Supervised Learning in Spiking Neural Networks with Synaptic Delay-Weight Plasticity

Malu Zhang[a,b], Jibin Wu[b], Ammar Belatreche[c], Zihan Pan[b], Xiurui Xie[d], Yansong Chua[d,*], Guoqi Li[e], Haizhou Li[b,d]

[a]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, China*
[b]*Department of Electrical and Computer Engineering, National University of Singapore, Singapore*
[c]*Department of Computer and Information Sciences, Faculty of Engineering and Environment, Northumbria University, U.K.*
[d]*Institute for Infocomm Research, A*STAR, Singapore*
[e]*Department of Precision Instrument, Beijing Innovation Center for Future Chip, Tsinghua University, Beijing 100084, P.R. China.*

Abstract

Spiking neurons encode information through their spiking temporal patterns. Although the precise spike-timing based encoding scheme has long been recognised, the exact mechanism that underlies the learning of such precise spike-timing in the brain remains an open question. Most of the existing learning methods for spiking neurons are based on synaptic weight adjustment. However, biological evidences suggest that synaptic delays can also be modulated to play an important role in the learning process. This paper investigates the viability of integrating synaptic delay plasticity into supervised learning and proposes a novel learning method that adjusts both the synaptic delays and weights of the learning neurons to make them fire precisely timed spikes, that is referred to as synaptic delay-weight plasticity. Remote Supervised Method (ReSuMe) and Perceptron Based Spiking Neuron Learning Rule (PBSNLR), two representative supervised learning methods, are studied to illustrate how the synaptic delay-weight plasticity works. The performance of the proposed learning method is thoroughly evaluated on synthetic data and is further demonstrated on real-world classification tasks. The experiments show that the synaptic delay-weight learning method outperforms the traditional synaptic weight learning methods

*Corresponding author: chuays@i2r.a-star.edu.sg

in many ways.

Keywords: spiking neurons, spiking neural networks, supervised learning, synaptic plasticity, synaptic weight, synaptic delay

1. Introduction

Spiking neural networks (SNNs) are biologically plausible models that are, unlike their traditional rate-based counterparts, capable of capturing the rich temporal dynamics of real biological neural assemblies and information representation and processing in the brain [1]. SNNs encode information in the temporal patterns of the transmitted spike trains. Despite their promising capabilities in achieving a performance similar to living brains, the computational efficiency and pattern classification potential have not been fully exploited. The research community has been looking to how to benefit from the biologically inspired computational models, such as SNN-based neuromorphic computing. The prominent programs include the SpiNNaker Project [2], Neurogrid [3], Spaun [4], TrueNorth Cognitive Computing [5, 6], SyNAPSE program [7], other neuromorphic circuits in [8], and Zeroth [9, 10].

Biological evidences suggest that the brain is able to perform supervised learning. The most documented findings for supervised learning in the central nervous system come from the studies on the cerebellum and the cerebellar cortex [11, 12]. For traditional rate-based neural networks, we have established a solid foundation for supervised learning algorithms and their applications, such as the perceptron learning rule [13] and the gradient backpropagation algorithm [14]. In SNNs, the computing unit is the spiking neuron whose fundamental computation is the transformation of incoming spike trains into precisely timed firing [15]. The rate-based learning methods cannot be directly applied to SNNs. To train the spiking neurons output precisely timed spikes, many supervised learning algorithms assume weight plasticity by adjusting only the weights. They can be categorized into the spike-driven methods and membrane potential-driven methods [16, 17] .

In spike-driven methods, the error between the target and actual output spikes are used to update the synaptic weights. SpikeProp [18] and the multi-spike learning algorithm [19] represent typical examples of the spike-driven methods. They construct the error functions directly between the desired and actual output spikes. The gradient descent of such errors is then used to adjust the synaptic connection weights. ReSuMe [20] is another spike-driven method which employs two weight update processes, namely strengthening synaptic weights using spike-timing dependent plasticity (STDP) and weakening them using anti-STDP. The precise-spike-driven (PSD) learning rule [21] adopts a learning mechanism similar to ReSuMe except that PSD applies different learning windows. The Chronotron E-learning [22] and the SPAN [23] learning rules are two other spike-driven learning methods, which try to reduce the distance between the actual and desired spike trains. The distance in Chronotron E-learning rule is defined by Victor and Purpura metric [24], and the SPAN applies a metric similar to the van Rossum metric [25]. While the spike-driven learning methods learn well in pattern classification tasks, their learning efficiency and accuracy remain much to be desired [26].

Membrane potential-driven methods were proposed in an attempt to improve the learning performance for spiking neurons. Some typical examples include the tempotron [27], PBSNLR [26], HTP [15], EMPD [17] and MemPo-Learn [28]. In contrast to the spike-driven methods, they take an entirely different approach where they use the postsynaptic membrane potential rather than spike times as the basis for synaptic updating. For instance, the tempotron [27] minimizes an error between the firing threshold and the maximum membrane potential. As we don't have the control over when the maximum membrane potential takes place, the binary tempotron learning rule is unable to output precisely timed spikes [29]. PBSNLR [26] and HTP [15] adapt the perceptron learning rule such that the membrane potential is driven towards the firing threshold at the desired spike times, and kept below the firing threshold otherwise [30]. Due to their linear property of perceptron learning, they cannot handle non-linearly separable problems. EMPD [17] and MemPo-Learn [28] implement the gradient descent dynamics that minimize two different error functions defined at the desired and undesired output times. Experiments suggest that both EMPD and Mempo-Learn outperform other learning methods due to their two error function strategy.

In addition, there are several gradient based learning algorithms for training deep spiking neural networks[31, 32, 33, 34, 35], and these methods can be grouped into two main categories. The first category trains a traditional artificial neural network (ANN) and transforms it into its SNN version where the rate of SNN neurons acts as the analog activity of ANN neurons [36, 37, 38]. The learning methods in the second category trains directly the SNN but differ in how they approximate the derivative of spike function. The SpikeProp backpropagates errors only at spike times and the derivative is calculated by a linear assumption of threshold crossing [18]. Recently, H. Mostafa has proposed a similar method without the assumption of linearization by using non-leaky integrate-and fire neuron[39]. The surrogate gradients methods are another kind of the second category, which provide an alternative approach to obtain the derivative of spike function with the typical examples of [40, 41, 35]

All of the above-mentioned learning algorithms are based solely on weight adjustment, that we call weight-based learning. They completely ignore the effect of synaptic delay. However, biological evidences suggest that the synaptic delay modulation can occur during the learning process, that greatly affects the learning performance [42, 43, 44, 45, 46]. A number of delay-based learning methods have been proposed, but they are far from perfect. For example, the delay selection methods [47, 48, 19] apply multiple sub-connections with various delays between two neurons, which increases the number of sub-connections, thus the training time as well. Furthermore, delay selection methods adopt constant synaptic delays that cannot be updated. The delay shift methods [49] are another type of synaptic delay-based learning. Unfortunately, they only update the synaptic delays to train a coincidence detector (CD), while keeping the synaptic weights constant.

The delay-learning remote supervised method (DL-ReSuMe) represents an attempt to integrate the delay shift approach and the weight adjustment method [50]. Unfortunately, DL-ReSuMe only allows the synaptic delays to be increased in a one-way adjustment, that is biologically counter-intuitive, and limits the ability to find the appropriate synaptic delays. In addition, the DL-ReSuMe is limited to specific neuron model of which the postsynaptic potential (PSP) is a exponential function. It remains an open question how to extend the DL-ReSuMe to other neuron models. Furthermore, the study of DL-ReSuMe has yet to show that delay-learning can be generalized beyond ReSuMe learning rule.

In this paper, we propose a synaptic delay learning method which can be incorporated with the synaptic weight supervised learning methods. ReSuMe and PBSNLR are selected as two typical examples to illustrate how such synaptic delay-weight plasticity works. We discuss how the synaptic weights and synaptic delays are adjusted to train the neuron to output the desired spikes, and at the same time, suppress the undesired ones. Experiments show that the ReSuMe and PBSNLR with synaptic delay-weight plasticity significantly outperforms the weight plasticity baseline in terms of efficiency and accuracy.

The remainder of this paper is organized as follows. Section 2 presents the spiking neuron model and the proposed synaptic delay plasticity. In section 3, ReSuMe and PBSNLR are selected as typical examples to illustrate how the joint synaptic delay-weight plasticity works. Section 4 describes the experiments conducted to evaluate the learning performance of the proposed delay-weight plasticity. The obtained results are discussed in section 5 and conclusions are drawn in section 6.

2. Neuron Model and Synaptic Delay Plasticity

Let's start by introducing the neuron model in our study. We will then formulate the learning of synaptic delays of the neuron model.

2.1. Integrate-and-Fire Neuron

A spiking neuron model is a mathematical description of the properties of certain cells in the nervous system that generate sharp electrical potentials across their cell membrane [1]. It is a major signaling unit of the spiking neural network. The current-based leaky integrate-and-fire neuron (LIF) model provides a faithful description of biological neurons [29]. It is also mathematically tractable. Hence, it has been widely used. To well connect our study with the prior work, we adopt the LIF model, without loss of generality.

We consider a LIF neuron with I inputs, and its postsynaptic membrane potential represented by $V_j(t)$ remains at the resting potential $V_{rest} = 0$ when it receives no spikes. When a spike produced at a pre-synaptic neuron i, a postsynaptic potential (PSP) is induced in the LIF neuron. By integrating the PSPs resulting from several incoming spikes, the LIF neuron fires a spike when its membrane potential $V(t)$ reaches the firing threshold ϑ. The dynamics of the neuron postsynaptic membrane $V(t)$ are governed by the following equation

$$V_j(t) = V_{rest} + \sum_{i}^{I} \omega_i \sum_{t_i^f + d_i < t} K(t - t_i^f - d_i) - \vartheta \sum_{t_j^s < t} \exp\big(-\frac{t - t_j^s}{\tau_m}\big) \quad (1)$$

where t_i^f is the fth spike of the presynaptic neuron i, and ω_i, d_i are the synaptic weight and synaptic delay, respectively. $\hat{i}$ denotes the normalized PSP kernel defined as

$$K(t - t_i^f - d_i) = V_0\big(\exp(\frac{-(t - t_i^f - d_i)}{\tau_m}) - \exp(\frac{-(t - t_i^f - d_i)}{\tau_s})\big) \quad (2)$$

The shape of PSPs is governed by the integration time constant of the postsynaptic membrane τ_m, and the decay time constant of synaptic currents τ_s. V_0 normalizes PSP so that the maximum value of the kernel is 1. The dynamics of the $K(t - t_i^f - d_i)$ are illustrated in Figure 1. The PSP generated by t_i^f will contribute maximum membrane potential at $t_i^f + d_i + \psi$, where $\psi = \tau_m \tau_s (\ln \tau_m - \ln \tau_s)/(\tau_m - \tau_s)$ [29]. The last term in Equation (1) is the refractoriness function, where t_j^s are the times of the output spikes emitted by the learning neuron j. In the following experiments, the parameters of the spiking neuron model are set as follows: $\tau_m = 5$ ms,$\tau_s = 1.25$ ms, and $\vartheta = 1$ mV.

