Mullti-ReSuMe in one layer is $O(TP)$. Since $D < T$, the number of operations in our algorithm is less than traditional methods.

Training Performance

In this section, the training performance of our algorithm is investigated and compared with two classical algorithms, the SpikeProp [28] and Multi-ReSuMe [32].

A spiking network structure employing an input layer with 50 neurons, a hidden layer with 100 spiking neurons, and an output neuron is devised in our simulations, which is shown in Fig 4. The training of the multilayer neural network consists of two steps, the feedforward

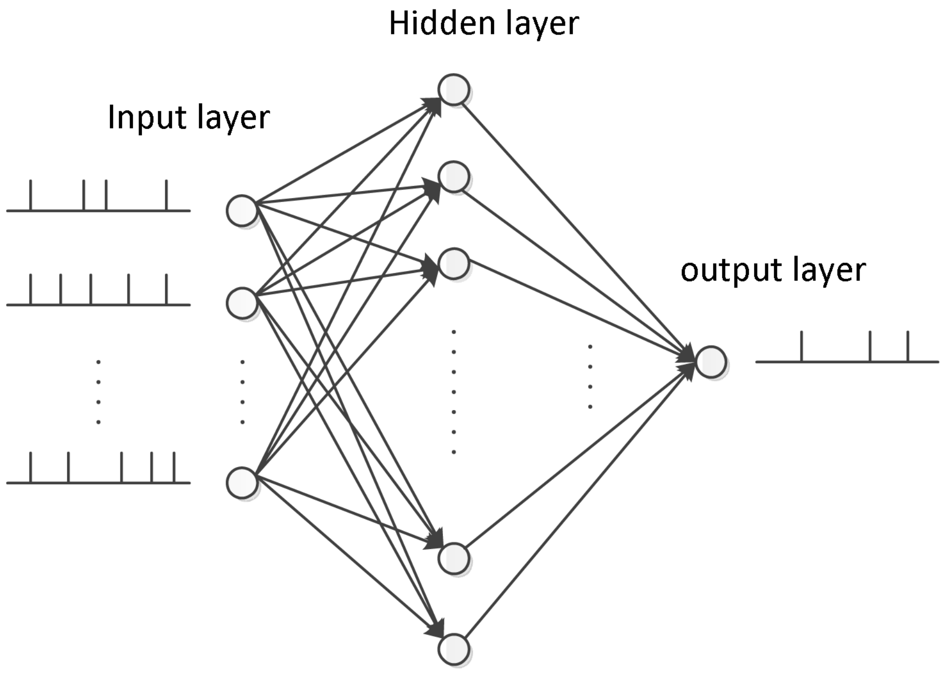

Fig 4. The network structure in our simulation. There are 50 input neurons, 100 hidden neurons, and one output neuron.

doi:10.1371/journal.pone.0150329.g004

calculation and feedback weight modification. The efficiency of the both two steps is studied in the following parts.

Feedforward Calculation

The feedforward calculation is an important step before learning, it computes the output spikes from the input ones. In this subsection, two simulations are conducted to investigate the computational performance of our proposed method described in Theorem 1 compared with the traditional precise time calculation method. Specifically, the network structure is shown in Fig 4, and these input output spikes are generated by a homogeneous Poisson process.

The first simulation is carried out to explore the feedforward calculation performance in different time lengths from 200 ms to 2800 ms, and for each input neuron, one spike generated by a homogeneous Poisson process is emitted. To evaluate the similarity of the output spike trains calculated by our algorithm and traditional method quantitatively, the correlation-based measure C [38] is employed, with

$$C = \frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{|\boldsymbol{v_1}||\boldsymbol{v_2}|},$$

where $\boldsymbol{v_1} \cdot \boldsymbol{v_2}$ is the inner product, and $|\boldsymbol{v_1}|$, $|\boldsymbol{v_2}|$ are the Euclidean norms of $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ respectively. The $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are vectors obtained by the convolution of the two spike trains using a Gaussian filter:

$$v_i(t) = \sum_{m=1}^{N_i} \exp\left[-(t - t_m^i)^2/\sigma^2\right],$$

where N_i is the number of spikes in the test spike train, and t_m^i is the mth spike in it. σ is the standard deviation of this Gaussian filter which is set to be 1 in our study. Generally, the measure C equals to 1 for identical spike trains and decrease towards zero for loosely correlated spike trains.

The simulation results are shown in Fig 5A, which indicate that our proposed method has the same output spike trains as the traditional method, but achieves higher efficiency than it, because our method detects only time intervals of the input spikes instead of all time points in the traditional method.

In the second simulation, the performance of our proposed method is tested with the input firing rate of a homogeneous Poisson process ranging from 100 Hz to 1100 Hz with time length fixed to 1000 ms. Obviously, the higher the input firing rate, the higher the input densities, and when the firing rate is higher than 1000 Hz, the density of input spikes and all time points is similar.

Simulation results shown in Fig 5B indicate that our method has the same output spike trains as the traditional ones, and its computational time is growing with the increase of the input spike density. However, our method is still a little more efficient than the traditional method even if the input spike density is similar to that of all time intervals with a rate above 1000 Hz.

Feedback Weight Modification

In this section, the training performance of our algorithm is investigated and compared with the traditional classical multi-layer algorithms, the Spikeprop and Multi-ReSuMe. The first simulation is devised to study the learning efficiency at different time lengths of spike trains, in which there are 50 input neurons, 100 hidden neurons, and one output neuron with a network

