THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 26, Number 1/2025, pp. 89–99
DOI: 10.59277/PRA-SER.A.26.1.11

DELAY LEARNING FOR SPIKING NEURONS WITH MULTIPLE SYNAPTIC CONNECTIONS

Li ZOU[1], Suichan WANG[1], Hong ZHANG[1], Tongrui LIANG[1], Ming ZHANG[1], Xiangwen WANG[2]

[1] Gansu Provincial Meteorological Information and Technic Support and Equipment Center, Lanzhou, 730020, China
[2] Northwest Normal University, College of Computer Science and Engineering, Lanzhou, 730070, China
Corresponding author: Xiangwen WANG, E-mail: wangxw2015@nwnu.edu.cn

Abstract. Recent studies in neuroscience have shown that synaptic delays are widespread in biological nervous systems and positively affect their information processing capacity. Moreover, the pattern of multiple synaptic connections between neurons is widespread in nervous systems, which increases the complexity and diversity of neural signal transmission. However, most of the existing supervised learning algorithms for spiking neural networks (SNNs) only consider the learning of synaptic weights while ignoring the dynamic adjustment of synaptic delays, and SNN modeling mostly uses single synaptic connections, which limits the potential performance of SNNs. In this paper, we combine the information processing mode of SNNs with the synergistic computation mode of synaptic weights and delays, and apply the kernel function representation of spike trains to propose offline and online supervised learning algorithms for synaptic weights and delays of spiking neurons with multiple synaptic connections. The learning performance of the proposed algorithms is verified through a series of spike train learning tasks and nonlinear pattern recognition problems on UCI datasets. The results show that the proposed algorithms can achieve high spike train learning accuracy and higher pattern classification accuracy than other supervised learning algorithms for SNNs, suggesting that the pattern of multiple synaptic connections and synaptic weight-delay synergistic plasticity can effectively enhance the learning performance of SNNs.

***Keywords**:* spiking neural networks, supervised learning, synaptic delay plasticity, spike train kernel.

1. INTRODUCTION

Spiking neural network (SNN) is a new generation of brain-inspired artificial neural network computational models with strong biological plausibility. At present, SNNs have become one of the research hotspots in the field of artificial intelligence, and have made many breakthroughs in pattern recognition applications and hardware implementation [1]. Supervised learning is one of the core contents in SNN research, which is usually realized by information forward propagation and error back propagation. The main goal of supervised learning tasks in SNNs is to design adjustment rules for learnable parameters based on the input, actual output and desired output spike trains of the network model [2]. Currently, researchers have developed a number of supervised learning algorithms for SNNs, but these algorithms need further in-depth research in terms of learning performance and their applications in real-world pattern recognition [3].

SNNs typically have two supervised learning strategies, namely offline learning and online learning, which differ in the way they update the learnable parameters of the network [4]. During the offline learning process, each training sample is first encoded into a spike train and input into the network, and then the network calculates the loss function according to the actual output spike trains and the corresponding desired output spike trains after a complete learning epoch, and finally updates all the learnable parameters of the network at once. During the online learning process, as soon as the output neuron fires a spike or encounters a desired output spike, the network immediately calculates the loss function and updates the learnable parameters of the network. Online learning algorithms typically have low complexity and fast computational speed, making them particularly useful for dealing with large real-time data streams or real-time applications. Therefore, it is of great importance to design online supervised learning algorithms for SNNs.

Recent studies in neuroscience have shown that synaptic delays are ubiquitous in biological nervous systems and positively influence their information processing capacity. Traditional research on supervised learning in SNNs mainly focuses on synaptic weight learning, and synaptic delays are often ignored or considered as static during the learning process. In recent years, researchers have proposed some synaptic delay learning algorithms for single-layer SNNs. Taherkhani *et al.* [5] introduced dynamic delays into the ReSuMe algorithm based on the spike-timing dependent plasticity (STDP) mechanism and proposed the synaptic delay learning algorithm DL-ReSuMe, which improves both learning accuracy and learning speed. Guo *et al.* [6] further introduced synaptic delays and axonal delays into the ReSuMe algorithm, and proposed a supervised learning algorithm for single-layer SNNs that can dynamically adjust synaptic delays and axonal delays, and applied it to the classification problem of epilepsy EEG data. Zhang *et al.* [7] proposed a synaptic delay plasticity mechanism based on the temporal relationship between presynaptic and postsynaptic spikes, and applied it to the ReSuMe and PBSNLR algorithms, improving the learning performance of the original algorithms. Yu *et al.* [8] introduced synaptic delays into their threshold-driven plasticity algorithm and proposed a delay learning algorithm TDP-DL, which improves the performance of the original algorithm. Compared with synaptic weights learning algorithms for SNNs, the research on synaptic delay learning is not sufficient, and most of the existing synaptic delay learning algorithms are offline learning algorithms. It is necessary to design high-performance online synaptic delay learning algorithms suitable for SNNs.

In biological nervous systems, neural information is transmitted from a presynaptic neuron to a postsynaptic neuron through multiple synapses, rather than directly through a single synapse [9]. The multiple synaptic connection mode is very common in the nervous system, especially in complex neural networks such as the cerebral cortex. It allows neurons to process and manipulate input information at a finer level because each synapse may have different effects on neural information, such as strengthening, weakening, or changing the nature of the information. In addition, multiple synaptic connections increase the plasticity of the nervous system because the number and strength of synaptic connections can be changed based on experience and learning. Therefore, constructing an SNN model with multiple synaptic connections between presynaptic and postsynaptic neurons and designing corresponding supervised learning algorithms is an effective way to study high-performance SNN learning systems. However, most of the existing studies use a single synaptic connection in SNN modelling, which limits the performance of SNNs.