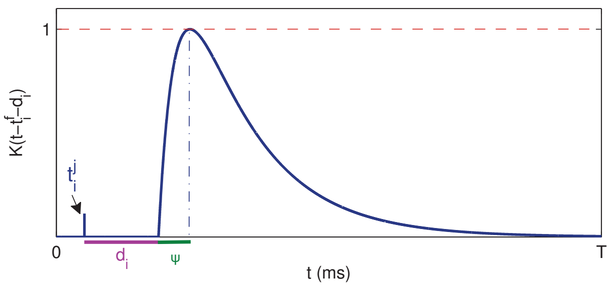

Figure 1: The dynamics of the PSP kernel $K(t - t_i^f - d_i)$. t_i^f is the fth spike of the presynaptic neuron i, d_i is the synaptic delay, and $d_i + \psi$ represents the delay by which the input spike t_i^f contributes the maximum PSP to the postsynaptic neuron j.

2.2. The Proposed Synaptic Delay Plasticity

In supervised learning, a neuron is trained to fire spikes at the desired times in response to a given class of inputs. The neurons should keep silent otherwise. ReSuMe and PBSNLR adjust the synaptic weights in a learning process to achieve this. These weight-based learning methods regard the synaptic delay as constant, in other words, they disregard the synaptic delay plasticity, and its possible active role in the learning process.

In this paper, we propose a novel learning rule for the synaptic delay plasticity. It enables a neuron to fire spikes at the desired times, and to keep silent otherwise. Next we study the properties of the learning rule in the two scenarios.

2.2.1. When a spike is supposed to fire

When a spike is supposed to fire at certain time, but the membrane potential is below the firing threshold, the membrane potential of the post-synaptic neuron should be increased. We propose a delay learning rule that applies a synaptic delay shift to the synapses in a way that we increase the membrane potential at the desired time. The delay learning rule is applied in 3 steps.

Step 1: Calculate the distance D_i^f of each input spike t_i^f, where D_i^f measures the distance between the current desired output time t and the time when input spike t_i^f has a maximum PSP. Specifically, D_i^f is calculated as

$$D_i^f = |t_i^f + d_i + \psi - t| \qquad \text{if} \quad t \geq t_i^f + \psi \tag{3}$$

where $t_i^f + d_i + \psi$ is the time when the input spike t_i^f contributes the maximum membrane potential (see Figure 1), and t is the current desired output time.

Step 2: Find the synapse $\hat{i}$ that need to be adjusted. The synapse $\hat{i}$ is subject to the following conditions: 1) synapse $\hat{i}$ is excitatory ($\omega_{\hat{i}} > 0$, where $\omega_{\hat{i}}$ is the weight of synapse $\hat{i}$.) 2) the delay of synapse $\hat{i}$ has not been adjusted previously, and 3) the synapse $\hat{i}$ has the spike $t_{\hat{i}}^{\hat{f}}$ whose $D_{\hat{i}}^{\hat{f}}$ is the smallest[1] among all D_i^f.

Step 3: Update the delay of synapse $\hat{i}$ as

$$d'_{\hat{i}} = \begin{cases} d_{\hat{i}} + D_{\hat{i}}^{\hat{f}} & \text{if} \quad t > t_{\hat{i}}^{\hat{f}} + d_{\hat{i}} + \psi, \omega_{\hat{i}} > 0 \\ d_{\hat{i}} - D_{\hat{i}}^{\hat{f}} & \text{if} \quad t < t_{\hat{i}}^{\hat{f}} + d_{\hat{i}} + \psi, \omega_{\hat{i}} > 0 \end{cases} \tag{4}$$

It is noted that the delay updating rule in Equation (4) adjusts the synaptic delay to increase the membrane potential. As shown in Figure 2A, if $t > t_{\hat{i}}^{\hat{f}} + d_{\hat{i}} + \psi$, the synaptic delay should be increased by $D_{\hat{i}}^{\hat{f}}$. On the other hand, as shown in Figure 2B, if $t < t_{\hat{i}}^{\hat{f}} + d_{\hat{i}} + \psi$, the synaptic delay should be decreased by $D_{\hat{i}}^{\hat{f}}$. By shifting the delays, the maximum PSP generated by $t_{\hat{i}}^{\hat{f}}$ is pushed towards the desired output time t,thus increasing the membrane potential and the likelihood of firing at the desired time.

2.2.2. When no spikes are supposed to fire

Other than the desired firing times, the membrane potential should remain below the firing threshold. In this case, we propose a delay learning rule that adjusts the synaptic delays to reduce the membrane potential. The delay shift process is shown in Equation (5) which is the same as Equation (4) except that

[1]Synaptic delay updating inevitably changes not only the membrane potential at the current time but also the membrane potential elsewhere. This leads to learning interference. To weaken this interference as much as possible, we select the smallest D_i^f, which will result in a minimal interference.

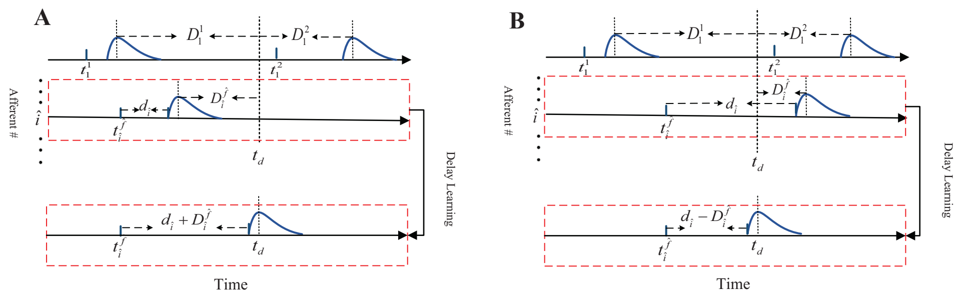

Figure 2: Illustration of the proposed delay learning at desired spike time. **(A)** The synaptic should delayed to increase the membrane potential; (B) The synaptic delay should be decreased to increase the membrane potential.

the synapse $\hat{i}$ is selected from inhibitory synapses.

$$d'_{\hat{i}} = \begin{cases} d_{\hat{i}} + D_{\hat{i}}^{\hat{f}} & \text{if} \quad t > t_{\hat{i}}^{\hat{f}} + d_{\hat{i}} + \psi, \omega_{\hat{i}} < 0 \\ d_{\hat{i}} - D_{\hat{i}}^{\hat{f}} & \text{if} \quad t < t_{\hat{i}}^{\hat{f}} + d_{\hat{i}} + \psi, \omega_{\hat{i}} < 0 \end{cases} \tag{5}$$

After updating the synaptic delays, the input spike $t_{\hat{i}}^{\hat{f}}$ has the maximum value of PSP at the undesired output time. As this is an inhibitory synapse with a negative weight $\omega_{\hat{i}} < 0$, the weighted PSP reduces the membrane potential maximum, and this reduction should drive the membrane potential below the firing threshold.

3. Joint Synaptic Delay-Weight Plasticity

3.1. ReSuMe with Joint Synaptic Delay-Weight Plasticity (ReSuMe-DW)

We now formulate the joint synaptic delay-weight plasticity by extending the ReSuMe and PBSNLR frameworks.

3.1.1. ReSuMe Learning Rule

ReSuMe is a spike-driven supervised learning method for SNNs. It aims to minimize the error between the desired and the actual output spikes by updating the synaptic weights according to the following equation:

$$\frac{d}{dt}w_{io}(t) = [S_d(t) - S_o(t)][a_d + \int_0^{\infty} W_{in}(s)S_i(t-s)ds], \tag{6}$$

where w_{io} is the synaptic weight between the pre-synaptic neuron i and the postsynaptic synaptic neuron o. $S_d(t)$, $S_o(t)$ and $S_i(t)$ are the target, post-, and presynaptic spike trains, respectively. The spike trains take the following form,

$$S(t) = \sum_f \delta(t - t^f), \tag{7}$$

where $f = 1, 2, ...$ is the index of the spikes and $\delta()$ is a Dirac function with $\delta(t) = 1$ for $t = 0$ (or 0 otherwise). The kernel $W_{in}(s)$ defines the shape of a learning window,

$$W_{in}(s) = A \cdot exp(\frac{-s}{\tau}) \tag{8}$$

190 where A is the maximal magnitude of the learning window and τ denotes the time constant of the learning process. The sign of the error signal $(S_d(t) - S_o(t))$ decides the direction of the synaptic adjustment. The kernel $a_d + \int_0^\infty W_{in}(s) S_i(t-s)ds$ decides the amount of the weight change. We illustrate the ReSuMe rule in Figure 3

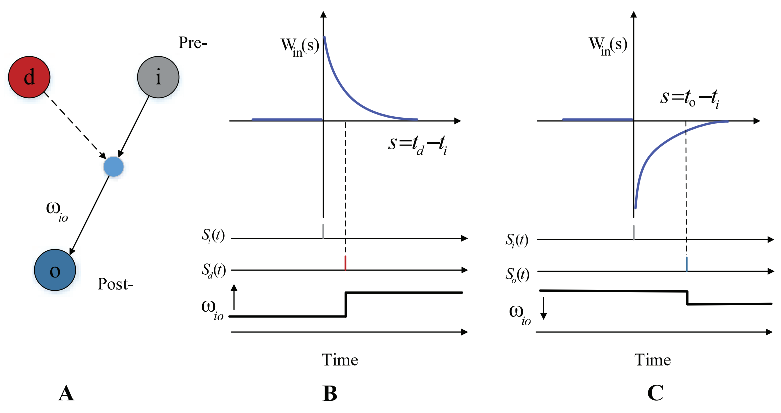

Figure 3: Illustration of the ReSuMe learning rule. **(A)** The synaptic efficacy ω_{io} between any presynaptic neuron i and any postsynaptic neuron o, depends on the correlation between the pre- and postsynaptic firing times and on the correlation between pre- and desired firing times (a 'remote' teacher neuron d). **(B)** The synaptic efficacy is potentiated whenever a desired spike is observed. The amplitude of change is determined by the learning windows $W_{in}(s)$. **(C)** The synaptic efficacy is depressed whenever the trained neuron fires. This figure is revised from [20, 51].

