A Supervised Learning Algorithm for Learning Precise Timing of Multiple Spikes in Multilayer Spiking Neural Networks

Aboozar Taherkhani, Ammar Belatreche, *Member, IEEE*, Yuhua Li, *Senior Member, IEEE*, and Liam P. Maguire, *Member, IEEE*

***Abstract*— There is a biological evidence to prove information is coded through precise timing of spikes in the brain. However, training a population of spiking neurons in a multilayer network to fire at multiple precise times remains a challenging task. Delay learning and the effect of a delay on weight learning in a spiking neural network (SNN) have not been investigated thoroughly. This paper proposes a novel biologically plausible supervised learning algorithm for learning precisely timed multiple spikes in a multilayer SNNs. Based on the spike-timing-dependent plasticity learning rule, the proposed learning method trains an SNN through the synergy between weight and delay learning. The weights of the hidden and output neurons are adjusted in parallel. The proposed learning method captures the contribution of synaptic delays to the learning of synaptic weights. Interaction between different layers of the network is realized through biofeedback signals sent by the output neurons. The trained SNN is used for the classification of spatiotemporal input patterns. The proposed learning method also trains the spiking network not to fire spikes at undesired times which contribute to misclassification. Experimental evaluation on benchmark data sets from the UCI machine learning repository shows that the proposed method has comparable results with classical rate-based methods such as deep belief network and the autoencoder models. Moreover, the proposed method can achieve higher classification accuracies than single layer and a similar multilayer SNN.**

***Index Terms*— Multilayer neural network, spiking neural network (SNN), supervised learning, synaptic delay.**

I. Introduction

SPIKE-timing-dependent plasticity (STDP) plays a prominent role in learning biological neurons, and it represents one form of synaptic plasticity which underpins synaptic weight changes based on the precise times of pre and postsynaptic spikes [1]. STDP highlights the important role of precise spike times in information processing in the brain [2]. In addition, the rapid sensory processing observed in the visual, auditory, and olfactory systems supports the assumption that information is encoded in the precise timing of the spikes [3]–[5]. Moreover, using precise timing of spikes results in a higher information encoding capacity compared with rate-based coding [6], and it can also convey the information related to rate of spikes in a multispike coding scheme [2]. Furthermore, as neural activity is metabolically expensive, the high number of spikes involved in rate coding scheme demands a significant amount of energy and resources [7], [8]. Despite the existing evidence supporting information encoding using the precise timing of spikes, the exact neuronal mechanisms that underlie learning to fire at precise times are still not clear and remain as one of the challenging problems in the field of spiking neural networks (SNNs) [2], [9]–[11].

In this paper, a novel supervised learning algorithm inspired by STDP is proposed to train an SNN to fire multiple spikes at precise desired times. Local synaptic biochemical events, produced by incoming spikes, are used to adjust weights and delays appropriately. In addition, neurons in the output and hidden layers interact with each other through a biofeedback signal sent by the output neurons to train the network. The main novelty of the proposed method consists in: 1) capturing the effect of synaptic delays on the learning of neuronal connection weights in an SNN, which has not been considered in previous works and 2) learning the spiking network synaptic delays. In addition, the proposed approach introduces an additional training mechanism to prevent the occurrence of undesired spikes which contribute to the misclassification of spatiotemporal input patterns. The proposed approach is validated using benchmark classification data sets and is compared against both spiking and rate-based neural models including state-of-the-art deep learning and autoencoder models. The experimental results show an improvement in learning accuracy over existing competitive SNN architectures and comparable performance to state-of-the-art rate-based neural models.

The remainder of this paper is structured as follows. A brief review of background and related work on SNNs is presented in Section II. Section III introduces the proposed method

Manuscript received April 4, 2017; revised October 27, 2017 and January 11, 2018; accepted January 16, 2018. This work was supported by the Ulster University's Vice Chancellor's Research Scholarship Scheme.
(Corresponding author: Aboozar Taherkhani.)

A. Taherkhani is with the Computational Neuroscience and Cognitive Robotics Laboratory, Nottingham Trent University, Nottingham NG1 4FQ, U.K. (e-mail: aboozar.taherkhani@ntu.ac.uk).

A. Belatreche is with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: ammar.belatreche@northumbria.ac.uk).

Y. Li is with the School of Computer Science and Informatics, Cardiff University, Cardiff CF10 3AT, U.K. (e-mail: yl7_99@yahoo.com).

L. P. Maguire is with the Faculty of Computing and Engineering, Ulster University, Londonderry BT48 7JL, U.K. (e-mail: lp.maguire@ulster.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2797801

in detail. The simulation results are then provided in Section IV. Finally, Section V concludes this paper.