In this paper, aiming at the drawbacks of ignoring synaptic delays and modelling with single synaptic connections in supervised learning for SNNs, we construct an SNN model with multiple synaptic connections and propose a synaptic weight-delay synergistic learning approach using the kernel function representation of spike trains, and provide offline and online adjustment rules for synaptic weights and delays. The main contributions of this paper can be summarized as follows: (1) A synergistic supervised learning approach for synaptic weights and delays is proposed, where both the weights and delays can be dynamically adjusted in offline and online learning modes. (2) The multiple synaptic connection mode is considered in SNN modelling, which can enhance the learning performance of SNNs. (3) The learning performance of dynamic and static delay learning in both offline and online learning modes is compared to highlight the role of synaptic delay learning and online learning.

The remainder of this paper is organized as follows. In Section 2, the proposed supervised learning algorithms based on spike train kernels for spiking neurons are derived in detail. In Section 3, the learning performance of the proposed learning algorithms is demonstrated through a series of spike train learning tasks and pattern recognition tasks on the UCI datasets. Finally, we discuss and conclude this paper in Section 4.

2. METHODOLOGY

In this section, the proposed learning rules for synaptic weights and delays are presented in detail. First, the spike train and its kernel function representation are introduced. Then, the online and offline supervised learning algorithms for synaptic weights and delays are derived in detail using the gradient descent rule. Finally, the process of training SNNs using the proposed supervised learning rules was presented using pseudocode, including both offline and online learning.

2.1. Spike train and its kernel function representation

Spike train $s(t)=\{t^f \in \Gamma : f=1,2,\cdots\}$ refers to the set of ordered events in which a neuron fires a spike t^f during a time interval Γ. We consider synaptic delays in the propagation of input spike events. Therefore, the spike train of the input neuron *i* transmitted through its *k*th synapse can be formally defined as follows:

$$s_i(t-d_i^k(t))=\sum_{f=1}^{N_i}\delta(t-t_i^f-d_i^k(t)) \tag{1}$$

where $d_i^k(t)$ is the synaptic delay for the *k*-th synapse of the presynaptic neuron *i*. N_i is the total number of spikes transmitted by the presynaptic neuron *i*, and t_i^f is the *f*-th spike ($f\in[1,N_i]$) in the input spike train $s_i(t-d_i^k(t))$. $\delta(\bullet)$ is the Dirac delta function, $\delta(x)=1$ if $x=0$ and $\delta(x)=0$ otherwise.

In this paper, we apply the kernel function representation of spike trains to design supervised learning algorithms of synaptic weights and delays for spiking neurons. A specific kernel function $\kappa(s)$ is chosen to uniquely transform the discrete spike train $s(t)$ into a continuous function through convolution calculation. The continuous functions $f_{s_i}(t-d_i^k(t))$, $f_{s_o}(t)$, and $f_{s_d}(t)$ corresponding to the input spike train $s_i(t-d_i^k(t))$, the actual output spike train $s_o(t)$, and the desired output spike train $s_d(t)$ can be formally defined as follows:

$$f_{s_i}(t-d_i^k(t))=s_i(t-d_i^k(t))\otimes\kappa(t-d_i^k(t))=\sum_{f=1}^{N_i}\kappa(t-t_i^f-d_i^k(t)) \tag{2}$$

$$f_{s_o}(t)=s_o(t)\otimes\kappa(t)=\sum_{h=1}^{N_o}\kappa(t-t_o^h) \tag{3}$$

$$f_{s_d}(t)=s_d(t)\otimes\kappa(t)=\sum_{g=1}^{N_d}\kappa(t-t_d^g) \tag{4}$$

where ⊗ represents the convolution calculation. t_o^h and t_d^g are spikes in $s_o(t)$ and $s_d(t)$, respectively. N_o and N_d are the total number of spikes in $s_o(t)$ and $s_d(t)$, respectively. The relationship between the postsynaptic spiking activity and the contributions of all presynaptic spiking activity can be expressed as a linear relationship through the convolved continuous functions [10]:

$$f_{s_o}(t)=\sum_{i=1}^{N_I}\sum_{k=1}^{N_S}w_i^k(t)f_{s_i}(t-d_i^k(t)) \tag{5}$$

where $w_i^k(t)$ is the synaptic weight between the presynaptic neuron *i* and the postsynaptic neuron, N_I and N_S are the numbers of presynaptic neurons and synaptic connections.

2.2. Synaptic weight learning rule

Gradient descent is a commonly used optimization method for designing supervised learning algorithms for SNNs, where the key premise is to define the mean square error according to the actual and desired output spike trains of the network. In the online learning scenario of SNNs, when the output neuron fires a spike or encounters a desired output spike, the SNN needs to update its learnable parameters and thus the real-time error function needs to be defined. Using the kernel function representation of the spike trains, the real-time error function is defined as follows:

$$E(t)=\frac{1}{2}\left[f_{s_o}(t)-f_{s_d}(t)\right]^2 \tag{6}$$

Applying the gradient descent rule, the change in weight $\Delta w_i^k(t)$ of the *k*th synapse of the presynaptic neuron *i* at time *t* can be calculated as the gradient of the error *E*(*t*) over the current synaptic weight $w_i^k(t)$ scaled by a synaptic weight learning rate η_w:

$$\Delta w_i^k(t) = -\eta_w \nabla E_w = -\eta_w \frac{\partial E(t)}{\partial w_i^k(t)} = -\eta_w \frac{\partial E(t)}{\partial f_{s_o}(t)} \frac{\partial f_{s_o}(t)}{\partial w_i^k(t)} \tag{7}$$

According to the error function defined in Eq. 6, the first partial derivative term on the right-hand side of Eq. 7 can be calculated as follows:

$$\frac{\partial E(t)}{\partial f_{s_o}(t)} = \frac{\partial\left(\frac{1}{2}\left[f_{s_o}(t) - f_{s_d}(t)\right]^2\right)}{\partial f_{s_o}(t)} = f_{s_o}(t) - f_{s_d}(t) \tag{8}$$