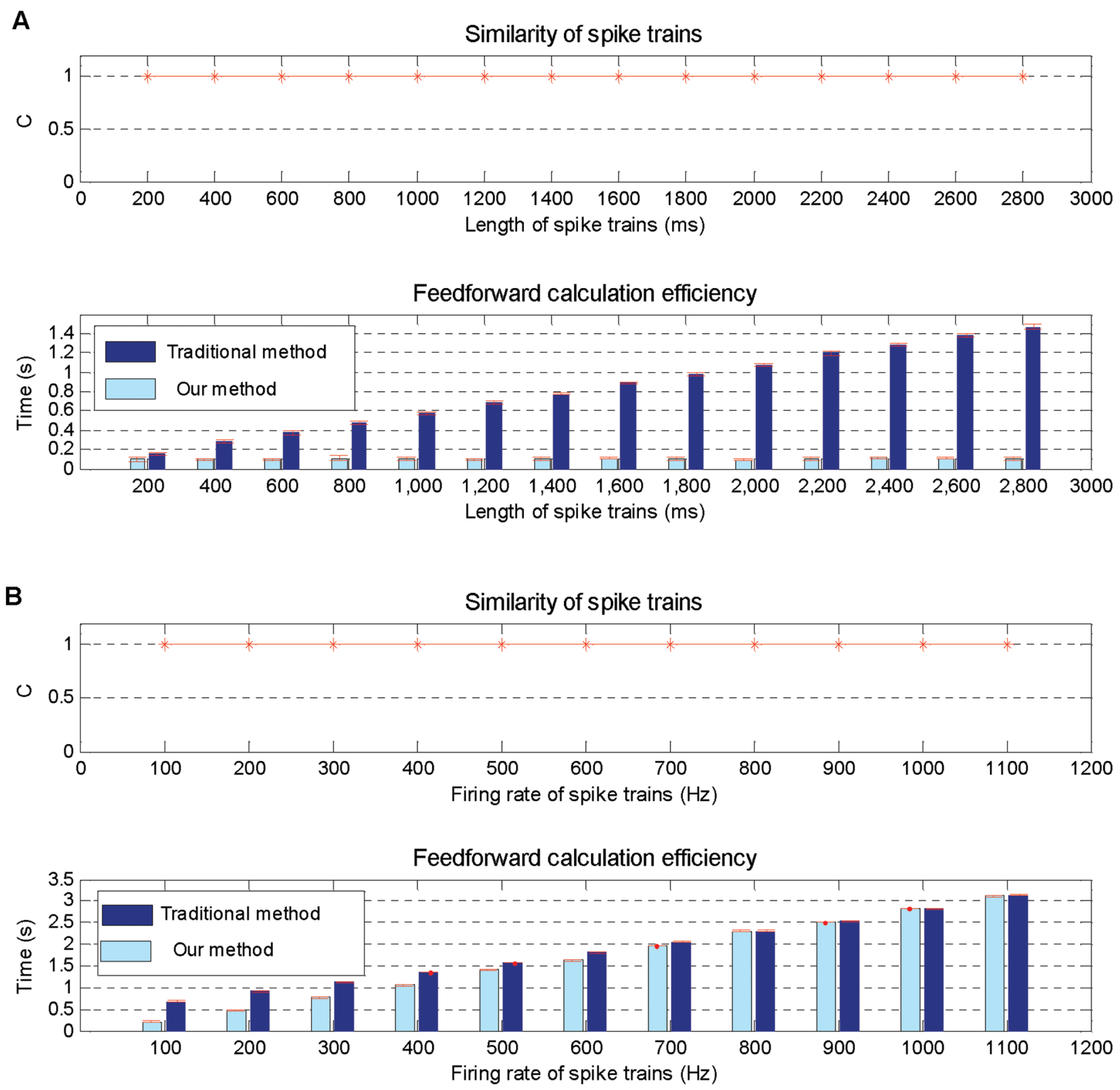

Fig 5. Feedforward calculation performance on various situations. A: Simulation results on different time lengths ranging from 200 ms to 2800 ms. B: Simulation results on different input firing rates ranging from 10 Hz to 1100 Hz.

doi:10.1371/journal.pone.0150329.g005

structure depicted in Fig 4. The input spike train of each input neuron is generated by a homogeneous Poisson process with $r = 10$ Hz, ranging the time length from 200 ms to 2800 ms. The output neuron is desired to emit only one spike in these time lengths because the SpikeProp cannot complete training for multiple target spikes. For fast convergence of traditional algorithms, the target time t_d is set to $t_o + 5$, where t_o is the output firing time of the first epoch. Similar to the previous simulation, C is employed here to measure the accuracy.

The comparison results are shown in Fig 6A, in which the upper sub-figure depicts the learning accuracy of these three algorithms. It illustrates that in this simulation, our algorithm

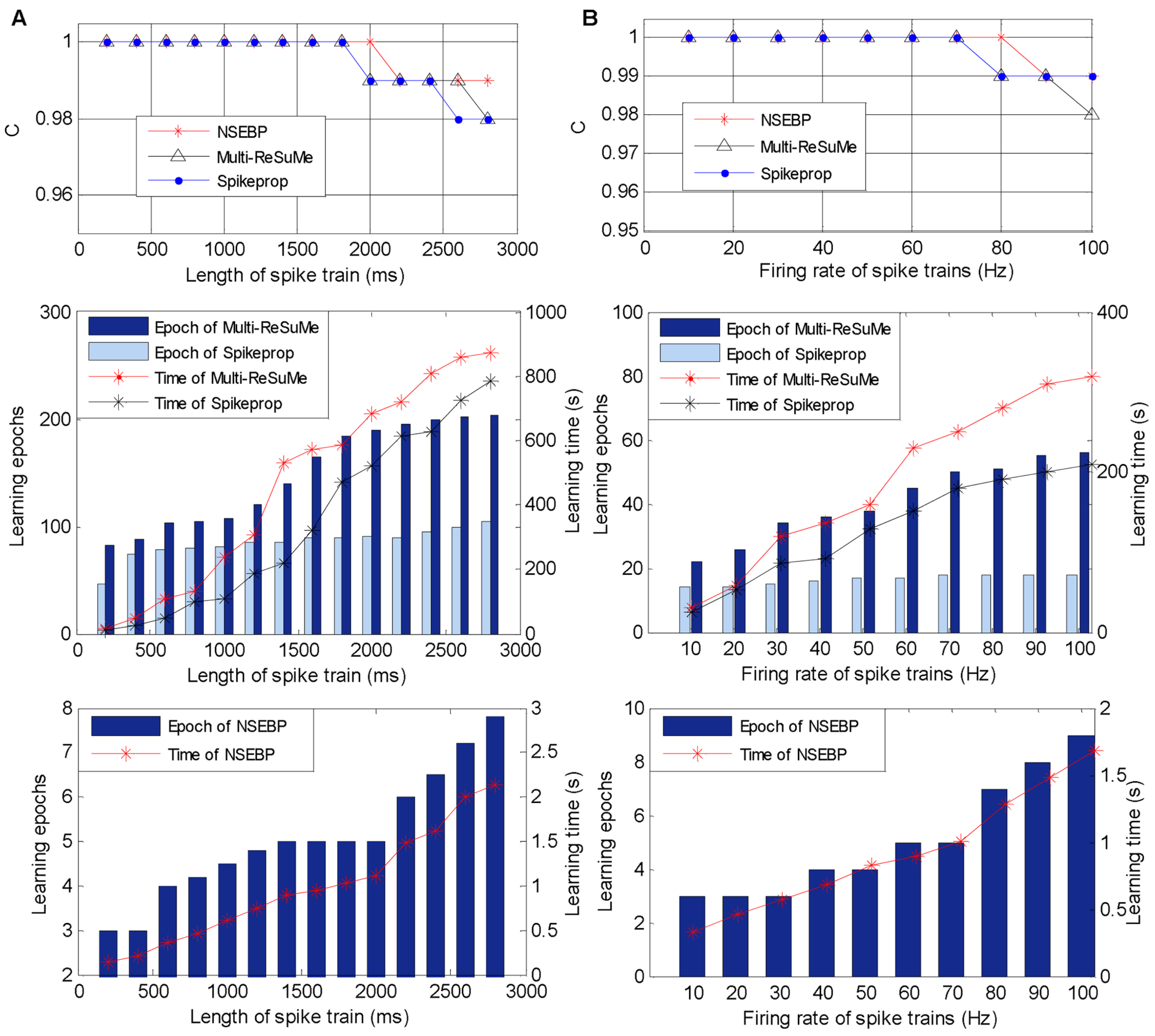

Fig 6. Training performance on various situations. A: Simulation results on different time lengths fixing the input spike rate to 10 Hz. B: Simulation results on different input firing rates with the time length 500 ms.