3.1.2. ReSuMe-DW Learning Rule

Applying the synaptic delay learning rule that we discuss in Section II to ReSuMe, we propose a joint synaptic delay-weight learning rule, ReSuMe-DW. During the learning process, both the synaptic delays and weights are trained. The synaptic delays are trained according to Equation (4) and Equation (5), and the weights are updated by the modified delay version of ReSuMe rule as

$$\frac{d}{dt}w_{io}(t) = [S_d(t) - S_o(t)][a_d + \int_0^{\infty} W_{in}(s)S_i(t-s-d_i)ds], \tag{9}$$

where d_i is the time delay of synapse i.

The detailed pseudocode of the proposed ReSuMe-DW learning algorithm is shown in Algorithm 1.

3.2. PBSNLR with Joint Synaptic Delay Plasticity (PBSNLR-DW)

3.2.1. PBSNLR Learning Rule

The well-known perceptron learning rule (PLR) performs input-output mapping in a supervised manner [52]. Recent studies on Perceptron, such as the PBSNLR and the HTP, suggest that the theoretical framework of PLR is well suitable for supervised learning of spiking neural network.

For PLR to work for spiking neurons, the expression of the dynamics of spiking neurons (Equation (1)) can be re-written in the Perceptron form as follow:

$$V(t) = \boldsymbol{\omega}\mathbf{P}^t + b \tag{10}$$

where $\boldsymbol{\omega}^T = \{\omega_1, \omega_2, ..., \omega_n\}$ is the synaptic weight vector. $\mathbf{P}^t = \{P_1^t, P_2^t, ..., P_n^t\}$, where P_i^t is the sum of PSPs induced by all spikes emitted from synapse i at time t. The P_i^t is computed as,

$$P_i^t = \sum_{t_i^f + d_i < t} K(t - t_i^f) \tag{11}$$

In addition, the basis value of b in Equation (10) is defined as

$$b = -\vartheta \sum_{t_j^s < t} \exp\big(-\frac{t - t_j^s}{\tau_m}\big) + V_{rest} \tag{12}$$

```
Algorithm 1: The Learning Algorithm of ReSuMe-DW
Definition:
$V(t)$: The membrane potential of the learning neuron.
$\boldsymbol{t_d}$: The set of target output spikes $\{t_d^1, t_d^2,..., t_d^N\}$ of the postsynaptic
  neuron.
$\boldsymbol{t_o}$: The set of actual output spikes $\{t_o^1, t_o^2,..., t_o^N\}$ of the postsynaptic
  neuron.
$\vartheta$: Firing threshold of the spiking neurons.
Initialization:
The weight matrix $\boldsymbol{\omega}$=$\{\omega_1, \omega_2, ...\}$ is initialized randomly
The delay matrix $\mathbf{D}$=$\{d_1, d_2, ...\}$ is initialized randomly
Training Epoch:
  For each desired output time $t_d^h$
    Step 1: update synaptic weights by:
 $\triangle\omega_{io} = a_d + \int_0^\infty W_{in}(s)S_i(t-s-d_i)ds$
    Step 2: If $V(t) < \vartheta$, then update synaptic delays according to
 Equation (4);
  EndFor
  For each actual output time $t_o^h$
    Step 1: update synaptic weights as:
 $\triangle\omega_{io} = -a_d - \int_0^\infty W_{in}(s)S_i(t-s-d_i)ds$;
    Step 2: If $t_o^h \notin t_d$, then update synaptic delays according to
 Equation (5);
  EndFor
Testing:
Test the trained neuron with the new synaptic delays and weights. If
 the trained neuron doesn't produce the expected precisely-timed spike
 train, we repeat the training cycle with one more epoch until it does.
 The training will stop upon completing the maximum learning epochs.
```

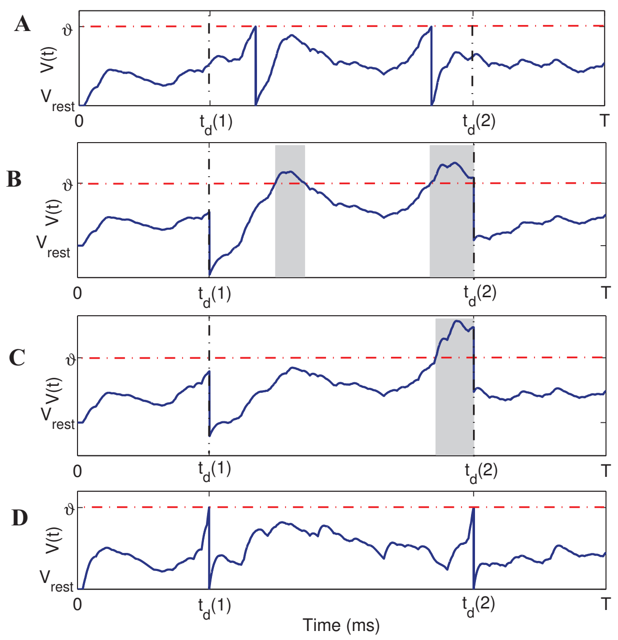

Figure 4: Illustration of the synaptic weight learning rule in PBSNLR. **(A)** Membrane potential trace of the neuron before learning. $t_d(1)$ and $t_d(2)$ are two desired output spikes. **(B)** During the learning process, the neuron does not spike at threshold crossing and membrane potential resets are forced at desired spike times. When the membrane potential is above threshold at undesired times (gray shaded areas), synaptic weights will be depressed until the membrane potential is driven below the firing threshold. When the membrane potential is below the threshold at desired output times, the synaptic weights will be increased to make a threshold crossing. **(C)** The voltage trace of the neuron after several learning epochs. **(D)** The membrane potential trace after a successful learning.

where t_j^s is the desired output spike time.

The PBSNLR [26] performs the supervised learning as a binary classification problem where the perceptron learning rule is used to classify the actual spikes into one of the two classes, that is positive when the spikes are expected, and negative otherwise. The misclassification occurs when (1) the membrane potential is below the firing threshold at the desired times and (2) it reaches or exceeds the firing threshold when it shouldn't. The synaptic weights are updated in the supervised learning as follows:

$$\boldsymbol{\omega}^{'} = \begin{cases} \boldsymbol{\omega} + \beta \mathbf{P}^t, & \text{if} \quad D_e(t) = 1 \quad \text{and} \quad V(t) < \vartheta \\ \boldsymbol{\omega} - \beta \mathbf{P}^t, & \text{if} \quad D_e(t) = 0 \quad \text{and} \quad V(t) \geq \vartheta \\ \boldsymbol{\omega}, & \text{otherwise} \end{cases} \tag{13}$$

where $D_e(t) = 1$ means t is the desired output time, and $D_e(t) = 1$ means otherwise. The working principle of PBSNLR learning rule is illustrated in Figure 4.

3.2.2. PBSNLR-DW Learning Rule

When a spike is supposed to fire at certain time, but the membrane potential is below the firing threshold, the membrane potential of the post-synaptic neuron should be increased. In PBSNLR-DW, we adjust both the synaptic delays and weights for the spikes to fire at the desired times.

The training samples P_i^t are computed as follows whenever a synaptic delay is updated,

$$P_i^t = \sum_{t_i^f + d_i < t} K(t - t_i^f - d_i) \tag{14}$$

We summarize the algorithm in Algorithm 2.

4. Experiments

Now let's examine the performance of the proposed joint synaptic delay-weight plasticity through experiments. In subsection 4.1, we will investigate the learning capabilities of ReSuMe-DW and PBSNLR-DW. In subsection 4.2 to subsection 4.4, we will investigate the effect of different factors on the learning performance. The factors include the total time duration (T_t), the number of the synaptic inputs (N_s), the input spike frequency (F_{in}) and the output spike frequency (F_o) of the spike trains. In subsection 4.5, we will look into the robustness of ReSuMe-DW and PBSNLR-DW. We will further study the proposed synaptic delay-weight plasticity for the learning of a real-world pattern classification task.

```
Algorithm 2: The Learning Algorithm of PBSNLR-DW
  Definition:
  $\mathbf{P}^t$: $\mathbf{P}^t = \{P_1^t, P_2^t, ..., P_n^t\}$, where $P_i^t$ is the sum of PSPs induced by all
    spikes that emitted from synapse $i$ at time $t$.
  $D_e(t)$: $D_e(t) = 1$ (or $D_e(t) = 0$ ) means $t$ is the desired (or undesired)
    output time.
  Initialization:
  The weight matrix $\boldsymbol{\omega}$=$\{\omega_1, \omega_2, ...\}$ is initialized randomly.
  The delay matrix $\mathbf{D}$=$\{d_1, d_2, ...\}$ is initialized randomly.
  Training Epoch:
    step 1: Constitute training samples $\{\mathbf{P}^1$,$d^1\}$, $\{\mathbf{P}^2$,$d^2\}$, ...
    step 2: Update the synaptic weights and synaptic delays
        If $D_e(t) = 1$ and $\boldsymbol{\omega}\mathbf{P}^t + b < \vartheta$
            $\boldsymbol{\omega}^{'} = \boldsymbol{\omega} + \beta\mathbf{P}^t$;
            Update delay of synapse $\hat{i}$ according to Equation (4);
        EndIf
        If $D_e(t) = 0$ and $\boldsymbol{\omega}\mathbf{P}^t + b \geq \vartheta$
            $\boldsymbol{\omega}^{'} = \boldsymbol{\omega} - \beta\mathbf{P}^t$;
            Update delay of synapse $\hat{i}$ according to Equation (5);
        EndIf
    step 3: Update the training samples $P_i^t$ according to Equation (14),
  return to step 2.
  Testing:
  Repeat the above steps until all training samples are correctly
    classified. Then, test the trained neuron with the new synaptic delays
    and weights.
```

4.1. Learning Sequence of Spikes

In this section, we present the experiments to demonstrate that spiking neurons trained according to ReSuMe-DW and PBSNLR-DW are capable of learning and precisely reproducing desired sequences of spikes. A spiking neuron with 400 synaptic inputs is trained to emit a precisely timed spike sequence. The length of input and desired output spike trains is 400 ms. The mean frequency of the input spike trains and the desired output spike train are set to F_{in}=2 Hz and F_o=100 Hz, respectively. The initial synaptic weights are selected as the uniform distribution in the interval [0, 0.01]. The initial synaptic delays are selected from the uniform distribution in the interval [0, 5]ms. We record the learning process of both ReSuMe-DW and PBSNLR-DW in Figure 5 and Figure 6, respectively .