II. Background and Related Work

Different artificial neural networks (ANNs) have been devised based on the working principle of their biological counterparts. AQ:1 ~~McCulloch and Pitts (1943) developed the first ANN where the neuron model is a logic unit which can be in an active or inactive (binary) mode depending on the weighted sum of their binary inputs.~~ Later, a continuous transfer function (e.g., sigmoid function) is applied to the weighted sum of continuous inputs to generate continuous output [12]. The continuous values represent the biological neuron spiking rates. ANNs are inspired by the biological nervous system and are successfully used in various applications. However, their high abstraction compared to their biological counterparts [13] and their inability to capture the complex temporal dynamics of biological neurons have resulted in a new area of ANNs where the focus is placed on more biologically plausible neuronal models known as SNNs. Thanks to their ability to capture the rich dynamics of biological neurons and to represent and integrate different information dimensions such as time, frequency, and phase, SNNs offer a promising computing paradigm and are potentially capable of modeling complex information processing in the brain [14]–[20].

In 1952, Hodgkin and Huxley [16] built a 4-D detailed conductance-based neuron model which can reproduce electrophysiological measurements to a high degree of accuracy. However, because of its intrinsic computational complexity, this model has a high computational cost. For this reason, simple phenomenological spiking neuron (SN) models are employed for simulating large-scale SNNs [15]. The leaky integrate-and-fire (LIF) model is a popular 1-D spiking neural model with low computational cost, but it offers relatively poor biological plausibility compared with the Hodgkin and Huxley model. Simple phenomenological SN models with low computational cost are highly popular for studies of neural coding, memory, and network dynamics [12].

The first supervised learning algorithms for multilayer SNNs using the precise timing of spikes could train only a single spike for each neuron. Bohte *et al.* [21] proposed the multilayer SNN called SpikeProp (inspired by the classical back-propagation algorithm) as one of the first supervised learning methods for feedforward multilayer SNNs. Backpropagation with momentum [22], QuickProp [22], resilient propagation [22], [23], and the SpikeProp based on adaptive learning rate [24] were proposed to improve the performance of SpikeProp. In all these methods, neurons in the input, output, and hidden layers can only fire a single spike.

Despite the capability of a single-spike learning method, single-spike coding schemes limit the diversity and capacity of information transmission in a network of SNs. In contrast, multiple spikes significantly increase the richness of the neural information representation [25], [26]. In addition, training a neuron to fire multiple spikes is more biologically plausible compared to single-spike learning methods [27], [28]. Temporal encoding through multiple spikes transfers important information which cannot be expressed by a single-spike coding scheme or a rate coding scheme. Although the exact mechanism of information coding in the brain is not clear, biological evidence shows that multiple spikes have a pivotal role in the brain. For instance, mapping between spatiotemporal spiking sensory inputs composed of spike trains to precise timing of spikes is an essential characteristic of neuronal circuits of the zebra finch brain to execute well-timed motor sequences [29]. In the mixed approaches proposed in [30] and [31], it is suggested that using both spike timing and spike rate increases processing speed. These methods use a combination of both correlated and uncorrelated spiking signals. So, there is useful information in the spike rate that cannot be captured by the precise timing of single spikes. Encoding information in the precise timing of multiple spikes which are used in this paper can capture not only the information in the spike rate but also the information in inter spike intervals.

Pfister *et al.* [32] designed a supervised learning algorithm for a single SN which updates synaptic weights to increase the likelihood of postsynaptic firing at several desired times. The algorithm is designed to train only a single neuron; however, it can train the neuron to fire multiple desired spikes. ~~ReSuMe~~ [25], spike pattern association neuron [33], AQ:2 perceptron-based SN learning rule [34], biologically plausible supervised learning method (BPSL) [35], and efficient membrane potential-driven supervised learning method [36] are other examples of learning methods that can train a single neuron to fire multiple desired spikes. Multispike learning methods focus on a single neuron or a single layer of neurons. It is difficult to design a multilayer SNN to fire multiple desired spikes because the complexity of the learning task is increased [27], [37]. In this situation, the learning algorithm should control several neurons to generate different desired spikes. However, a real biological nervous system is composed of a large number of interconnected neurons [27], [28], [37].

A multilayer neural network has a higher information processing ability than a single layer of neurons. Sporea and Grüning [28] have shown that a multilayer SNN can perform a nonlinearly separable logical operation; however, the task cannot be accomplished without the hidden layer neurons.

Ghosh-Dastidar and Adeli [37] and Booij and tat Nguyen [38] extended the multilayer SpikeProp [21] to allow each neuron in the input and hidden layers to fire multiple spikes. However, each output neuron can fire only a single spike. Xu *et al.* [27] proposed the first supervised learning method based on the classical error back-propagation method that can train all the neurons in a multilayer SNN to fire multiple spikes. Gradient learning methods suffer from various known problems which can lead to learning failure such as sudden jumps (called surge) or discontinuities in the error function [24]. The problem becomes more severe when the output neurons are trained to fire more than a single spike. In addition, the construction of an error function becomes difficult when multiple desired spikes should be learned as the number of actual output spikes may differ from the number of desired spikes in each learning epoch [27]. After investigation of the gradient-based methods in [23], [39], and [40], it is concluded that the application of STDP is worth further investigation to implement a more biologically plausible learning algorithm for multilayer SNNs [37].