doi:10.1371/journal.pone.0150329.g006

has a similar accuracy as traditional ones. The middle sub-figure of Fig 6A shows the learning efficiency of the SpikeProp and the Multi-ReSuMe, and the efficiency of our algorithm is displayed independently in the below sub-figure because the magnitude of the learning epochs and learning time of our algorithm is not the same as that of the traditional algorithms. The comparison of these two sub-figure denotes that our algorithm requires less learning epochs and learning time than the SpikeProp and Multi-ReSuMe in various situations.

Table 1. Training time of one epoch for various firing rates.

Firing rate (Hz)	Time of NSEBP (s)	Time of SpikeProp (s)	Time of Multi-ReSuMe (s)
10	0.312	0.812	0.471
20	0.441	1.255	0.814
30	0.552	1.744	1.218
40	0.634	2.185	1.598
50	0.715	2.762	2.077
60	0.746	3.278	2.468
70	0.767	3.817	2.936
80	0.771	4.496	3.422
90	0.781	5.023	3.893
100	0.794	5.591	4.314

doi:10.1371/journal.pone.0150329.t001

The second simulation is conducted to test the learning efficiency of our algorithm under different firing rates, in which the input and output spike trains share the same time length of 500 ms, and the input spike train of each input neuron is generated by a homogeneous Poisson process ranging from 10 Hz to 100 Hz.

The simulation results are shown in Fig 6B, where the upper sub-figure shows the accuracy of these three algorithms, the middle and the below ones depict the learning efficiency of traditional algorithms and our algorithm respectively. Similar with the previous simulation, our algorithm achieves an approximate accuracy with the SpikeProp and Multi-ReSuMe, and improves the training efficiency significantly both in training epochs and training time.

To further explore the learning efficiency of these algorithms, the training time of one epoch is tested in various firing rates and time lengths. Firstly, the time length is fixed to 500 ms, and each input neuron emits a spike train generated by a homogeneous Poisson process with firing rate ranging from 10 Hz to 100 Hz, and the output neuron emits only one spike.

The simulation results shown in Table 1 indicate that the higher the firing rate, the more time required for these three algorithms to complete one training epoch, since more input spikes required to be trained. Besides, our algorithm consumes less time than traditional ones in various firing rates.

Secondly, in order to verify the effect of the time length on the training efficiency, another simulation is carried out where each input neuron emits two spikes and the output neuron emits one spike generated by a homogeneous Poisson process, and the maximum time length varies form 100 ms to 1000 ms.

The simulation results are shown in Table 2, which denotes that in various time lengths, our algorithm has the similar running time for training one epoch, while the time of the SpikeProp and Multi-ReSuMe increases obviously. This reveals the advantage of the selective attention mechanism which enables our NSEBP to concentrate attention on target contents and does not have to scan all time points as traditional algorithms do. It also indicates that the running time of NSEBP has no direct relation to the time length. These simulations in this section demonstrate that our algorithm achieves a significant improvement in efficiency compared with traditional algorithms.

Non-linear Spike Pattern Classification

The XOR Benchmark

In this section, we perform experiments with the NSEBP on a classical example of a non-linear problem, the XOR benchmark to investigate its classification capability and the influence of

Table 2. Training time of one epoch for various time lengths.

Time length (ms)	Time of NSEBP (s)	Time of SpikeProp (s)	Time of Multi-ReSuMe (s)
100	0.117	0.144	0.112
200	0.123	0.238	0.125
300	0.121	0.351	0.146
400	0.123	0.477	0.199
500	0.124	0.611	0.241
600	0.127	0.751	0.293
700	0.123	0.857	0.361
800	0.128	0.978	0.416
900	0.123	1.036	0.537
1000	0.128	1.297	0.595

doi:10.1371/journal.pone.0150329.t002

different parameters. The network architecture shown in Fig 4 is employed in this section, with 4 input neurons, 10 hidden neurons and one output neuron.

The encoded method and the generation of the input spike trains are shown in Fig 7. It depicts that the input 0 and 1 are encoded randomly, which is set to the spike time [1, 2] and [3, 4] respectively in this simulation. Then the four input patterns {0, 0}, {0, 1}, {1, 0}, {1, 1} are encoded by two segments S_1 and S_2 copying the encoded results of 0 and 1. The classification objects are the input spike patterns, among which the input spike trains corresponding to {0, 0} and {1, 1} are one class C_1, and the input spike trains corresponding to {0, 1}, {1, 0} are the other class C_2. The desired outputs of the output neuron corresponding to C_1 and C_2 are set to 10 ms and 15 ms respectively satisfying the convergent condition in Theorem 3.

Our algorithm is applied to the feed-forward network described above with $\vartheta = 1$, $\tau_1 = 5$ and $r = 0.5$. Different from traditional multi-layer networks in [28, 32], our algorithm requires