From Figure 5C, we note that the actual spikes are very different from the desired output at the beginning. After several learning epochs, the difference is reduced gradually. At about 25 epochs the actual output spike train becomes the same as the desired one. We use learning accuracy C [53] to quantitatively evaluate the learning performance (C is assumed 0 for uncorrelated spike trains and 1 for perfectly matched firing patterns). The learning accuracy C plotted as a function of learning epoch is shown in Figure 5D, which indicates that the initial value of C is close to 0.2, and C increases to 1 after about 25 learning epochs.

Figure 6 summarizes the learning process of PBSNLR-DW. At the beginning, the trained neuron is observed to fire at arbitrary time, resulting in a small C value. During the learning process, the neuron gradually learns to produce spikes at the desired time, as evidenced by the increasing C. The learning accuracy increases sharply after 10 epochs, and saturates after 20 epochs.

4.2. Effect of the Total Time Duration of Spike Trains (T_t)

In the supervised learning algorithms that we discussed, a neuron learns from a target spike train, that can be considered as an episode of a continuous spike sequence. In this experiment, we would like to examine the effect of the

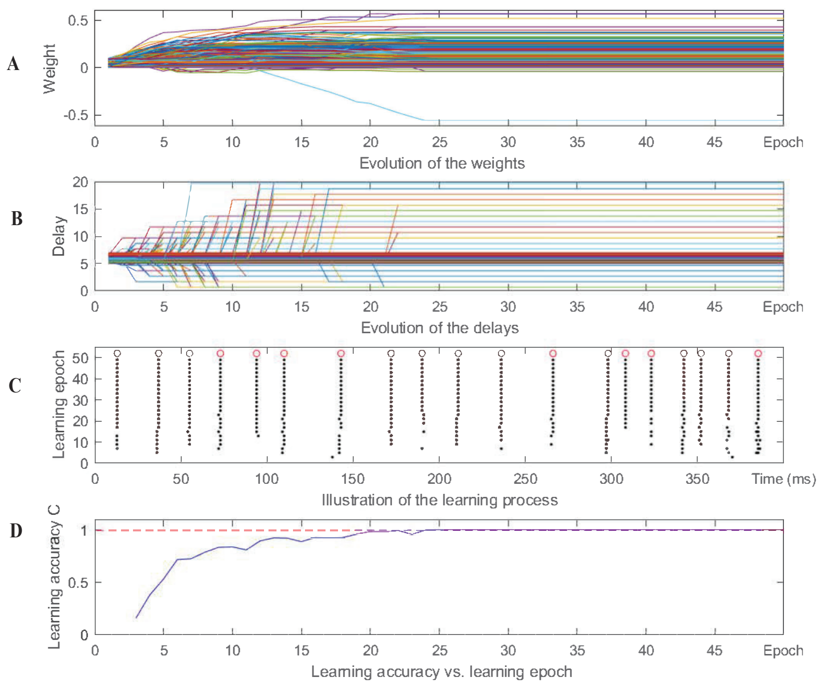

Figure 5: Illustration of ReSuMe-DW learning process of a neuron with 400 synapses over 50 epochs. Each colored line in panels **(A)** and **(B)** represents one synapse. **(A)** The evolution of the synaptic weights. **(B)** The evolution of the synaptic delays. **(C)** The actual output spike trains, denoted by • vs the target denoted by red ◦, during the learning process. **(D)** The learning accuracy [53] as a function of the number of learning epochs.

duration of such a target spike train on the learning process. We continue to test on a neuron of 400 synaptic inputs. The neuron is trained to produce target spike trains of different lengths.

Every input spike train and the target output spike train are generated according to a homogeneous Poisson process with firing rates $F_{in} = 2$ Hz and $F_o = 100$ Hz, respectively. The ratio between inhibitory and excitatory synapses is set to 1/4, and the weights of excitatory synapses are initialized to 0.05 and the weights of inhibitory synapses are initialized to -0.05. The length of the target spike trains varies from 100 ms to 1000 ms with an interval of 100 ms. For each T_t value, 20 experiments are carried out for different input and desired output spike trains. The average maximum value of C measure, the average number of

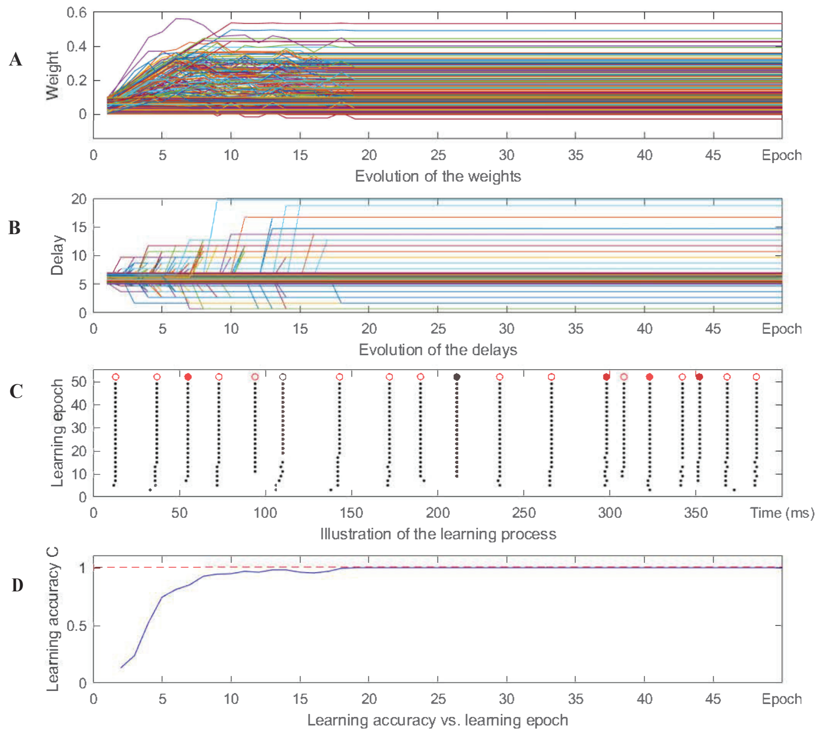

Figure 6: Illustration of PBSNLR-DW learning process of a neuron with 400 synapses over 50 epochs. Each colored line in panels **(A)** and **B** represents one synapse. **(A)** The evolution of the synaptic weights. **(B)** The evolution of the synaptic delays. **(C)** The actual output spike trains, denoted by • vs the target denoted by red ◦, during the learning process. **(D)** The learning accuracy [49] as a function of the number of learning epochs.

epochs and the average computing time that are needed to reach the maximum of C are reported for comparison. The experimental results of ReSuMe-DW and PBSNLR-DW are shown in Figure 7 and Figure 8, respectively.

Figure 7A and Figure 8A show the learning accuracy as a function of the durations of the spike trains. There is a trend that the learning accuracies of these four methods drop as the length increases gradually. We are glad to see that both ReSuMe-DW and PBSNLR-DW outperform the ReSuMe and PBSNLR baseline consistently. For example, when $T_t = 1,000$ ms, the learning accuracy of PBSNLR-DW is almost 1, while the learning accuracy of PBSNLR is about 0.94. In addition, the standard deviations of the mean accuracies of

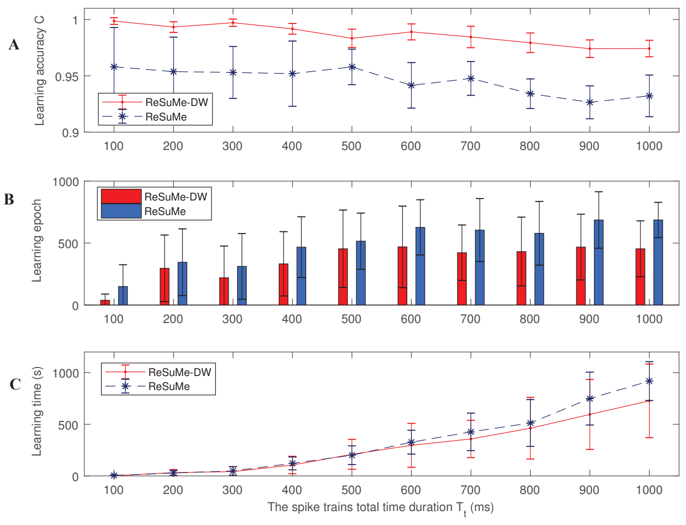

Figure 7: A comparative study between ReSuME-DW and ReSuMe on their learning performance as a function of the length of the target spike trains.**(A)** Learning accuracies of ReSuMe-DW and ReSuMe. **(B)** Required learning epochs of ReSuMe-DW and ReSuMe. **(C)** Required learning time of ReSuMe-DW and ReSuMe.

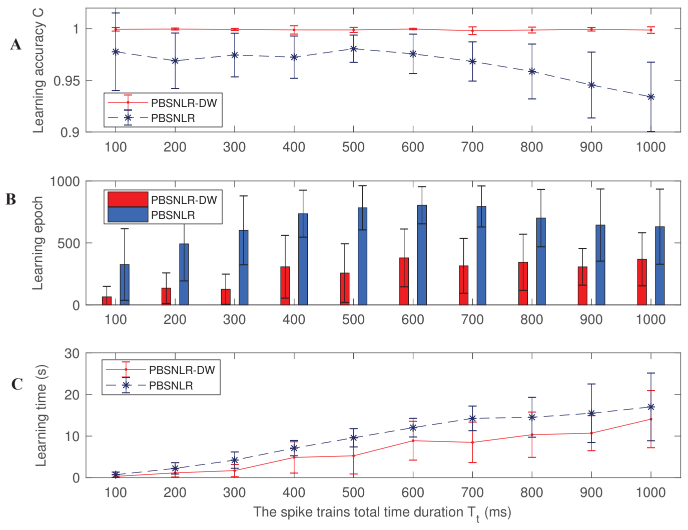

Figure 8: A comparative study between PBSNLR-DW and PBSNLR on their learning performance as a function of the length of the target spike trains. **(A)** Learning accuracies of PBSNLR-DW and PBSNLR. **(B)** Required learning epochs of PBSNLR-DW and PBSNLR. **(C)** Required learning time of PBSNLR-DW and PBSNLR.

ReSuMe-DW and PBSNLR-DW are lower than those of ReSuMe and PBSNLR, that suggest the joint synaptic delay-weight plasticity leads to a more stable and robust neuron. Figure 7B and Figure 8B show that the required learning epochs for ReSuMe-DW and PBSNLR-DW to reach the maximum accuracy C are less than that of ReSuMe and PBSNLR. For instance, when the length of the spike train is 1,000, PBSNLR completes the training with about 630 learning epochs while PBSNLR-DW just requires about 370 learning epochs. Figure 7C shows that the required learning time of ReSuMe and ReSuMe-DW is comparable. and ReSuMe-DW has a little advantage. However, as shown in Figure 8C, the learning time of PBSNLR-DW is clear shorter than that of PBSNLR, which means that PBSNLR-DW outperforms PBSNLR in terms of learning efficiency.