According to the linear relationship between the presynaptic and postsynaptic spike train defined in Eq. 5, the second derivative term on the right-hand side of Eq. 7 can be calculated as follows:

$$\frac{\partial f_{s_o}(t)}{\partial w_i^k(t)} = \frac{\partial\left[\sum_{i=1}^{N_I}\sum_{k=1}^{N_S} w_i^k(t) f_{s_i}(t - d_i^k(t))\right]}{\partial w_i^k(t)} = f_{s_i}(t - d_i^k(t)) \tag{9}$$

By substituting Eqs. 8 and 9 into Eq. 7, the online learning rule for synaptic weights can be obtained, which is represented as follows:

$$\Delta w_i^k(t) = \eta_w \left[f_{s_d}(t) - f_{s_o}(t)\right] f_{s_i}(t - d_i^k(t)) \tag{10}$$

Further integrating the online learning rule in Eq. 10, the offline learning rule for synaptic weights can be obtained, which can be represented as follows:

$$\Delta w_i^k = \int_\Gamma \Delta w_i^k(t) dt = \eta_w \left[\sum_{g=1}^{N_d}\sum_{f=1}^{N_i} \kappa(t_d^g - t_i^f - d_i^k) - \sum_{h=1}^{N_o}\sum_{f=1}^{N_i} \kappa(t_o^h - t_i^f - d_i^k)\right] \tag{11}$$

2.3. Synaptic delay learning rule

Similar to the derivation of the synaptic weight learning rule, the change in the delay $\Delta d_i^k(t)$ of the *k*-th synapse of the presynaptic neuron *i* at time *t* can be calculated as the gradient of the error *E*(*t*) over the current synaptic delay $d_i^k(t)$ scaled by a synaptic delay learning rate η_d:

$$\Delta d_i^k(t) = -\eta_d \nabla E_d = -\eta_d \frac{\partial E(t)}{\partial d_i^k(t)} = -\eta_d \frac{\partial E(t)}{\partial f_{s_o}(t)} \frac{\partial f_{s_o}(t)}{\partial d_i^k(t)} \tag{12}$$

The first partial derivative term on the right-hand side of Eq. 12 can be calculated according to Eq. 8. According to Eq. 5, the second derivative term on the right-hand side of Eq. 12 can be calculated as follows:

$$\frac{\partial f_{s_o}(t)}{\partial d_i^k(t)} = \frac{\partial\left[\sum_{i=1}^{N_I}\sum_{k=1}^{N_S} w_i^k(t) f_{s_i}(t - d_i^k(t))\right]}{\partial d_i^k(t)} = w_i^k(t) \frac{\partial\left[\sum_{f=1}^{N_i} \kappa(t - t_i^f - d_i^k(t))\right]}{\partial d_i^k(t)} \tag{13}$$

For simplicity, we use the Laplacian kernel function $\kappa(s)=\exp(-|s|/\tau_k)$ to calculate the partial derivative term on the right-hand side of Eq. 13, which can be calculated as follows:

$$\frac{\partial\left[\sum_{f=1}^{N_i}\kappa(t-t_i^f-d_i^k(t))\right]}{\partial d_i^k(t)}=\frac{1}{\tau_\kappa}\sum_{f=1}^{N_i}\exp\left(-\frac{\left|t-t_i^f-d_i^k(t)\right|}{\tau_\kappa}\right)=\frac{1}{\tau_\kappa}f_{s_i}(t-d_i^k(t)) \tag{14}$$

By substituting Eqs. 8, 13, and 14 into Eq. 12, the real-time adjustment rule for synaptic delays is obtained and can be represented as follows:

$$\Delta d_i^k(t)=\eta_d\frac{1}{\tau_\kappa}w_i^k(t)\left[f_{s_d}(t)-f_{s_o}(t)\right]f_{s_i}(t-d_i^k(t))=\eta_d\frac{1}{\tau_\kappa}\frac{1}{\eta_w}w_i^k(t)\Delta w_i^k(t) \tag{15}$$

Since η_d, τ_k and η_w in Eq. 15 are constants set before training the network, we assume that $\eta_d=\eta_d/(\tau_k\cdot\eta_w)$. In this way, the online learning rule for synaptic delays can be represented as follows:

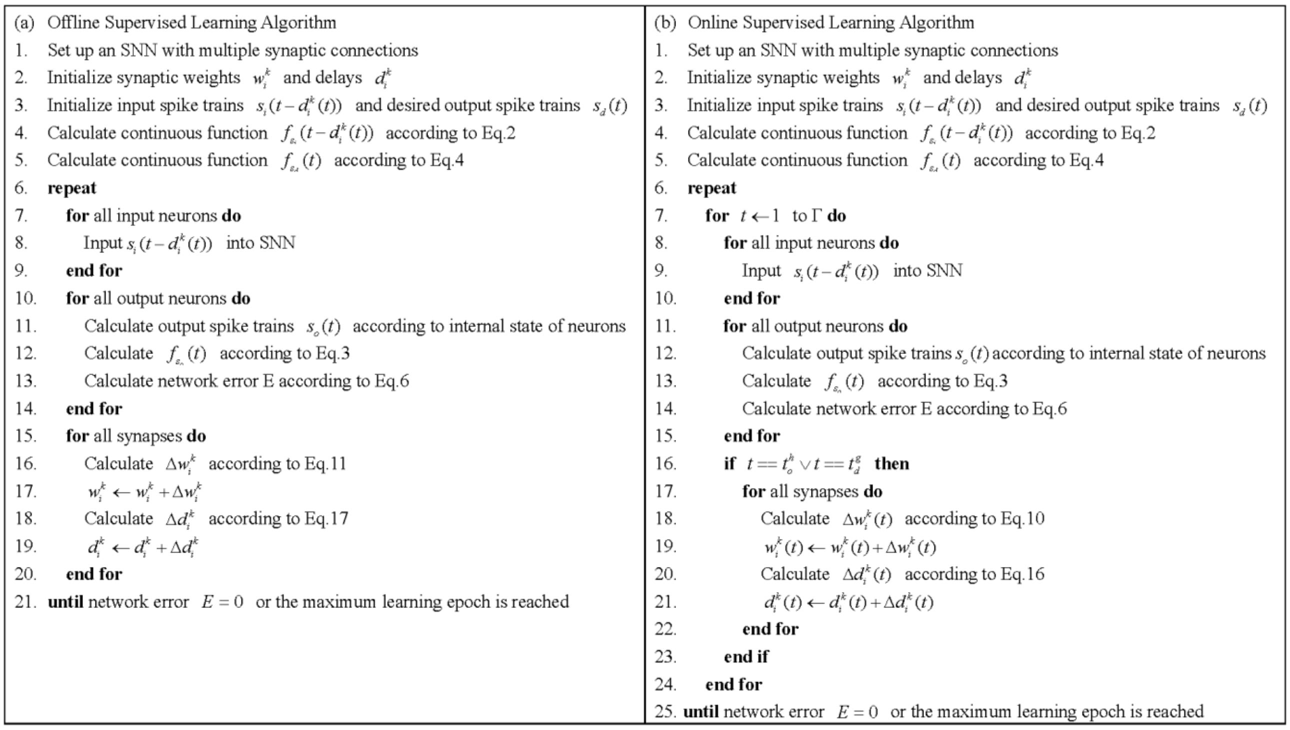
$$\Delta d_i^k(t)=\eta_d w_i^k(t)\Delta w_i^k(t) \tag{16}$$