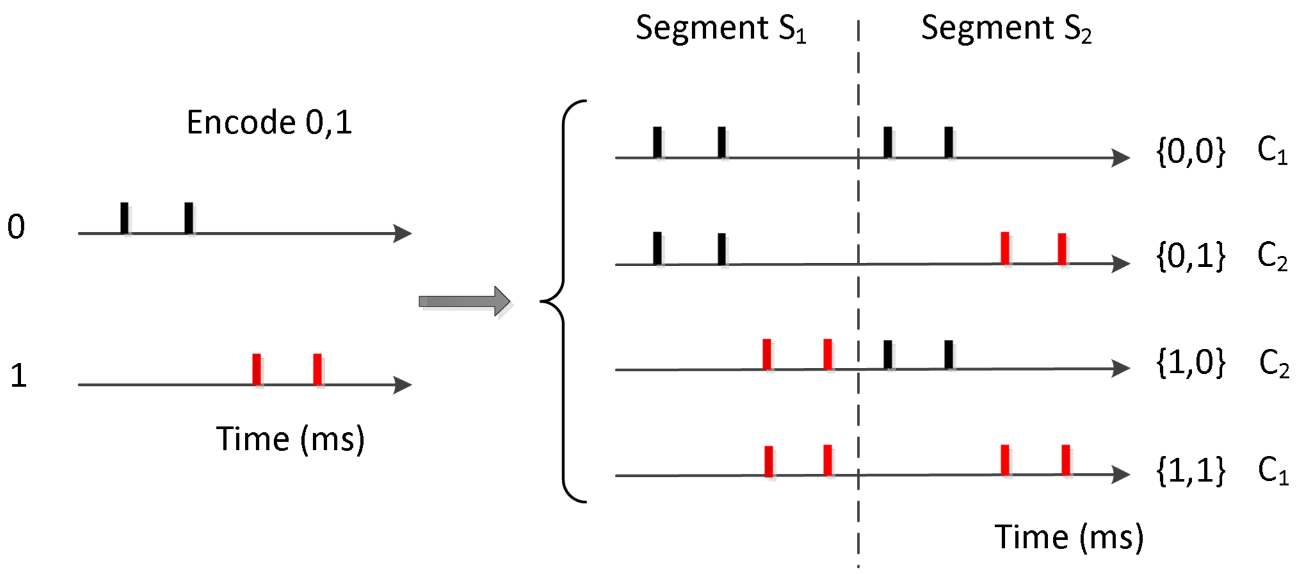

Fig 7. Generation of the input spike trains. Generation of the input spike trains for the classification task in the XOR problem.

doi:10.1371/journal.pone.0150329.g007

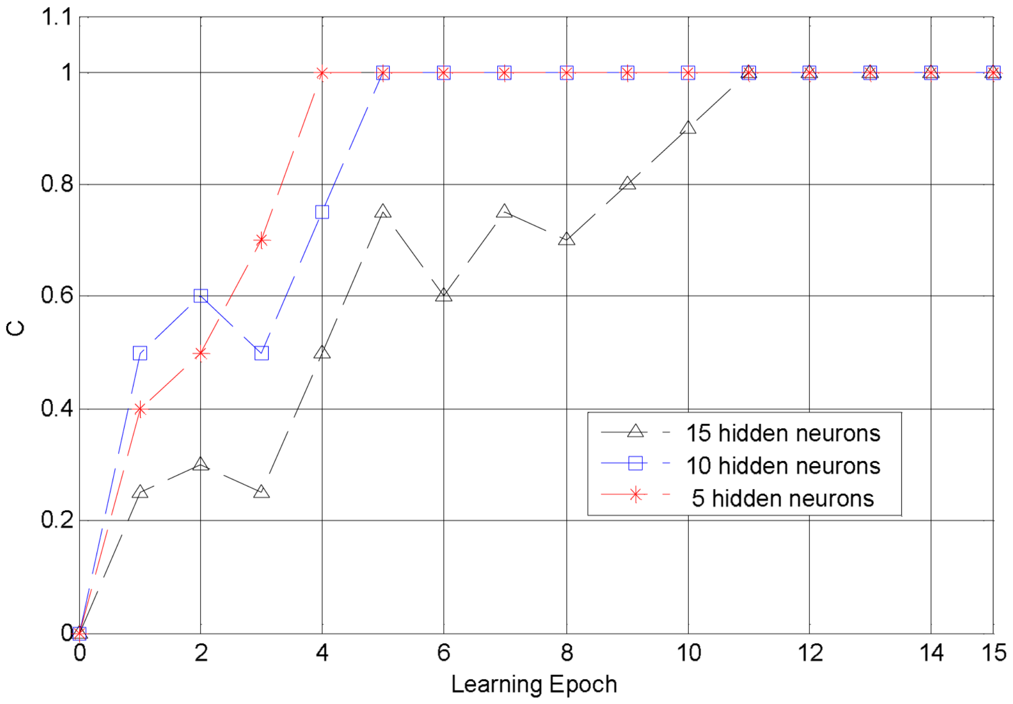

Fig 8. The convergent process of our algorithm. The convergent process of our algorithm with the number of hidden neurons 5, 10, 15. All of these simulations achieve accuracy 1.

doi:10.1371/journal.pone.0150329.g008

none sub-connections, which reduces the number of weight modification. With these parameters, our algorithm can complete training efficiently in 15 learning epochs and achieve accuracy 1 in various number of hidden neurons, as shown Fig 8. This training efficiency is higher than traditional algorithms that is at least 63 epochs in Multi-ReSuMe [32] and 250 cycles in SpikeProp [28]. In the following we systematically vary the parameters of our algorithm and investigate their influence.

The Parameters

In this part, we explore the influence of the parameter τ_1, the number of hidden neurons, the parameter r defined in Eq (6), and the threshold ϑ on the convergent epochs. 50 simulations are carried out and the average learning epoch is obtained.

Fig 9(a) shows the convergent epochs for different values of the threshold ϑ, with the number of hidden neurons fixed to 10, $r = 0.5$, and $\tau_1 = 5$. It suggests that the convergence of our algorithm is insensitive to the threshold, and it can complete training in 8 epochs in various situations.

Fig 9(b) depicts the convergent epoch with different numbers of hidden neurons, which indicates that in the beginning, more neurons in the hidden layer lead to less learning epochs, while when it above 30, this change is not obvious. This is mainly because more hidden neurons make more sparse representation of the input patterns in the hidden layer, which are easier to be trained because there is less interference between different patterns. Different amount of information requires different numbers of hidden neurons for a sparse representation, and if