4.3. Effect of the Number of the Synaptic Input (N_s)

As discussed in Section II, a neuron model is also defined by the number of synaptic inputs, that has a direct impact on the size and efficiency of a spiking neural network. In this experiment, we investigate the effect of the number of the synaptic inputs N_s. Every input spike train and the desired output spike train are generated according to a homogeneous Poisson process with rates $F_{in} = 2$ Hz and $F_o = 100$ Hz, respectively. The length of the input and desired output spike train is 400 ms. N_s varies from 50 to 500 with an interval of 50. The experimental results are shown in Figure 9 and Figure 10.

We note that both ReSuMe-DW and PBSNLR-DW achieve a higher learning accuracy with a lower number of synaptic connections than their weight-based counterparts. As shown in Figure 9, when $N_s = 250$, ReSuMe-DW reaches a learning accuracy of 1 while ReSuMe only reaches around 0.9. In terms of learning efficiency, the learning epochs and learning time required by ReSuMe-DW and PBSNLR-DW are less than ReSuMe and PBSNLR in most cases. For example, when the number of synaptic inputs is 300, the training time of PBSNLR-DW and PBSNLR measures 6 and 10 seconds respectively.

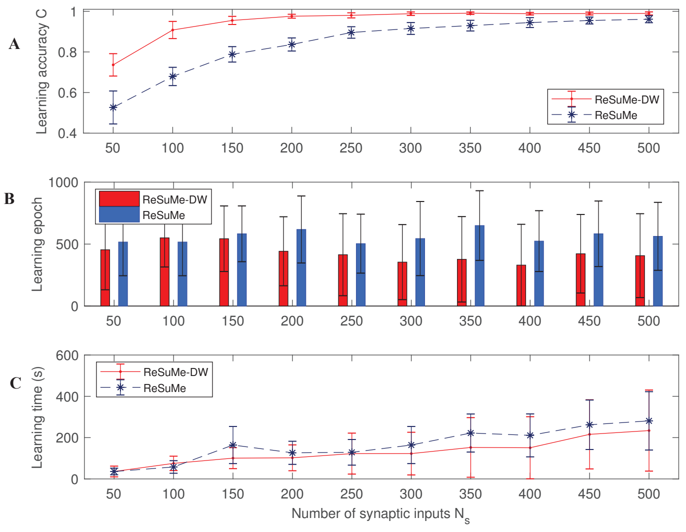

Figure 9: A comparative study between ReSuME-DW and ReSuMe on their learning performance as a function of the number of synaptic inputs. **(A)** Learning accuracies of ReSuMe-DW and ReSuMe. **(B)** Required learning epochs of ReSuMe-DW and ReSuMe. **(C)** Required learning time of ReSuMe-DW and ReSuMe.

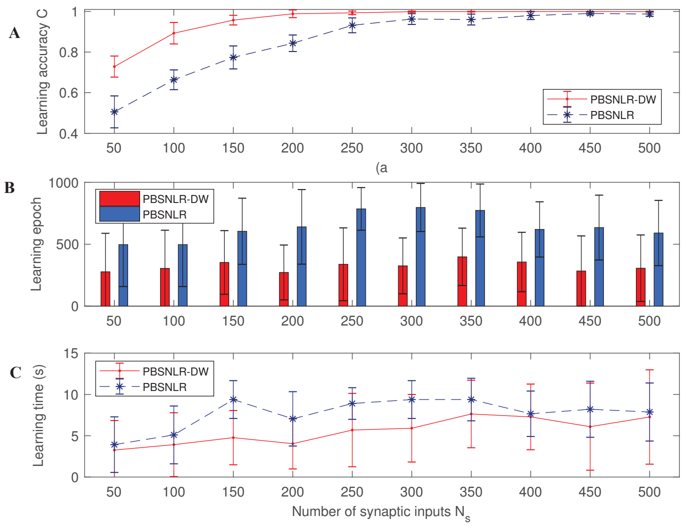

Figure 10: A comparative study between PBSNLR-DW and PBSNLR on their learning performance as a function of the number of the synaptic inputs. **(A)** Learning accuracies of PBSNLR-DW and PBSNLR. **(B)** Required learning epochs of PBSNLR-DW and PBSNLR. **(C)** Required learning time of PBSNLR-DW and PBSNLR.

4.4. Effect of Spike Train Frequency

Both the input spike frequency and the target spike frequency has a great impact on the learning performance as well [17] [26] [28]. For instance, a higher target spike frequency means that more target spikes are desired, which certainly increases the learning difficulty. In this section, experiments are conducted to investigate the effect of the spike frequency on different learning algorithms. The firing rates of the input spike trains F_{in} vary from 1 Hz to 10 Hz with an interval of 1 Hz. The firing rates of the target spike trains F_{out} vary from 10 Hz to 100 Hz with an interval of 10 Hz. The length of spike trains T_t is set to 500 ms, and the number of the synaptic inputs is set to 400. For each F_{in} and F_{out}, 20 experiments are carried out, and the average maximum learning accuracy is reported in Figure 11 and 12

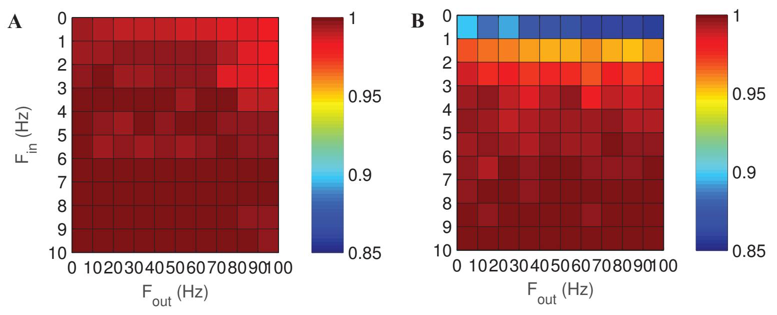

Figure 11: The comparison of learning performance between ReSuMe-DW and ReSuMe with different values of F_{in} and F_{out}. **(A)** Learning performance of ReSuMe-DW. **(B)** Leanrin performance of ReSuMe.

It is observed from Figure 11 and 12 that all the methods have a higher learning accuracy with a lower value of F_{out}, and there is a trend that the learning accuracy decreases as F_out increases. In addition, as shown in Figure 11, when F_{in} varies from 5 Hz to 10 Hz, both ReSuMe and ReSuMe-DW reach a high accuracy. The learning accuracy of ReSuMe-DW outperforms ReSuMe significantly when F_{in} varies from 1 Hz to 5 Hz. This also happens in Figure 12.

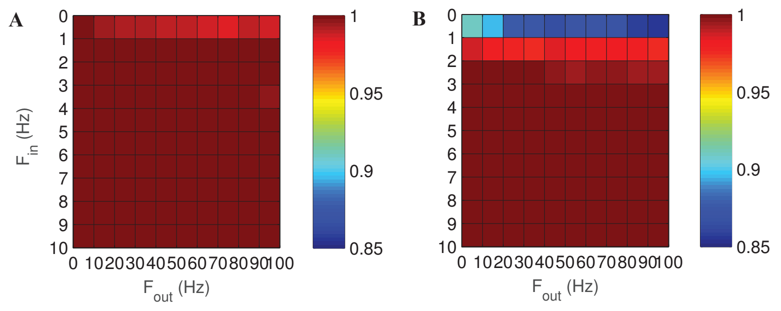

Figure 12: The comparison of learning performance between PBSNLR-DW and PBSNLR with different values of F_{in} and F_{out}. **(A)** Learning performance of PBSNLR-DW. **(B)** Learning performance of PBSNLR.

4.5. Robustness to Noise

Now we move on to investigate the robustness of the joint synaptic delay-weight plasticity. A neuron with 400 synaptic inputs is considered. The input and target spike trains are generated randomly according to a homogeneous Poisson process with a frequency $F_{in} = 2$ Hz and $F_0 = 100$ Hz, respectively. The total trains length T_t is set to 500 ms, and the number of synaptic input N_s is set to 400. After training, the reliability of the target recall is tested against two noise cases: 1) background noise on the membrane potential; 2) presynaptic spike time jitter.

4.5.1. Membrane Potential Noise

In this case, background membrane potential noise is considered as the noise source. After learning, the trained neuron is subjected to simulated background Gaussian white noise. The mean value of the added noise is 0, and its variance σ_b is systematically increased within the range of $[0.05, 0.5]$. For each value σ_b, 20 experiments are carried out. The learning accuracy C of a similarity between the desired and actual output spike trains is calculated and reported. The experimental results are shown in Figure 13.

Figure 13 shows that all learning methods work well with high learning accuracy when noise level is low. However, PBSNLR-DW and ReSuMe-DW

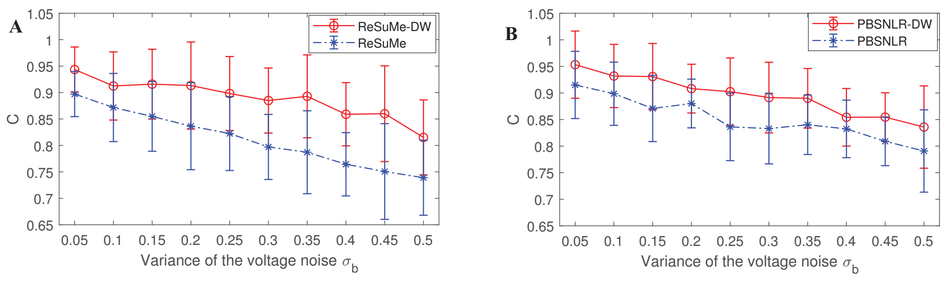

Figure 13: Robustness of different learning algorithms against background membrane potential noise. **(A)** Robustness of ReSuMe-DW and ReSuMe against back ground voltage noise. **(B)** Robustness of different learning algorithms against background membrane potential noise

outperform PBSNLR and ReSuMe consistently at all noise levels. These results confirm that the neuron trained by the proposed joint synaptic delay-weight plasticity helps the existing learning methods to improve their robustness.

4.5.2. Input Spike Time Jitter

In this case, input jittering noise is considered as the noise source. After learning, we jitter the input spike times. The jitter intervals are randomly drawn from a Gaussian distribution with mean 0 and variance $\sigma_j \in [0.2, 2]$ ms. We report the recall accuracy C with spike time jitters in Figure 14.