Further integrating the online learning rule in Eq. 16, the offline learning rule for synaptic delays is obtained and can be represented as follows:

$$\Delta d_i^k=\int_\Gamma \Delta d_i^k(t)dt=\eta_d w_i^k\Delta w_i^k \tag{17}$$

2.4. Supervised learning using the proposed learning rules

In the supervised learning process of SNNs, the learnable parameters such as synaptic weights and delays of the network are adjusted to make the actual output spike train as similar as possible to the desired output spike train. Figure 1 shows the pseudocode for training SNNs using the proposed supervised learning rules, including offline learning and online learning.



```
(a) Offline Supervised Learning Algorithm
1.  Set up an SNN with multiple synaptic connections
2.  Initialize synaptic weights $w_i^k$ and delays $d_i^k$
3.  Initialize input spike trains $s_i(t-d_i^k(t))$ and desired output spike trains $s_d(t)$
4.  Calculate continuous function $f_{s_i}(t-d_i^k(t))$ according to Eq.2
5.  Calculate continuous function $f_{s_d}(t)$ according to Eq.4
6.  repeat
7.    for all input neurons do
8.      Input $s_i(t-d_i^k(t))$ into SNN
9.    end for
10.   for all output neurons do
11.     Calculate output spike trains $s_o(t)$ according to internal state of neurons
12.     Calculate $f_{s_o}(t)$ according to Eq.3
13.     Calculate network error E according to Eq.6
14.   end for
15.   for all synapses do
16.     Calculate $\Delta w_i^k$ according to Eq.11
17.     $w_i^k \leftarrow w_i^k + \Delta w_i^k$
18.     Calculate $\Delta d_i^k$ according to Eq.17
19.     $d_i^k \leftarrow d_i^k + \Delta d_i^k$
20.   end for
21. until network error $E = 0$ or the maximum learning epoch is reached
```

```
(b) Online Supervised Learning Algorithm
1.  Set up an SNN with multiple synaptic connections
2.  Initialize synaptic weights $w_i^k$ and delays $d_i^k$
3.  Initialize input spike trains $s_i(t-d_i^k(t))$ and desired output spike trains $s_d(t)$
4.  Calculate continuous function $f_{s_i}(t-d_i^k(t))$ according to Eq.2
5.  Calculate continuous function $f_{s_d}(t)$ according to Eq.4
6.  repeat
7.    for $t \leftarrow 1$ to $\Gamma$ do
8.      for all input neurons do
9.        Input $s_i(t-d_i^k(t))$ into SNN
10.     end for
11.     for all output neurons do
12.       Calculate output spike trains $s_o(t)$ according to internal state of neurons
13.       Calculate $f_{s_o}(t)$ according to Eq.3
14.       Calculate network error E according to Eq.6
15.     end for
16.     if $t == t_o^h \vee t == t_d^g$ then
17.       for all synapses do
18.         Calculate $\Delta w_i^k(t)$ according to Eq.10
19.         $w_i^k(t) \leftarrow w_i^k(t) + \Delta w_i^k(t)$
20.         Calculate $\Delta d_i^k(t)$ according to Eq.16
21.         $d_i^k(t) \leftarrow d_i^k(t) + \Delta d_i^k(t)$
22.       end for
23.     end if
24.   end for
25. until network error $E = 0$ or the maximum learning epoch is reached
```

Fig. 1 – Pseudocode of offline and online supervised learning using the proposed synaptic weight and delay update rules.

Firstly, initialize all parameters of the SNN, including the spiking neuron model and its parameters, input and desired output spike trains, synaptic weights and delays. Secondly, calculate the actual output spike trains of the network according to the input spike trains and the neuron model, and calculate the network error according to the actual and desired output spike trains. Finally, adjust the synaptic weights and delays of all synapses according to the proposed synaptic weight and delay update rules. Repeat this learning process until the network error is 0 or reaches the maximum learning epoch, then the training process is finished. The main difference between offline learning and online learning is that the offline learning algorithm updates the synaptic weights and delays only once during a learning epoch, while the online learning algorithm updates synaptic weights and delays several times.

3. RESULTS

In this section, the performance of the proposed algorithms is verified through spike train learning tasks as well as nonlinear pattern recognition tasks on the UCI dataset. To highlight the role of delays, the performance of SNNs with dynamic delays and static delays is compared. For the convenience of description, the proposed algorithms are named as follows: (1) Off-SD: Offline Static Delay, where the weights are adjusted in the offline learning mode while the delays are static (no longer changed). (2) Off-DD: Offline Dynamic Delay, where both weights and delays are adjusted in the offline learning mode. (3) On-SD: Online Static Delay, where the weights are adjusted in the online learning mode while the delays are static. (4) On-DD: Online Dynamic Delay, where both the weights and delays are adjusted in the online learning mode.