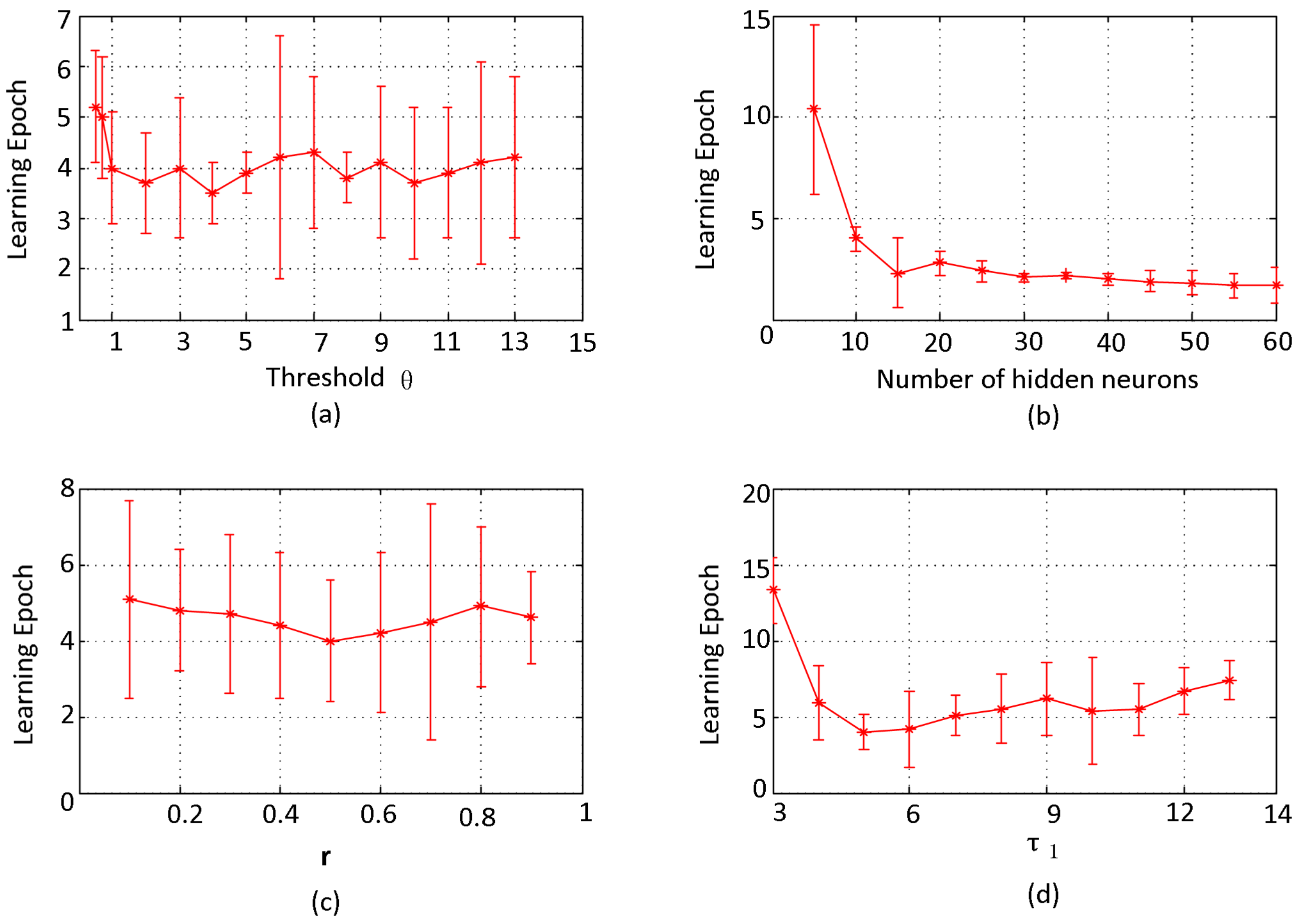

Fig 9. Convergent epochs with various parameters. Convergent epochs with various values of parameter ϑ (a), number of hidden neurons (b), r (c), and τ_1 (d). All of these simulations achieve accuracy 1.

doi:10.1371/journal.pone.0150329.g009

the number of hidden neurons is large enough, the variation of the convergent epoch is not apparent.

Fig 9(c) displays the convergent epoch with different r defined in Eq (6), which determines the proportion of the error back-propagated to the previous layers. The simulation results demonstrate that the convergence of our algorithm has no noticeable relationship with r. To balance the load of each layer, we suggest $r = 1/n$ when there are n layers required to be trained in our algorithm. In real world applications, r can be set to different values according to different requirements. Fig 9(d) shows the convergent epoch for different τ_1, which indicates that our algorithm can achieve rapid convergence in various cases, and different values of τ_1 have some influence on the convergent speed, but not obvious.

Simulations in this section demonstrate that our algorithm can complete non-linear classification efficiently. Besides, simulation results shown in Fig 9 indicate that our algorithm is not sensitive to various parameters, which makes our algorithm more convenient to be applied to various applications.