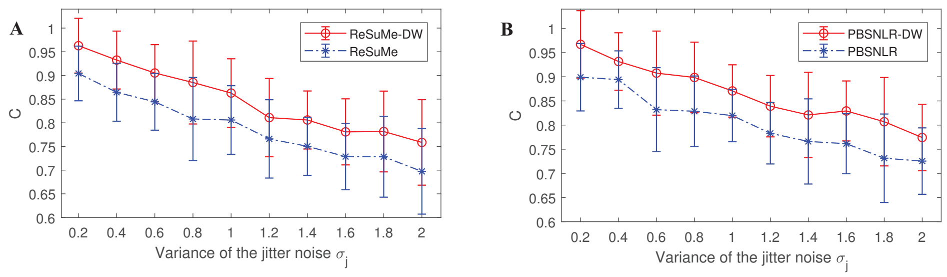

Figure 14: Robustness of different learning algorithms against jittering noise. **(A)** Robustness of ReSuMe-DW and ReSuMe against jittering noise. **(B)** Robustness of different learning algorithms against jittering noise

As shown in Figure 14, as the intensity of noise increases, the learning accuracy C decreases. We note that neurons trained by ReSuMe-DW and PBSNLR-

DW are more robust than those by ReSuMe and PBSNLR against jitering noise.

4.6. On Spiking Sparsity

In this section, we present the experiments to quantitatively evaluate the spike sparsity improvement by incorporating the proposed synaptic delay. A spiking neuron with 400 synaptic inputs is trained to emit a precisely timed spike sequence. The length of input and desired output spike trains is 400 ms. The mean frequency of the desired output spike train is set to F_o=50 Hz, and the firing rates of the input spike trains F_{in} vary from 1 Hz to 10 Hz with an interval of 1 Hz. The initial synaptic weights are selected as the uniform distribution in the interval [0, 0.01]. The initial synaptic delays are selected from the uniform distribution in the interval [0, 5]ms. We record the learning accuracies of both ReSuMe-DW and ReSuMe in Figure 15.

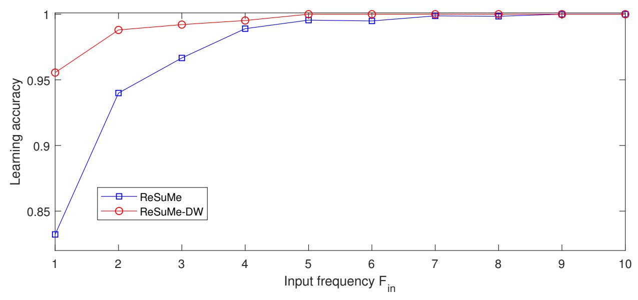

Figure 15: Learning accuracies of both ReSuMe and ReSuMe-DW with different input frequency.

Figure 15 demonstrates that the learning accuracies of both ReSuMe and ReSuMe-DW increase with the increase of the input spike frequency. The proposed ReSuMe-DW can make the learning neuron output desired precisely timed spike with an input frequency of 5 Hz, while ReSuMe requires a 8 Hz input spike frequency. To better illustrate the spiking sparsity improvement, Figure 16 gives an example of input spike pattern with frequency of 5 Hz and 8 Hz. By incorporating the proposed delay learning, the number of required spikes is reduced by nearly half. Therefore, the proposed ReSuMe-DW works more effectively with

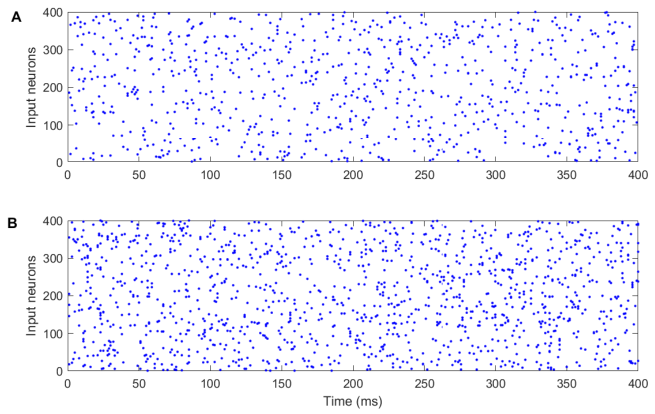

Figure 16: Input spike patterns with different frequencies. Each spike is denoted by a dot. **(A)** Input spike patterns with a frequency of 5 Hz. **(B)** Input spike patterns with a frequency of 8 Hz.

sparse input spike patterns, which means better energy efficiency.

4.7. Speech Recognition

The proposed ReSuMe-DW and PBSNLR-DW train spiking neurons to transform input spike patterns into a temporally specific output, and are hence well suited for processing temporally rich signals, such as motion and speech recognition. To demonstrate that the proposed learning algorithms are capable of capturing rich temporal dynamics, we apply them to solve a speaker-independent speech recognition task. The TIDIGITS corpus[54] is used in this experiment, which is one of the most commonly used datasets for benchmarking in speech recognition tasks[55, 56, 57, 58, 38, 59]. This dataset consists of isolated spoken digit strings from a vocabulary of 11 words (i.e.,'zero' to 'nine', and 'oh') and speakers from 22 different dialectical regions.

To fulfill the task of speech recognition, the raw speech waveforms are further processed by an auditory neural encoding front-end, in which sparse representation of spike timing patterns are obtained (as shown in Figure 17). The one-dimensional waveforms are first decomposed into 20 sub-channels with frequency range between 0 to 8000Hz, by a time-domain Constant-Q Transform cochlear filter bank [57]. Then we get the logarithm scale of the 20 parallel streams of sub-band signals, which mimics the function of energy detection of hair-cells in mammalian auditory systems. Finally each sub-band signal is encoded by threshold coding [29].

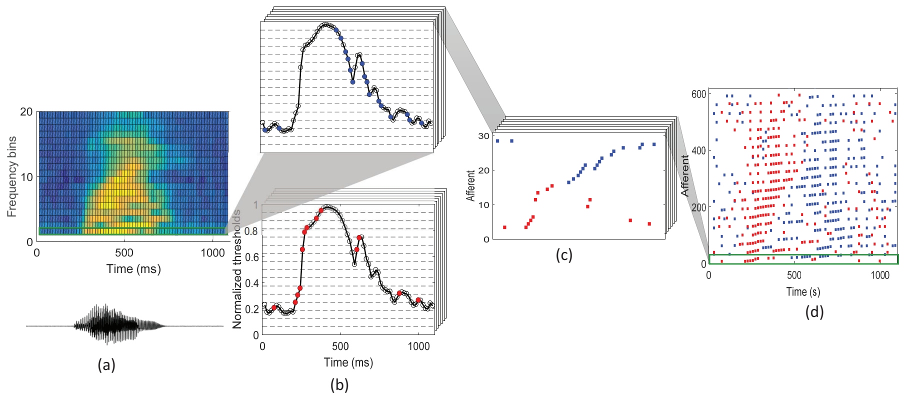

Figure 17: Illustration of the threshold coding mechanism. (a) The raw speech waveform and the spectrogram generated from the CQT cochlear filter bank. (b) The spectrogram is further encoded into spikes using the threshold coding. The top and bottom sub-figures depict the upward (red dots) and downward (blue dots) crossing events, respectively. For better visualization, only the output from the 1st cochlear filter is displayed. (c) The upward and downward events from (b) are merged to visualize the neuronal activation trajectory. The upward and downward crossing events for the 1st cochlear filter are represented by presynaptic neurons 1-30, respectively. The threshold coding preserves temporal dynamics of the filtered spectral information. (d) The entire threshold encoded spike pattern by concatenating the spike events from (c) vertically. The spike events that corresponds to the first filter in (c) is shaded in grey.

After threshold coding, the speech signals are transformed into spike patterns, and then the encoded spike patterns are transmitted to the next layer for learning and classification. There are eleven groups of output neurons in this layer with each group corresponding to one class. Each group consists of ten neurons. In order to discriminate between different spoken digits, neurons are trained to generate the desired spike train only when a spike pattern from their assigned class is presented, and remain silent otherwise. However, how to set the desired spike train remains an open question. To resolve this problem, we

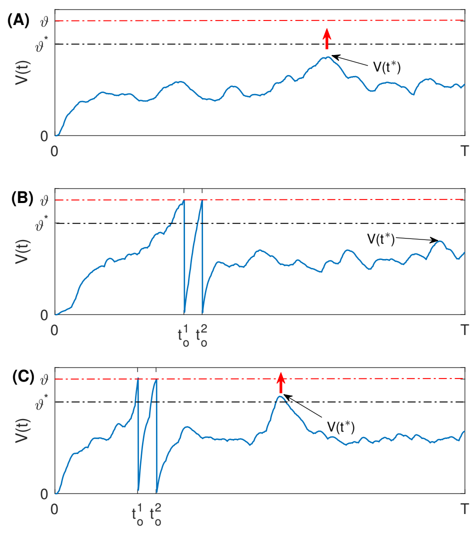

Figure 18: Illustration of the dynamic output decoding strategy. **(A)**The membrane potential remains lower than the firing threshold, then $T_d = t^*$. **(B)** The neuron generates two spikes and $V(t^*)$ is less than the pre-defined ϑ^*, then $T_d = \{t_o^1, t_o^2\}$. **(C)** The neuron generates two spikes and $V(t^*)$ is above ϑ^*, then $T_d = \{t_o^1, t_o^2, t^*\}$.

propose a data-driven dynamic decoding scheme.

When a training spike pattern is presented, we observe the membrane potential trace of the corresponding output neuron. The desired spike train T_d is decided based on the following cases:

- If the membrane potential $V(t)$ remains lower than the firing threshold ϑ and no spike is generated (as shown in Fig. 18A), then $T_d = t^*$, where t^* is the time at which the membrane potential $V(t^*)$ is the maximum among all peaks of subthreshold membrane potential.
- If the neuron generates a spike train $T_o = \{t_o^1, t_o^2, ...\}$ and $V(t^*)$ is less than the pre-defined decoding threshold ϑ^* (as shown in Fig. 18B), then $T_d = \{T_o\}$.
- If the neuron generates a spike train $T_o = \{t_o^1, t_o^2, ...\}$ and $V(t^*)$ is above

the pre-defined decoding threshold ϑ^* (as shown in Fig. 18C), then $T_d = \{T_o, t^*\}$.