3.1. Spike train learning tasks

This subsection tests the performance of the proposed algorithms through spike train learning tasks. The main task of spike train learning is to iteratively train the SNN so that its actual output spike train is as similar as possible to the desired output spike train. This is a training process, not a testing process, so there is no overfitting issue. We constructed a single-layer SNN consisting of short-term SRM neurons, where the number of input and output neurons is $N_I = 200$ and $N_O = 1$, respectively. The number of synaptic connections between neurons is $N_S = 5$. A clock-driven simulation strategy is adopted, and the maximum learning epoch is 100. Both the input and desired output spikes are randomly generated within the time interval $\Gamma = 300$ ms by the homogeneous Poisson encoding method and set at the same spike firing rate $r = 20$ Hz. The weights and delays are randomly generated within the intervals [0, 1] and [0, 10] ms, respectively. The learning rates for weights and delays are set to $\eta_w = 0.01$ and $\eta_d = 5$, respectively, which are determined after several repeated tests under benchmark parameters. The performance of the proposed algorithms is evaluated by the learning accuracy and the number of learning epochs required to achieve the highest learning accuracy. The learning accuracy is measure by a correlation-based metric C [3], which is a commonly used metric in supervised learning of SNNs to measure the similarity between the actual and desired output spikes. The closer the value of C is to 1 the more accurate the learning is, and the closer the value of C is to 0 the less accurate the learning is. The number of learning epochs required to achieve the highest learning accuracy can reflect the convergence speed of an algorithm. The fewer required learning epochs means that the learning algorithm can achieve the highest learning accuracy in fewer learning epochs, which can reduce the maximum learning epoch in practical applications. The results of all spike train learning are the average obtained from 50 repeated tests.

In specific pattern recognition tasks, the number of input neurons N_I is usually related to the dimensionality of the sample data. The first spike train learning task tests the effect of N_I on the learning performance of the proposed algorithms, and the corresponding learning results are shown in Fig. 2. It can be seen from Fig. 2a that the learning accuracy C of the four algorithms increases gradually as N_I gradually increases. This is because more input neurons enable SNNs to extract more useful information from input spikes, resulting in higher learning accuracy. Moreover, the learning accuracy of On-DD is significantly higher than the other three algorithms when there are a large number of input neurons. It can be seen from Fig. 2b that the learning epochs required for the two online learning algorithms to achieve the highest learning accuracy are less than those of the two offline learning algorithms in most cases. The results of this spike train learning task indicate that the proposed algorithms, especially the On-DD algorithm, have potential advantages in spike pattern learning with different dimensions of sample data.

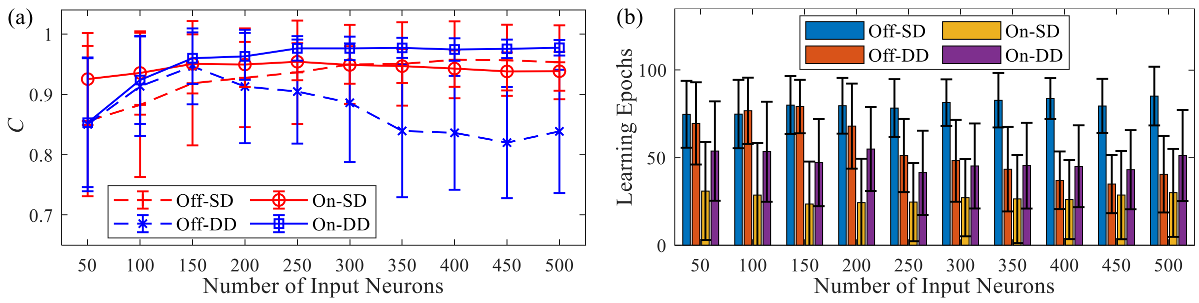


Fig. 2 – Learning results with different numbers of input neurons: (a) Learning accuracy *C*; (b) Learning epochs.

In biological nervous systems, the number of synaptic connections N_S between presynaptic and postsynaptic neurons is usually not fixed. The second spike train learning task tests the effect of N_S on the learning performance of the proposed algorithms, and the corresponding learning results are shown in Fig. 3. The learning accuracy *C* of the four algorithms is shown in Fig. 3a. It can be seen that these four algorithms can achieve higher learning accuracy in multi-synaptic connection scenarios than in single-synaptic connection scenarios. Moreover, when there are more synaptic connections between presynaptic and postsynaptic neurons, the learning accuracy of the On-DD algorithm is significantly higher than that of the other three algorithms. Figure 3b shows the learning epochs required for the four algorithms to achieve the highest learning accuracy. It can be seen that the learning epochs of the Off-SD algorithm are significantly higher than those of the other three algorithms, and the learning epochs of the two online learning algorithms are less than those of the two offline learning algorithms in most cases. The results of this spike train learning task suggest that introducing multiple synaptic connections can enhance the learning performance of SNNs.

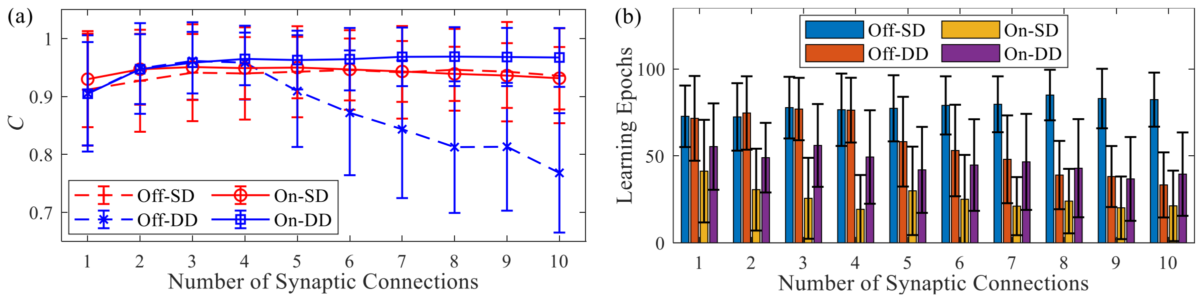


Fig. 3 – Learning results with different numbers of synaptic connections: (a) Learning accuracy *C*. (b) Learning epochs.