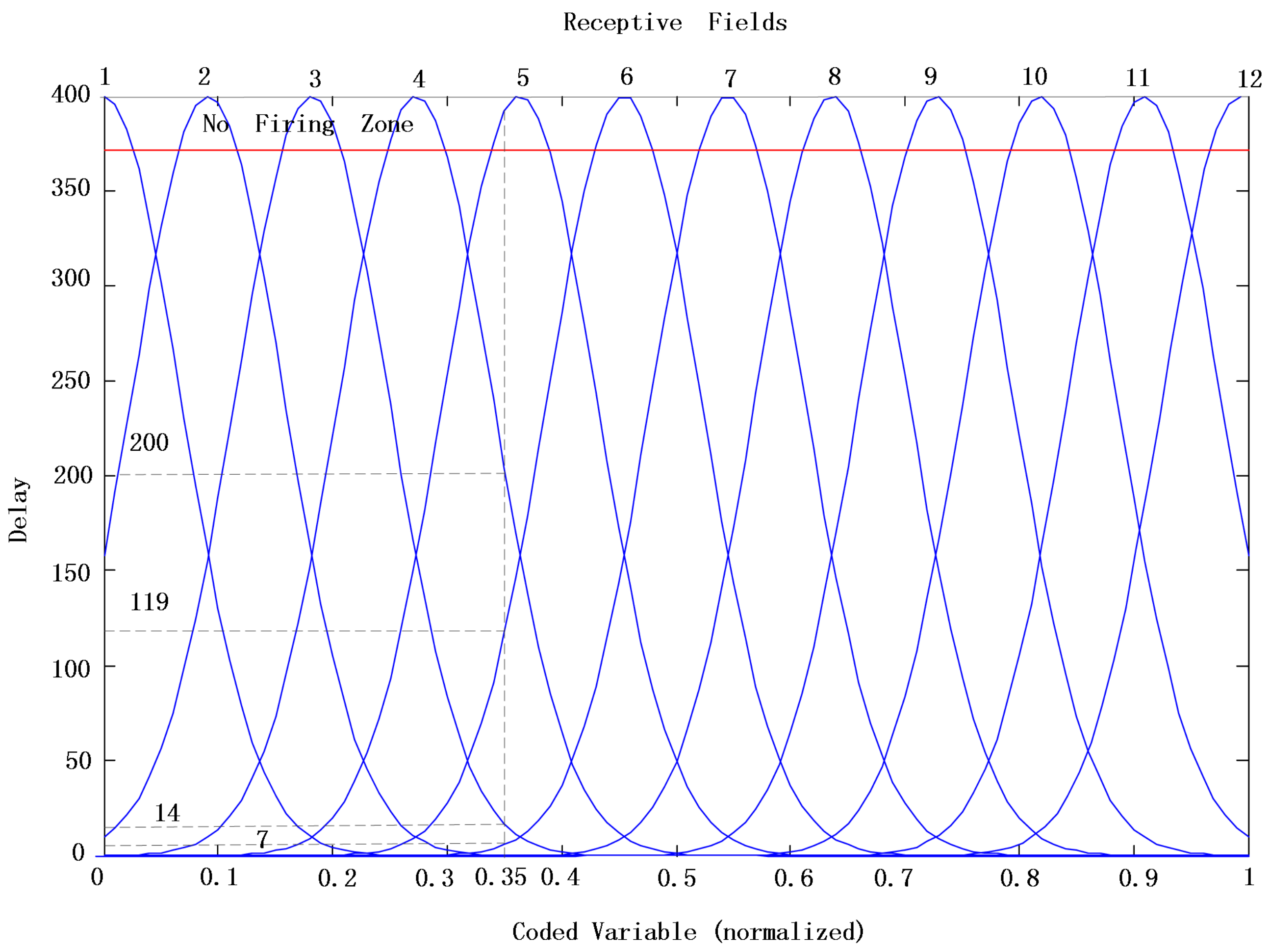

Fig 10. Continuous input variable encoded by means of local receptive fields. The input variable is normalized to [0, 1], and a non-firing zone is defined to avoid spikes in later time. Every no firing neuron has code −1. For instance, 0.35 is encoded to a spike train of 12 neurons: (−1, −1, 14, 200, −1, 119, 7, −1, −1, −1, −1, −1).

doi:10.1371/journal.pone.0150329.g010

Classification on the UCI Datasets

In this section, we apply NSEBP to classify both the Iris and Breast Cancer Wisconsin (BCW) datasets of the UCI [39] to investigate the capability of our algorithm over classification tasks.

Iris Dataset

The Iris dataset is firstly applied to benchmark our algorithm. It contains three classes, each with 50 samples and refers to a type of the iris plant: Iris Setosa (class 1), Iris Versicolour (class 2), and Iris Virginica (class 3) [40]. Each sample contains four attributes: sepal length (feature 1), sepal width (feature 2), petal length (feature 3), and petal width (feature 4).

To make the difference between the data apparent, the data of each feature are mapped into a high dimensional space using the population time encoding method [41]. There are 12 uniforming distributed Gaussian receptive fields in [0, 1], which distribute an input variable over 12 input neurons which are shown in Fig 10.

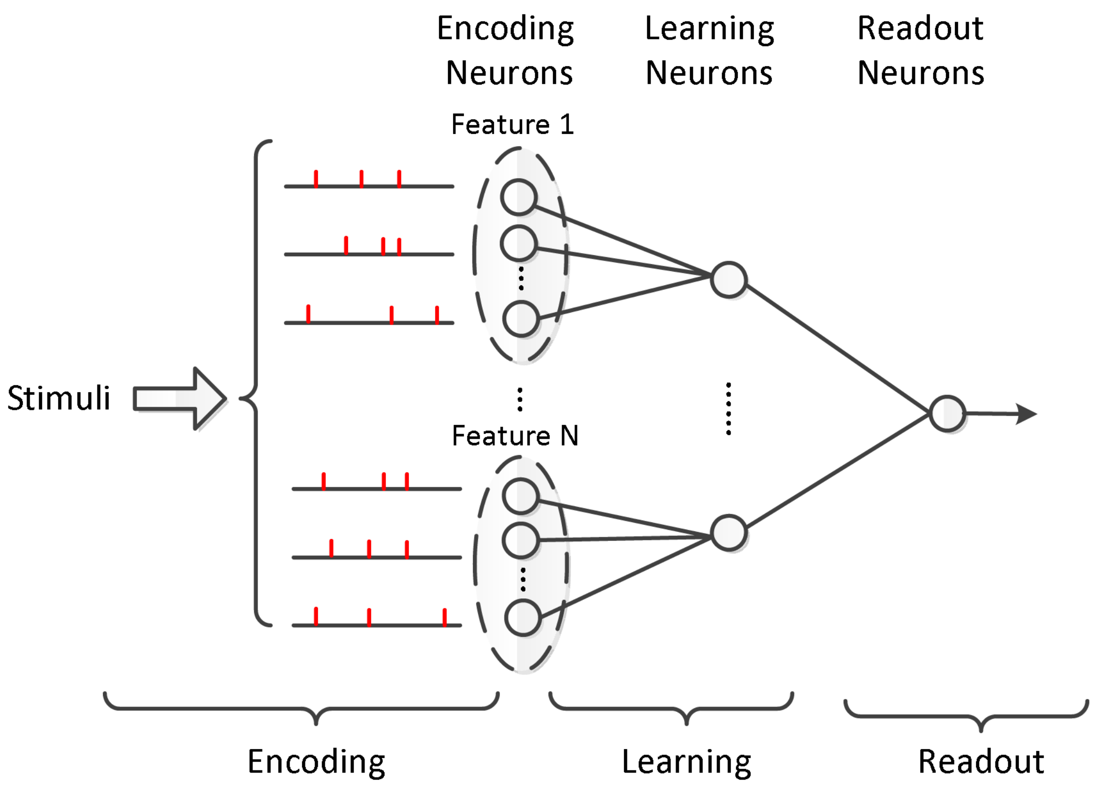

Fig 11. Network structure for classification. Network structure consisting of 12 $*$ F input neurons, F hidden neurons and one output neuron, where F is the number of features.

doi:10.1371/journal.pone.0150329.g011

The network structure devised for this classification task is shown in Fig 11, in the iris data set, the input layer has 48 neurons with each feature 12 inputs. The hidden layer has 4 neurons, each possesses local connections with 12 neurons of the input layer that represent one feature. In the training period, only synaptic weights from input to hidden neurons are adjusted, and the output neuron has weight 1 which is applied to decision making.

In this application, the ith sample of class c has the input spike train $\boldsymbol{ti}_c^i$ for four features and a target spike train $\boldsymbol{td}_c^i$ which is obtained by $\boldsymbol{td}_c^i = \boldsymbol{ti}_c^i + \mathbf{2}\tau_2 \ln \mathbf{2}$, with parameter τ_2 defined in Eq (2). The local connections in our network enable each feature to train an independent sub-network with 12 synapses. Since samples of one class have similar spike trains, encoding results reveal that there are only few different target time trains for each feature. As in the previous simulations, our algorithm chooses the class with minor voltage error.

Table 3 compares the classification accuracy and convergent epoch of the NSEBP with four classical neural network classifier: Multi-ReSuMe [32], Spikeprop [28], MatlabBP, SWAT [22] for the Iris dataset on the training set and testing set. The classifier of the Multi-ReSuMe and Spikeprop are conducted with three layer SNN structures proposed in [32] and [28] respectively. The SWAT employs a SNN architecture of 13 input neurons with each connects to 16 neurons in middle layer, and the squared cosine encoding method is employed [22]. The simulation results are shown in Table 3.