As shown in Table. 1, it is encouraging to note that the proposed PBSNLR-DW and ReSuMe-DW achieve competitive results compared with other SNN-based models and traditional models. The classification accuracy of the proposed PBSNLR-DW is 96.50%, which outperforms all other bio-inspired systems on the TIDIGITS dataset. We note that the traditional RNN based systems offers a better classification accuracy of 97.90%[60]. However, our methods is based on the research of brain information processing mechanisms, which is fundamentally different from traditional deep learning approaches. In addition, the proposed delay learning rule can be applied to a deep spiking neural network to further enhance its classification accuracy.

Table 1: Classification performance of the proposed framework and other baseline frameworks on speech recognition task. The TIDIGITS corpus is used in this experiment, which consists of isolated spoken digit strings from a vocabulary of 11 words (i.e.,'zero' to 'nine', and 'oh') and speakers from 22 different dialectical regions.

Model	**Accuracy**
Single-layer SNN and SVM [61]	91.00%
Spiking CNN and HMM [62]	96.00%
AER Silicon Cochlea and SVM [58]	95.58%
Auditory Spectrogram and SVM [58]	78.73%
AER Silicon Cochlea and Deep RNN [38]	96.10%
Liquid State Machine [59]	92.30%
SOM and Tempotron Learning Rule [57]	92.10%
MFCC and GRU RNN [60]	97.90%
ReSuMe-DW	92.45%
PBSNLR-DW	96.50%

5. Discussion

5.1. On the Reasons of Better Learning Performance

The experimental results show that the proposed synaptic delay-weight learning method outperforms the traditional synaptic weight learning methods in terms of learning accuracy and efficiency. In this section, we explore why the proposed synaptic delay-weight learning can improve learning performance.

As shown in Figure 19A, there are three input synapses, and one spike arrives at each synapse. We would like to make the neuron j output a desired spike at t_d. Figure 19B shows the response of neuron j before learning, in which the membrane potential is below the firing threshold at t_d. To increase the membrane potential towards the firing threshold ϑ, the weight-based learning methods, such as ReSuMe or PBSNLR, will increase the synaptic weights. Figure 19C shows the neuronal response after weight-based learning, in which the neuron output a spike before t_d. By weight-based learning, the neuron cannot precisely output a spike at t_d. However, this problem is solved by the joint delay-weight learning rule. As shown in Figure 19D, the delay learning rule moves the input spikes toward the desired spike time.

As shown in Figure 20, there are 200 input spike trains, and the desired spike train contains three spikes($t_d(1)$,$t_d(2)$,$t_d(3)$). The spikes in the input spike pattern are denoted by a dot. In Figure 20A, the spatiotemporal input pattern does not have any spikes during the time interval shortly before the desired spike $t_d(2)$, and this time interval is called the silent window. In this situation, the weight-based learning rules, such as ReSuMe and PBSNLR, cannot make the learning neuron fire a spike at $t_d(2)$. However, this silent window problem can be solved by the proposed delay-weight learning method. As shown in Figure 20B, the proposed delay learning rule adjusts the synaptic delays to prepare some input spikes shortly before the desired spike $t_d(2)$, then adjust the synaptic weights to make the learning neuron fire a spike precisely at $t_d(2)$. This is one of the reasons for better learning performance of our method.

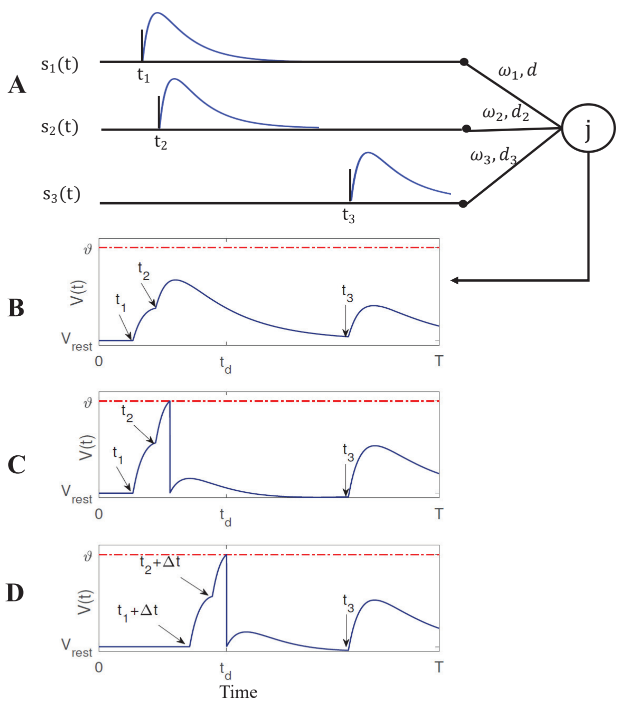

Figure 19: The necessity of delay based learning. **(A)** Structure of the neuron with three input synapses. **(B)** Membrane potential trace without learning. **(C)** Membrane potential trace after training with weight-based learning rules. **(D)** Membrane potential trace after traing with both weight-based and delay based learning rules.

5.2. On the Novelty of the Synaptic Delay Learning Rule

In section 1, we introduce several synaptic delay learning methods, such as delay selection [47],[48][19], delay shift method [49]. As DL-ReSuMe represents a recent success in learning synaptic delays, it is worth investigating the difference between the proposed joint synaptic delay-weight plasticity and DL-ReSuMe. While DL-ReSuMe outperforms ReSuMe in general, the delay learning method in DL-ReSuMe has some limitations. First, the synaptic delay in DL-ReSuMe can only be increased. Due to the one-way adjustment of synaptic delay, the learning efficiency is greatly affected. However, our proposed delay learning rule can both increase or decrease the synaptic delay to improve the learning performance. Secondly, the PSPs in DL-ReSuMe are limited to the exponential

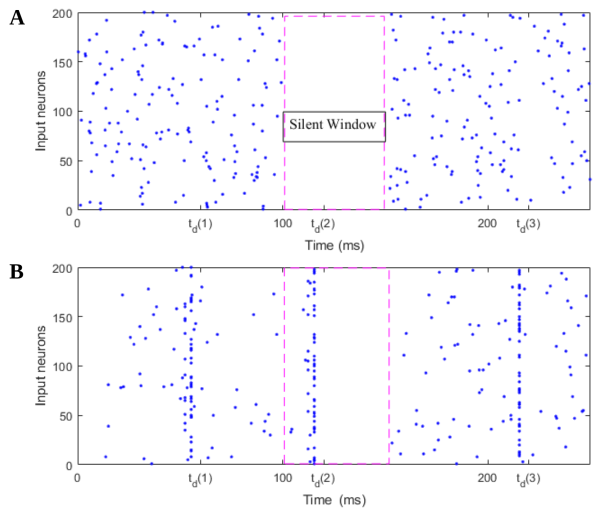

Figure 20: **(A)** Spatiotemporal input spike pattern containing 200 spike trains associated with 200 input neurons. Each spike is denoted by a dot. There are three desired spikes $t_d(1), t_d(2), t_d(3)$, and the input spike pattern does not have any spike in an interval shortly before the desired spike at $t_d(2)$. **(B)** The proposed delay learning rule adjust the synaptic delay to shift the input spikes toward the desired spikes.

function, which is less biological plausible, and cannot be generalized. However, the proposed delay learning algorithm is applicable to different types of PSP functions, and hence more general. Third, the delay learning rule in DL-ReSuMe has only been applied to ReSuMe, it is not clear how the delay learning rule can be generalized to other weight-based learning schemes. However, our proposed delay learning algorithm can be applied to both spike driven methods (such as ReSuMe) and membrane potential driven methods (such as PBSNLR).

5.3. On When to Apply Synaptic Delay Learning

In Figure 11, we observe that the learning accuracy of ReSuMe-DW outperforms ReSuMe significantly when F_{in} varies from 1 Hz to 5 Hz. However, when F_{in} varies from 5 Hz to 10 Hz, ReSuMe-DW and ReSuMe perform equally well. We believe that a higher input firing rate implies a higher number of postsynaptic potentials. In the case of high input spiking rate, the weight-based learning rule effectively optimizes the potential objective functions to achieve the target spike trains. However, in the case of low input spiking rate, the weight-based learning rule is less effective in moving the spiking time. Therefore, the synaptic delay learning is essential with sparse input spike patterns. In addition, when we apply SNNs for some practical applications, energy consumption is an important aspect of our consideration. The proposed delay-weight plasticity works more effective with sparse input spike patterns than traditional weight-based algorithms, which means better energy efficiency.

5.4. On Potential Application Prospects

Recently, there are several gradient based learning algorithms for training deep spiking neural networks, and these methods can be grouped into two main categories. The first category trains a traditional artificial neural network (ANN) and converts it into a SNN version with some accuracy loss [36, 37, 38]. The second category trains directly on the SNN with the typical examples from [31, 32, 33, 34, 35]. Experimental results demonstrate that these methods can achieve a good performance, and promote the research of SNN. However, the above mentioned learning algorithms are still based solely on weight adjustment, which completely ignore the effect of synaptic delay.

Despite the extensive exploration of learning algorithms, the exact learning mechanisms of a biological neuron remain unknown, and exploration of effective learning algorithm at the single neuron level is still necessary. To develop brain-inspired computing, this paper takes from a single neuron as a starting point, which proposes a new learning algorithm that adjusts both the synaptic delays and weights to make a neuron fire precisely timed spikes. This is a fundamental work to develop a more complex spiking neural network. Several works have attempted to integrate delay learning to deep spiking neural networks, and the experimental results demonstrate that the delay learning method can enhance the performance of deep SNNs[63]. Therefore, the proposed delay learning rule is a promising way to enhance the learning performance of deep SNNs on complex and large-scale tasks.

6. Conclusion

In this paper, we proposed a new delay based learning rule, which can be integrated into other existing weight-based learning rules. We illustrated how the proposed joint delay-weight plasticity works through ReSuMe-DW and PBSNLR-DW. Experimental results demonstrated that ReSuMe-DW and PBSNLR-DW achieve high learning accuracy with a substantial improvement in learning time and better robustness to different types of noise. Our future work will explore the possibilities to extend the proposed delay learning rule to sequence learning [64, 65, 66] and deep spiking neural networks.

7. Acknowledgments

This research is supported by Programmatic Grant No. A1687b0033 from the Singapore Government's Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain), the National Natural Science Foundation of China (Grant No. 61806040 and 61573081), the China Postdoctoral Science Foundation (Grant No. 2018M633348)

References

[1] W. Gerstner, W. M. Kistler, Spiking neuron models: Single neurons, populations, plasticity, Cambridge university press, 2002.

[2] S. B. Furber, F. Galluppi, S. Temple, L. A. Plana, The spinnaker project, Proceedings of the IEEE 102 (5) (2014) 652–665.

[3] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, K. Boahen, Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, Proceedings of the IEEE 102 (5) (2014) 699–716.