Spike firing rate *r* is an important indicator of the state of neuronal activity. The third spike train learning task tests the effect of *r* on the learning performance of the proposed algorithms, and the learning results are shown in Fig. 4. The learning accuracy *C* shown in Fig. 4a indicates that the learning accuracy of the two online learning algorithms is significantly higher than that of the two offline learning algorithms. The learning epochs shown in Fig. 4b indicate that as *r* gradually increases, the learning epochs of all the four algorithms are gradually decreasing. The results of this spike train learning task suggest that online learning algorithms have potential advantages in spike pattern learning with different spike firing rates.

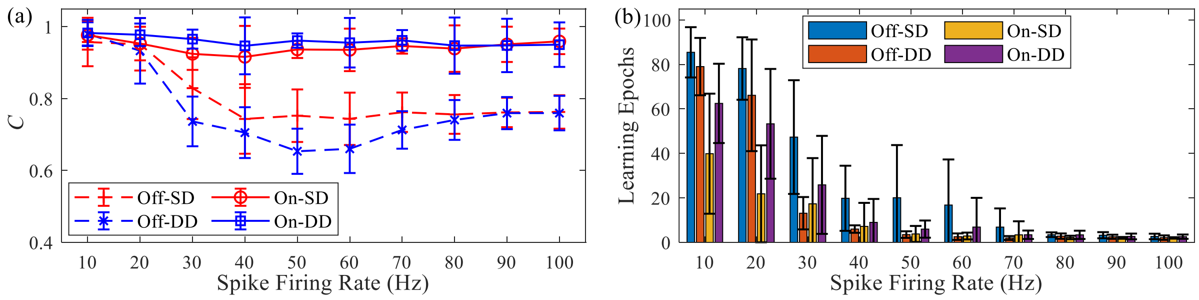


Fig. 4 – Learning results with different spike firing rates: (a) Learning accuracy *C*; (b) Learning epochs.

The length of spike trains Γ is an important indicator for testing the learning performance of SNNs in complex learning scenarios. The last spike train learning task tests the effect of Γ on the learning performance of the proposed algorithms, and the corresponding learning results are shown in Fig. 5. Figure 5a shows the learning accuracy C of the four algorithms. It can be seen that the learning accuracy of the two online learning algorithms is higher than that of the two offline learning algorithms. Moreover, the learning accuracy of the On-DD algorithm is still the highest. The learning epochs required for the four algorithms to achieve the highest learning accuracy are shown in Fig. 5b. It can be seen that as the length of spike trains gradually increases, the learning epochs of the On-SD algorithm do not change much, while the learning epochs of the other three algorithms decreases. In summary, the accuracy of the dynamic delay learning algorithms is higher than that of the static delay learning algorithms, and the accuracy of the online learning algorithms is higher than that of the offline learning algorithms.

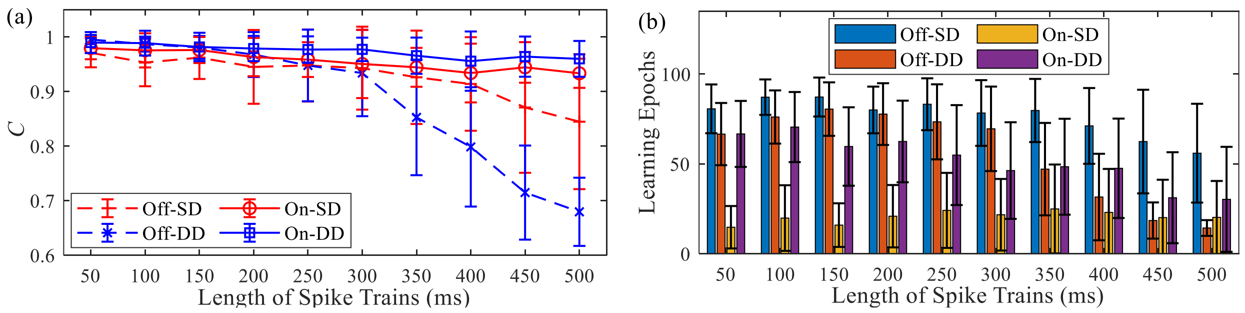


Fig. 5 – Learning results with different lengths of spike trains: (a) Learning accuracy C; (b) Learning epochs.

3.2. Nonlinear pattern classification

This subsection validates the ability of the proposed algorithms to solve nonlinear pattern recognition problems on two UCI datasets, namely the original Wisconsin breast cancer database (WBC) and the Pima Indians diabetes database (PIMA). The key to applying SNNs to solve nonlinear pattern classification problems is to use supervised learning algorithms to train an SNN classifier, which is further applied to predict the category of the sample. The dataset is divided into a training set and a testing set, where the training set is used to train the SNN classifier and the testing set is used to test the classification performance of the SNN classifier. The predicted label of a sample is determined based on the similarity between the actual output spike train $s_o(t)$ and the desired output spike train $s_d(t)$. If $s_o(t)$ corresponding to the sample is more similar to $s_d(t)$ corresponding to which class of sample, then it is considered to belong to that class. The classification accuracy is then calculated based on the predicted and actual labels of the samples. The classification process of applying the proposed algorithms to the WBC and PIMA datasets is shown in Fig. 6, which contains the following four steps: (1) *Data pre-processing*. Normalize all attribute values of each sample to the specified interval. (2) *Spike train encoding*. Using linear encoding to encode each normalized attribute value into a corresponding input spike train $s_i(t)$. (3) *Training*. The proposed algorithms are applied to adjust the weights and delays so that the $s_o(t)$ of each sample in the training set is as similar as possible to its $s_d(t)$ specified according to its label after several rounds of iterations. (4) *Testing*. The predicted labels of the samples in the testing set are determined based on the similarity between their corresponding $s_o(t)$ and the corresponding $s_d(t)$ for each class, and the classification accuracy is further calculated.