The comparison results indicate that our algorithm achieves comparable or even higher accuracy compared with the traditional classifiers, while our algorithm only requires 18 epochs

Table 3. Comparison Results for Iris Dataset.

Classifier	Training accuracy	Testing accuracy	Training epochs
Matlab BP	0.98	0.95	$2.6 \cdot 10^6$
SpikeProp [28]	0.97	0.96	1000
SWAT [22]	0.95	0.95	500
Multi-ReSuMe [32]	0.96	0.94	174
NSEBP	0.98	0.96	18

doi:10.1371/journal.pone.0150329.t003

to complete training, instead of $2.6 \cdot 10^6$ epochs for the MatlabBP, 1000 epochs for Spikeprop, 500 epochs for the SWAT, and 174 for multi-RuSuMe. The simulation results prove that our algorithm is the most efficient one, and outperforms the compared neural networks methods significantly.

Breast Cancer Wisconsin Dataset

The two-class Breast Cancer Wisconsin (BCW) dataset is also applied to analyze our algorithm. This dataset contains 699 samples, while 16 samples are abandoned because of missing data. Each sample has nine features obtained from a digitized image of a fine needle aspirate (FNA) of a breast mass [42].

The same network structure and training settings as these in the Iris classification simulations are employed here. There are nine features instead of four, then there are 108 input neurons and 9 hidden neurons in the network structure depicted in Fig 11.

Table 4 compares the accuracy and efficiency of the NSEBP against the existing algorithms for the BCW dataset. It shows that the test data accuracy of the NSEBP is comparable to that of the other approaches, while the NSEBP only requires 16 epochs for complete training, instead of 1500 epochs for the Spikeprop, 500 epochs for SWAT, and $9.2 \cdot 10^6$ epochs for MatlabBP. Then, the training efficiency of our algorithm is improved significantly compared with these classical neural network algorithms.

Simulation results in this section demonstrate that our algorithm achieves a higher efficiency and even a higher learning accuracy than the traditional neural network methods in the classification tasks. The selective mechanism and the presynaptic spike jitter adopted in our algorithm make the efficiency of the NSEBP to be independent with the length of the spike train, and overcome the drawbacks of low efficiency in traditional back-propagation methods. With this efficiency, the training of SNNs can meet the requirement of real world applications.

Table 4. Comparison Results for BCW Dataset.

Classifier	Training accuracy	Testing accuracy	Training epochs
MatlabBP	0.98	0.96	$9.2 \cdot 10^6$
SpikeProp [28]	0.98	0.97	1500
SWAT [22]	0.96	0.96	500
NSEBP	0.97	0.96	16

doi:10.1371/journal.pone.0150329.t004

Conclusion

In this paper, an efficient multi-layer supervised learning algorithm, the NSEBP, is proposed for spiking neural networks. The accurate feedforward calculation and weight modification employing the normalized PSP learning window 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 un-target ones, which makes our algorithm to achieve a significant improvement in efficiency of training one epoch. Simulation results demonstrate that our algorithm outperforms traditional learning algorithms in learning efficiency, and is not sensitive to parameters.

The classification results on the UCI data sets indicate that the generalization ability of our algorithm is a little better than the traditional backpropagation method, and similar to the SpikeProp, but lower than the SWAT. It means that our algorithm does not make great contribution to the over fitting problem. However, the traditional methods to improve the training generalization ability can also be applied to our algorithm, such as employing an optimized network structure and better decision-making method, or better sample validate methods, that will be studied in the future work.

Our algorithm is derived from the SRM_0 model, but the same derivation process is feasible to other models when the voltage u can be expressed by an equation of time t and can be transformed to a quadratic function by the substitute method. Besides, the proposed feed forward calculation method can be applied to the existing algorithms to improve their learning performance. Consequently, employing these training strategies, the SNNs can be applied efficiently to various applications with multilayer network structure and arbitrary real-valued analog inputs.

Supporting Information

S1 Table. Experiment Data for Feedforward Calculation Performance.
(XLSX)

S2 Table. Experiment Data for Training Performance with Different Time Lengths.
(XLSX)

S3 Table. Experiment Data for Training Performance with Different Firing Rates.
(XLSX)

S4 Table. Experiment Data for Training time of one epoch.
(XLSX)

S5 Table. Experiment Data for The Convergent Process of Our Algorithm with Different Number of Hidden Neurons.
(XLSX)

S6 Table. Experiment Data for The Convergent Epochs with Different Parameters.
(XLSX)

S7 Table. Experiment Data for Iris Dataset.
(XLSX)

S8 Table. Experiment Data for Breast Cancer Winsconsin Dataset.
(XLSX)

S1 File. Data Descriptions.
(DOCX)

Acknowledgments

This work was partially supported by the National Science Foundation of China under Grants 61273308 and 61573081, and the China Scholarship Council.

Author Contributions

Conceived and designed the experiments: XX HQ. Performed the experiments: XX MZ. Analyzed the data: XX GL. Contributed reagents/materials/analysis tools: XX MZ. Wrote the paper: XX JK.