[4] A. Borst, F. E. Theunissen, Information theory and neural coding, Nature neuroscience 2 (11) (1999) 947.

[5] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al., A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 345 (6197) (2014) 668–673.

[6] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, et al., Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–10.

[7] N. Srinivasa, J. M. Cruz-Albrecht, Neuromorphic adaptive plastic scalable electronics: analog learning systems, IEEE pulse 3 (1) (2012) 51–56.

[8] E. Chicca, F. Stefanini, C. Bartolozzi, G. Indiveri, Neuromorphic electronic circuits for building autonomous cognitive systems, Proceedings of the IEEE 102 (9) (2014) 1367–1388.

[9] J. Gehlhaar, Neuromorphic processing: a new frontier in scaling computer architecture, ACM SIGPLAN Notices 49 (4) (2014) 317–318.

[10] Y. C. Yoon, Lif and simplified srm neurons encode signals into spikes via a form of asynchronous pulse sigma–delta modulation, IEEE transactions on neural networks and learning systems 28 (5) (2017) 1192–1205.

[11] W. Thach, On the specific role of the cerebellum in motor learning and cognition: Clues from pet activation and lesion studies in man, Behavioral and Brain Sciences 19 (3) (1996) 411–433.

[12] M. Ito, Mechanisms of motor learning in the cerebellum1, Brain research 886 (1-2) (2000) 237–245.

[13] M. Minsky, S. A. Papert, Perceptrons: An introduction to computational geometry, MIT press, 2017.

[14] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533.

[15] R.-M. Memmesheimer, R. Rubin, B. P. Ölveczky, H. Sompolinsky, Learning precisely timed spikes, Neuron 82 (4) (2014) 925–938.

[16] M. Zhang, H. Qu, X. Xie, J. Kurths, Supervised learning in spiking neural networks with noise-threshold, Neurocomputing 219 (2017) 333–349.

[17] M. Zhang, H. Qu, A. Belatreche, X. Xie, Empd: An efficient membrane potential driven supervised learning algorithm for spiking neurons, IEEE Transactions on Cognitive and Developmental Systems 10 (2) (2018) 151–162.

[18] S. M. Bohte, J. N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (1-4) (2002) 17–37.

[19] Y. Xu, X. Zeng, L. Han, J. Yang, A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks, Neural Networks 43 (2013) 99–113.

[20] F. Ponulak, A. Kasiński, Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting, Neural computation 22 (2) (2010) 467–510.

[21] Q. Yu, H. Tang, K. C. Tan, H. Li, Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns, Plos one 8 (11) (2013) e78318.

[22] R. V. Florian, The chronotron: a neuron that learns to fire temporally precise spike patterns, PloS one 7 (8) (2012) e40233.

[23] A. Mohemmed, S. Schliebs, S. Matsuda, N. Kasabov, Span: Spike pattern association neuron for learning spatio-temporal spike patterns, International journal of neural systems 22 (04) (2012) 1250012.

[24] J. D. Victor, K. P. Purpura, Metric-space analysis of spike trains: theory, algorithms and application, Network: computation in neural systems 8 (2) (1997) 127–164.

[25] M. v. Rossum, A novel spike distance, Neural computation 13 (4) (2001) 751–763.

[26] Y. Xu, X. Zeng, S. Zhong, A new supervised learning algorithm for spiking neurons, Neural computation 25 (6) (2013) 1472–1511.

[27] R. Gütig, H. Sompolinsky, The tempotron: a neuron that learns spike timing–based decisions, Nature neuroscience 9 (3) (2006) 420.

[28] M. Zhang, H. Qu, A. Belatreche, Y. Chen, Z. Yi, A highly effective and robust membrane potential-driven supervised learning method for spiking neurons, IEEE Transactions on Neural Networks and Learning Systems.

[29] R. Gütig, Spiking neurons can discover predictive features by aggregate-label learning, Science 351 (6277) (2016) aab4113.

[30] C. Albers, M. Westkott, K. Pawelzik, Learning of precise spike times with homeostatic membrane potential dependent synaptic plasticity, PloS one 11 (2) (2016) e0148948.

[31] E. O. Neftci, C. Augustine, S. Paul, G. Detorakis, Event-driven random back-propagation: Enabling neuromorphic deep learning machines, Frontiers in neuroscience 11 (2017) 324.

[32] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, A. Maida, Deep learning in spiking neural networks, Neural Networks.

[33] D. Huh, T. J. Sejnowski, Gradient descent for spiking neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 1433–1443.

[34] Y. Jin, W. Zhang, P. Li, Hybrid macro/micro level backpropagation for training deep spiking neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 7005–7015.

[35] E. O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks, arXiv preprint arXiv:1901.09948.

[36] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, M. Pfeiffer, Real-time classification and sensor fusion with a spiking deep belief network, Frontiers in neuroscience 7 (2013) 178.

[37] E. Hunsberger, C. Eliasmith, Spiking deep networks with lif neurons, arXiv preprint arXiv:1510.08829.

[38] D. Neil, S.-C. Liu, Effective sensor fusion with event-based sensors and deep network architectures, in: Circuits and Systems (ISCAS), 2016 IEEE International Symposium on, IEEE, 2016, pp. 2282–2285.

[39] H. Mostafa, Supervised learning based on temporal coding in spiking neural networks, IEEE transactions on neural networks and learning systems 29 (7) (2018) 3227–3235.

[40] F. Zenke, S. Ganguli, Superspike: supervised learning in multilayer spiking neural networks, Neural computation 30 (6) (2018) 1514–1541.

[41] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, W. Maass, Long short-term memory and learning-to-learn in networks of spiking neurons, in: Advances in Neural Information Processing Systems, 2018, pp. 787–797.

[42] B. Glackin, J. A. Wall, T. M. McGinnity, L. P. Maguire, L. J. McDaid, A spiking neural network model of the medial superior olive using spike timing dependent plasticity for sound localization, Frontiers in computational neuroscience 4 (2010) 18.

[43] B. Xu, Y. Gong, B. Wang, Delay-induced firing behavior and transitions in adaptive neuronal networks with two types of synapses, Science China Chemistry 56 (2) (2013) 222–229.

[44] M. Gilson, M. Bürck, A. N. Burkitt, J. L. van Hemmen, Frequency selectivity emerging from spike-timing-dependent plasticity, Neural computation 24 (9) (2012) 2251–2279.

[45] J.-W. Lin, D. S. Faber, Modulation of synaptic delay during synaptic plasticity, Trends in neurosciences 25 (9) (2002) 449–455.

[46] S. Boudkkazi, L. Fronzaroli-Molinieres, D. Debanne, Presynaptic action potential waveform determines cortical synaptic latency, The Journal of physiology 589 (5) (2011) 1117–1131.

[47] S. Ghosh-Dastidar, H. Adeli, Improved spiking neural networks for eeg classification and epilepsy and seizure detection, Integrated Computer-Aided Engineering 14 (3) (2007) 187–212.

[48] S. Ghosh-Dastidar, H. Adeli, A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection, Neural networks 22 (10) (2009) 1419–1431.

[49] P. Adibi, M. R. Meybodi, R. Safabakhsh, Unsupervised learning of synaptic delays based on learning automata in an rbf-like network of spiking neurons for data clustering, Neurocomputing 64 (2005) 335–357.

[50] A. Taherkhani, A. Belatreche, Y. Li, L. P. Maguire, Dl-resume: a delay learning-based remote supervised method for spiking neurons, IEEE transactions on neural networks and learning systems 26 (12) (2015) 3137–3149.

[51] Q. Yu, H. Tang, K. C. Tan, H. Li, Rapid feedforward computation by temporal encoding and learning with spiking neurons, IEEE transactions on neural networks and learning systems 24 (10) (2013) 1539–1552.

[52] C. Albers, M. Westkott, K. Pawelzik, Perfect associative learning with spike-timing-dependent plasticity, in: Advances in neural information processing systems, 2013, pp. 1709–1717.

[53] S. Schreiber, J.-M. Fellous, D. Whitmer, P. Tiesinga, T. J. Sejnowski, A new correlation-based measure of spike timing reliability, Neurocomputing 52 (2003) 925–931.

[54] R. G. Leonard, G. Doddington, Tidigits speech corpus, Texas Instruments, Inc.

[55] J. Wu, Y. Chua, H. Li, A biologically plausible speech recognition framework based on spiking neural networks, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, 2018, pp. 1–8.

[56] J. Wu, Y. Chua, M. Zhang, H. Li, A spiking neural network framework for robust sound classification, Frontiers in Neuroscience 12 (2018) 836.

[57] Z. Pan, H. Li, J. Wu, Y. Chua, An event-based cochlear filter temporal encoding scheme for speech signals, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, 2018, pp. 1–8.

[58] M. Abdollahi, S.-C. Liu, Speaker-independent isolated digit recognition using an aer silicon cochlea, in: Biomedical circuits and systems conference (biocas), 2011 ieee, IEEE, 2011, pp. 269–272.

[59] Y. Zhang, P. Li, Y. Jin, Y. Choe, A digital liquid state machine with biologically inspired learning and its application to speech recognition, IEEE transactions on neural networks and learning systems 26 (11) (2015) 2635–2649.

[60] J. Anumula, D. Neil, T. Delbruck, S.-C. Liu, Feature representations for neuromorphic audio spike streams, Frontiers in neuroscience 12 (2018) 23.

[61] A. Tavanaei, A. S. Maida, A spiking network that learns to extract spike signatures from speech signals, Neurocomputing 240 (2017) 191–199.

[62] A. Tavanaei, A. Maida, Bio-inspired multi-layer spiking neural network extracts discriminative features from speech signals, in: International Conference on Neural Information Processing, Springer, 2017, pp. 899–908.

[63] S. B. Shrestha, G. Orchard, Slayer: Spike layer error reassignment in time, in: Advances in Neural Information Processing Systems, 2018, pp. 1412–1421.

[64] P. Panda, K. Roy, Learning to generate sequences with combination of hebbian and non-hebbian plasticity in recurrent spiking neural networks, Frontiers in neuroscience 11 (2017) 693.

[65] P. Panda, J. M. Allred, S. Ramanathan, K. Roy, Asp: Learning to forget with adaptive synaptic plasticity in spiking neural networks, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 8 (1) (2018) 51–64.

[66] F. Zuo, P. Panda, M. Kotiuga, J. Li, M. Kang, C. Mazzoli, H. Zhou, A. Barbour, S. Wilkins, B. Narayanan, et al., Habituation based synaptic plasticity and organismic learning in a quantum perovskite, Nature communications 8 (1) (2017) 240.