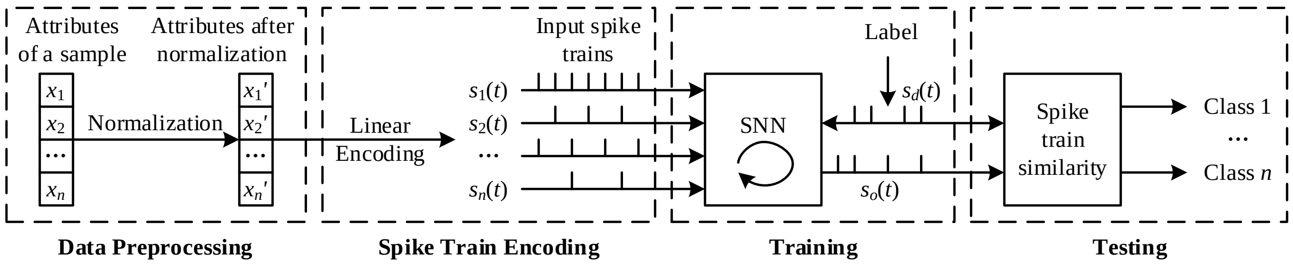


Fig. 6 – The process of applying the proposed algorithms to classify the UCI datasets.

For the WBC dataset, the 9 attribute values of each sample were first normalized to the interval [0, 1], and each attribute value was converted to a frequency value within [5, 20] Hz. Then the frequency value corresponding to each attribute value is used as the frequency for linear encoding to generate a spike train within [0, 50] ms. Finally, the 9 attribute values of each sample were encoded into 9 linearly encoded input spike trains. The desired output spike trains corresponding to the benign and malignant tumor samples are represented by linearly encoded spike trains with frequencies of 5 Hz and 10 Hz, respectively. For the PIMA dataset, the same encoding method is used to encode the 8 attribute values of each sample into 8 linearly encoded input spike trains. The desired output spike trains corresponding to the healthy and diabetes samples are represented by linearly encoded spike trains with frequencies of 5 Hz and 10 Hz, respectively. The frequency range corresponding to the attribute values of the samples during linear encoding and the frequency values for generating the desired output spike trains are determined after several repeated tests. 20 repeated tests were performed on each dataset, where the SNN was randomly initialized and the dataset was randomly divided into the training set containing 50% samples and the testing set containing the remaining 50% samples. The maximum number of iterations for each test was 50.

The comparison of the classification accuracy of the proposed algorithms with other supervised learning algorithms for SNNs on these two datasets is shown in Table 1. These algorithms are all supervised learning algorithms for single-layer SNNs, and their classification accuracy comes from the original paper. It can be seen that although the accuracy of the proposed algorithms is not the highest on the training sets of the two datasets, the proposed On-DD algorithm with multiple synaptic connections ($N_S = 5$) achieves the highest accuracy on the testing set of both datasets. These are not overfitting classification results. In addition, the proposed algorithms outperform the classification performance at multiple synaptic connections ($N_S = 5$) than at single synaptic connection ($N_S = 1$), and outperform the classification performance at dynamic synaptic delays than at static synaptic delays. These two tests demonstrate that the proposed algorithms are also effective in solving practical nonlinear pattern classification problems.

Table 1

Comparison of classification performance on the WBC and PIMA datasets

Methods	WBC dataset		PIMA dataset	
	Training Accuracy (%)	Testing Accuracy (%)	Training Accuracy (%)	Testing Accuracy (%)
SpikeTemp [11]	99.6	92.1	77.5	67.6
Online SRESN [12]	93.9 ± 1.8	94.0 ± 2.6	67.0 ± 0.8	66.1 ± 1.4
Batch SRESN [12]	97.7 ± 0.6	97.2 ± 0.7	70.5 ± 2.4	69.9 ± 2.1
FE-Learn [13]	94.8 ± 0.9	94.3 ± 1.7	79.3 ± 1.2	71.2 ± 2.0
MPD-AL [14]	99.9 ± 0.1	97.2 ± 0.6	71.4 ± 1.9	69.6 ± 1.3
Off-SD (N_S = 1)	92.3 ± 6.1	92.7 ± 4.8	66.1 ± 2.3	65.8 ± 2.7
Off-SD (N_S = 5)	97.0 ± 1.0	96.2 ± 0.9	67.6 ± 1.9	68.0 ± 2.0
Off-DD (N_S = 1)	95.8 ± 1.2	95.9 ± 0.9	66.8 ± 1.6	67.5 ± 1.9
Off-DD (N_S = 5)	96.7 ± 0.9	97.1 ± 0.7	69.9 ± 2.3	70.1 ± 3.0
On-SD (N_S = 1)	92.7 ± 5.4	92.3 ± 5.7	66.0 ± 3.1	65.7 ± 4.0
On-SD (N_S = 5)	96.7 ± 0.9	96.5 ± 1.0	69.3 ± 2.6	69.3 ± 2.8
On-DD (N_S = 1)	94.7 ± 1.7	96.7 ± 0.6	69.7 ± 2.6	69.7 ± 1.9
On-DD (N_S = 5)	96.6 ± 1.1	97.3 ± 0.7	70.1 ± 2.7	71.3 ± 1.5