References

1. Theunissen F, Miller JP. Temporal encoding in nervous systems: a rigorous definition. Journal of computational neuroscience. 1995; 2: 149–162. doi: 10.1007/BF00961885 PMID: 8521284
2. Rullen R V, Guyonneau R, Thorpe SJ. Spike times make sense. Trends in Neurosciences. 2005; 28: 1–4. doi: 10.1016/j.tins.2004.10.010
3. Hu J, Tang H, Tan KC, Li H, Shi L. A Spike-Timing-Based Integrated Model for Pattern Recognition. Neural Computation. 2013; 25: 450–472. doi: 10.1162/NECO_a_00395 PMID: 23148414
4. O'Brien MJ, Srinivasa NA. Spiking Neural Model for Stable Reinforcement of Synapses Based on Multiple Distal Rewards. Neural Computation. 2013; 25:123–156. doi: 10.1162/NECO_a_00387 PMID: 23020112
5. Naveros F, Luque NR, Garrido JA. A Spiking Neural Simulator Integrating Event-Driven and Time-Driven Computation Schemes Using Parallel CPU-GPU Co-Processing: A Case Study. IEEE Transactions on Neural Networks and Learning Systems. 2015; 26: 1567–1574. doi: 10.1109/TNNLS.2014.2345844 PMID: 25167556
6. Zhang Z, Wu QX. Wavelet transform and texture recognition based on spiking neural network for visual images. Neurocomputing. 2015; 151: 985–995. doi: 10.1016/j.neucom.2014.03.086
7. McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics. 1943; 5: 115–133. doi: 10.1007/BF02478259
8. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review. 1958; 65: 386. doi: 10.1037/h0042519 PMID: 13602029
9. Tiesinga P, Fellous JM, Sejnowski TJ. Regulation of spike timing in visual cortical circuits. Nature reviews neuroscience. 2008; 9: 97–107. doi: 10.1038/nrn2315 PMID: 18200026
10. Mehta MR, Lee AK. Role of experience and oscillations in transforming a rate code into a temporal code. Nature. 2002; 417: 741–746. doi: 10.1038/nature00807 PMID: 12066185
11. Rullen RV, Thorpe SJ. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. Neural Computation. 2001; 13: 1255–1283. doi: 10.1162/08997660152002852 PMID: 11387046
12. Bohte S.M. The Evidence for Neural Information Processing with Precise Spike-times: A Survey. Natural Computation. 2004; 3: 195–206. doi: 10.1023/B:NACO.0000027755.02868.60
13. Benchenanel K, Peyrachel A, Khamassi M, Tierney PL, Gioanni Y, Battaglia FP, et al. Coherent theta oscillations and reorganization of spike timing in the hippocampal-prefrontal network upon learning. Neuron. 2010; 66: 921–936. doi: 10.1016/j.neuron.2010.05.013
14. Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing-based decisions. Nature neuroscience. 2006; 9: 420–428. doi: 10.1038/nn1643 PMID: 16474393
15. Florian RV. The chronotron: a neuron that learns to fire temporally precise spike patterns. Plos one. 2012; 7: e40233. doi: 10.1371/journal.pone.0040233 PMID: 22879876
16. Mohemmed A, Schliebs S, Matsuda S, Kasabov N. Span: Spike pattern association neuron for learning spatio-temporal spike patterns. International Journal of Neural Systems. 2012; 22: 1250012 doi: 10.1142/S0129065712500128 PMID: 22830962
17. Victor JD, Purpura KP. Metric-space analysis of spike trains: theory, algorithms and application. Network: computation in neural systems. 1997; 8: 127–164 doi: 10.1088/0954-898X_8_2_003
18. van Rossum MC. A novel spike distance. Neural Computation. 2001; 13: 751–763. doi: 10.1162/089976601300014321 PMID: 11255567
19. Masquelier T, Guyonneau R, Thorpe SJ. Competitive STDP-based spike pattern learning. Neural computation. 2009; 21: 1259–1276. doi: 10.1162/neco.2008.06-08-804 PMID: 19718815

20. Ponulak F, Kasinski A. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. Neural Computation. 2010; 22: 467–510. doi: 10.1162/neco.2009.11-08-901 PMID: 19842989

21. Xu Y, Zeng X, Zhong S. A new supervised learning algorithm for spiking neurons. Neural computation. 2013; 25: 1472–1511. doi: 10.1162/NECO_a_00450 PMID: 23517101

22. Wade JJ, McDaid LJ, Santos J, Sayers HM. SWAT: a spiking neural network training algorithm for classification problems. IEEE Transactions on Neural Networks. 2010; 21: 1817–1830. doi: 10.1109/TNN.2010.2074212 PMID: 20876015

23. Yu Q, Tang H, Tan KC, Li H. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns. Plos one. 2013; 8: e78318. doi: 10.1371/journal.pone.0078318 PMID: 24223789

24. Ponulak F, Kasiński A. Introduction to spiking neural networks: Information processing, learning and applications. Acta neurobiologiae experimentalis. 2010; 71: 409–433.

25. Hubel DH, Wiesel TN. Receptive fields and functional architecture of monkey striate cortex. The Journal of physiology. 1968; 195: 215–243. doi: 10.1113/jphysiol.1968.sp008455 PMID: 4966457

26. Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology. 1962; 160: 106–154 doi: 10.1113/jphysiol.1962.sp006837 PMID: 14449617

27. Song W, Kerr CC, Lytton WW, Francis JT. Cortical plasticity induced by spike-triggered microstimulation in primate somatosensory cortex. Plos one. 2013; 8(3): e57453. doi: 10.1371/journal.pone.0057453 PMID: 23472086

28. Bohte SM, Kok JN, La Poutre H. Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing. 2002; 48: 17–37. doi: 10.1016/S0925-2312(01)00658-0

29. McKennoch S, Liu D, Bushnell LG. Fast Modifications of the SpikeProp Algorithm. IEEE International Joint Conference on Neural Networks, IEEE. 2006; 48: 3970–397

30. Ghosh-Dastidara S, Adeli H. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. Neural Networks. 2009; 22: 1419–1431. doi: 10.1016/j.neunet.2009.04.003

31. Xu Y, Zeng X, Han L, Yang J. A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. Neural Networks. 2013; 43: 99–113. doi: 10.1016/j.neunet.2013.02.003 PMID: 23500504

32. Sporea I, Grüning A. Supervised Learning in Multilayer Spiking Neural Networks. Neural Computation. 2013; 25: 473–509. doi: 10.1162/NECO_a_00396 PMID: 23148411

33. Thorpe SJ, Imbert M. Biological constraints on connectionist modelling. Connectionism in perspective. 1989; 63–92.

34. Fink C G, Zochowski M, Booth V. Neural network modulation, dynamics, and plasticity. Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2013; 843–846. doi: 10.1109/GlobalSIP.2013.6737023

35. Desimone R, Ungerleider LG. Neural mechanisms of visual processing in monkeys. Handbook of neuropsychology. 1989; 2: 267–299

36. Ungerleider SKALG. Mechanisms of visual attention in the human cortex. Annual Review of Neuroscience. 2000; 23: 315–341. doi: 10.1146/annurev.neuro.23.1.315 PMID: 10845067

37. Gerstner W, Kistler WM. Spiking Nerual Models: Single Neurons, Populations, Plasticity. 1st ed. Cambridge: Cambridge University Press. 2002.

38. Schreiber S, Fellous JM. A new correlation-based measure of spike timing reliability. Neurocomputing. 2003; 52:925–931. doi: 10.1016/S0925-2312(02)00838-X PMID: 20740049

39. Bache K, Lichman M. UCI Machine Learning Repository; 2013. Accessed: http://archive.ics.uci.edu/ml

40. Fisher RA. The use of multiple measurements in taxonomic problems. Annals of eugenics. 1936; 7: 179–188 doi: 10.1111/j.1469-1809.1936.tb02137.x

41. Snippe HP. Parameter extraction from population codes: A critical assessment. Neural Computation. 1996; 8: 511–529. doi: 10.1162/neco.1996.8.3.511 PMID: 8868565

42. Wolberg WH, Mangasarian OL. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceedings of the national academy of sciences. 1990; 87: 9193–9196. doi: 10.1073/pnas.87.23.9193