4. DISCUSSION AND CONCLUSION

The synergistic plasticity of synaptic weights and delays is currently a hot topic in the research of supervised learning for SNNs. In this paper, we apply the kernel representation of spike trains to propose synaptic weight-delay synergistic supervised learning algorithms for SNNs with multiple synaptic connections. We will discuss the advantages and disadvantages of the proposed approach from the following three aspects. (1) *Comparison with other related algorithms* [5–8]. In terms of methodology, the proposed learning rules are derived using a gradient descent method at the spike train level represented by kernel functions, while the learning rules in [5–7] are constructed using the STDP mechanism, and the learning rule in [8] is derived using a threshold driven gradient descent method at the spike time level. In terms of network topology, the learning performance of the proposed algorithms is tested using SNNs with multiple synaptic connections, while the algorithms in [5–8] are all tested using SNNs with single synaptic connection. (2) *Advantages*. In terms of biological plausibility, the proposed approach comprehensively considers neuroscience mechanisms such as synaptic delay plasticity, multiple synaptic connections, and online learning, which is a biologically plausible learning approach. In terms of learning performance, the results of spike train learning and nonlinear pattern classification tasks indicate that introducing learnable dynamic synaptic delays and multiple synaptic connections can significantly improve the learning accuracy of SNNs and reduce the learning epochs, which effectively enhances the learning performance of SNNs. (3) *Disadvantages*. In terms of SNN applications, due to the simplicity of the network structure adopted in this paper, the proposed approach has not been applied to solve complex pattern recognition problems, such as image recognition and classification, time series data classification and prediction. In future research, we will extend the proposed approach to large-scale deep SNNs with complex topological structures and apply them to solve complex pattern recognition problems in the real world. In terms of computational complexity, due to the difficulty in analysing the computational complexity of supervised learning algorithms for SNNs and the limited available reference materials, we did not analyse the computational complexity of the proposed approach. However, it has been verified in [3] that the learning algorithm based on spike train kernels requires shorter program running time than the learning algorithm derived using gradient descent method at the spike time level, such as the learning rule in [8]. In future research, the computational complexity and convergence of such algorithms will be analysed theoretically, and parallel computing and GPUs will be used to accelerate the network simulation.

In this paper, we proposed synaptic weight-delay synergistic supervised learning algorithms based on spike train kernels for SNNs with multiple synaptic connections, including offline and online learning algorithms. The performance of the proposed algorithms is validated through spike train learning tasks and nonlinear pattern classification problems, and the following conclusions are drawn: (1) SNNs with multiple synaptic connections are able to process and manipulate information at a finer level and thus exhibit better learning performance than SNNs with a single synaptic connection. (2) Synaptic delay plasticity allows spiking neurons to integrate information from different sources more efficiently, so that SNNs with dynamic delays have better learning performance than SNNs with static delays. (3) Online learning models can continuously learn from input spikes and adapt to changes in the data distribution, so online learning algorithms for SNNs typically achieve higher learning accuracy in fewer learning epochs than offline learning algorithms.

ACKNOWLEDGEMENTS

This research was supported by the Youth Science and Technology Foundation of Gansu Province (23JRRA1577), the Natural Science Foundation of Gansu Province (24JRRA127) and the Scientific Research Ability Enhancement Program for Young Teachers of Northwest Normal University (NWNU-LKQN2024-21).

REFERENCES

[1] Wu J, Wang Y, Li Z, Lu L, Li Q. A review of computing with spiking neural networks. Computers, Materials & Continua. 2024; 78(3): 2909–2939.

[2] Rast A, Aoun MA, Elia EG, Crook N. Efficient learning in spiking neural networks. Neurocomputing. 2024; 597: 127962.

[3] Wang X, Lin X, Dang X. Supervised learning in spiking neural networks: a review of algorithms and evaluations. Neural Networks. 2020; 125: 258–80.

[4] Lobo JL, Del Ser J, Bifet A, Kasabov N. Spiking neural networks and online learning: an overview and perspectives. Neural Networks. 2020; 121: 88–100.

[5] Taherkhani A, Belatreche A, Li Y, Maguire LP. DL-ReSuMe: A delay learning-based remote supervised method for spiking neurons. IEEE Transactions on Neural Networks and Learning Systems. 2015; 26(12): 3137–3149.

[6] Guo L, Wang Z, Cabrerizo M, Adjouadi M. A cross-correlated delay shift supervised learning method for spiking neurons with application to interictal spike detection in epilepsy. International Journal of Neural Systems. 2017; 27(3): 1750002.

[7] Zhang M, Wu J, Belatreche A, Pan Z, Xie X, Chua Y, Li G, Qu H, Li H. Supervised learning in spiking neural networks with synaptic delay-weight plasticity. Neurocomputing. 2020; 409: 103–118.

[8] Yu Q, Gao J, Wei J, Li J, Tan KC, Huang T. Improving multispike learning with plastic synaptic delays. IEEE Transactions on Neural Networks and Learning Systems. 2023; 34(12): 10254–10265.

[9] Rabinowitch I, Colón-Ramos DA, Krieg M. Understanding neural circuit function through synaptic engineering. Nature Reviews Neuroscience. 2024; 25(2): 131−139.

[10] Lin X, Hu J, Zheng D, Hu T, Wang X. An online supervised learning algorithm based on feedback alignment for multilayer spiking neural networks. Proceedings of the Romanian Academy; Series A – Mathematics, Physics, Technical Sciences, Information Science. 2022; 23(2): 187−196.

[11] Wang J, Belatreche A, Maguire LP, McGinnity TM. SpikeTemp: an enhanced rank-order-based learning approach for spiking neural networks with adaptive structure. IEEE Transactions on Neural Networks and Learning Systems. 2017; 28(1): 30−43.

[12] Dora S, Subramanian K, Suresh S, Sundararajan N. Development of a self-regulating evolving spiking neural network for classification problem. Neurocomputing. 2016; 171: 1216−1229.

[13] Luo X, Qu H, Zhang Y, Chen Y. First error-based supervised learning algorithm for spiking neural networks. Frontiers in Neuroscience. 2019; 13: 559.

[14] Zhang M, Wu J, Chua Y, Luo X, Pan Z, Liu D, Li H. MPD-AL: an efficient membrane potential driven aggregate-label learning algorithm for spiking neurons. In: AAAI Conference on Artificial Intelligence, vol. 33. 2019, pp. 1327−1334.

Received October 14, 2024