Sporea and Grüning [28] have used STDP and anti-STDP to devise the first biologically plausible supervised learning algorithm for the classification of real-world data by a multilayer SNN in which each neuron in the input, hidden, and output layers can fire multiple spikes. The authors did not consider the spikes fired by hidden neurons when training the hidden neurons parameters. However, in a biological neuron, STDP usually works on the pre- and postsynaptic spikes of the neuron. In addition, the output spikes of the hidden neurons have significant effects on a training task in a multilayer SNN. Another drawback of this method [28] is that it has used the same learning adjustment method for inhibitory and excitatory neurons in hidden layers. However, inhibitory and excitatory neurons have different effects in a network by generating positive and negative postsynaptic potentials (PSPs). In this paper, a method is proposed to use spikes fired by hidden neurons during learning, and excitatory and inhibitory neurons are trained appropriately.

Delays of spike propagation are an important characteristic of real biological neural systems, and they have a significant effect on the information processing ability of the nervous system [18], [41], [42]. In ~~EDL~~ [43], an extended delay learning-based remote supervised method for SNs, and in DL-ReSuMe [41], a delay learning-based remote supervised method for SNs, investigated the viability of adjusting the neuron synaptic weights and delays for training a single SN to map a given spatiotemporal input pattern into a desired output spike train. STDP and anti-STDP were used to adjust the synaptic weights, and a delay shift approach was used to adjust their delays. It is worth noting that constant synaptic delays have been employed in [28], hence neglecting the effect of a synaptic delay between a hidden neuron and an output neuron on the weight adjustment of the hidden neuron. It trains the hidden neuron to fire at the time of an output desired spike. However, the generated spike is shifted by the network synaptic delay and causes an error in the firing time of the output neuron. SpikeProp and its related gradient-based methods [21], [23], [37] have taken into account the effect of a delay between a hidden neuron and an output neuron on the input weight adjustment of the hidden neurons. However, the use of multiple connections with different delays after a hidden neuron causes each of the different delays to affect the adjustment of the hidden neuron weights in different and opposite directions. Because, different errors are propagated from an output neuron to a hidden neuron corresponding to the different subconnections between the two neurons. The different errors force the hidden neuron to fire at different times depending on the different delays related to the multiple connections, and it disturbs the learning procedure. This might be one reason for the huge sudden rise in learning error of SpikeProp, as reported in [24].

In this paper, a learning algorithm is proposed to train both weights and delays of a multilayer SNN to fire multiple desired spikes. In the proposed method, each neuron at input, hidden, and output layers can fire multiple spikes. Supervised training of SNs which fire multiple spikes in a multilayer SNN remains a challenge. Furthermore, the proposed approach trains the synaptic delays in the multilayer SNN and also takes into the effect of delays on weight adjustments which is not considered in [21]–[24] and [28]. In the proposed method, the effect of the delays between a hidden neuron and an output neuron is considered during weight adjustments of the hidden neuron. In addition, the proposed method trains the weights of the hidden neurons by using the spikes fired by hidden neurons during STDP and anti-STDP, which results in a more biologically plausible and a highly accurate learning. Moreover, different weight adjustment strategies are used to train excitatory and inhibitory hidden neurons based on the effect of the excitatory (positive) and inhibitory (negative) PSPs (EPSP and IPSP) produced by the trained hidden neurons. In Section II, the principle of the proposed method is described.

III. Materials and Methods

The aim of the proposed supervised learning algorithm is to train a multilayer SNN to map spatiotemporal input patterns to their corresponding desired spike trains which implements a classification of the spatiotemporal input patterns. The network is composed of an input, a hidden, and an output layer. An output neuron, called a readout neuron, is fully connected to the hidden neurons. A spatiotemporal input pattern is emitted by the neurons in the input layer. Each input neuron is randomly connected to a fraction number of hidden neurons as used in [18]. The LIF neuron model described in [41] is used. The proposed method trains the spiking network by adjusting the learning parameters of the hidden and output neurons in parallel.

A. Overview of the Proposed Learning Method

The proposed learning method aims to train the multilayer SNN to enable each readout (output) neuron to fire actual output spikes at desired times and to cancel out undesired output spikes. A remote supervising signal is considered for an output neuron similar to ReSuMe [25]. At the time of a desired spike where there are not any actual output spikes at the readout neuron, the network learning parameters are adjusted to increase the total PSP of the readout neuron to hit the threshold level and generate an actual output spike at the desired time by using biologically plausible local events. The output neuron does the following three activities in parallel at the desired spike time.

First, at the time of the desired spike, the output neuron sends back an instruction signal (biofeedback) that shows the time of desired spike to the hidden neurons. After receiving the instruction signal, an excitatory hidden neuron potentiates its weights based on STDP to fire an output spike (hidden spike) at a specific time interval before the desired time. The specific time interval is equal to the delay related to the connection between the excitatory hidden neuron and the output neuron. The effect of the generated hidden spike (i.e., the PSP generated by the hidden spike) is shifted to the desired spike time after the related delay between the hidden neuron and the output neuron. The potentiation of the excitatory hidden neuron weights is stopped when the hidden neuron firing rate reaches a certain value, because a biological neuron cannot fire with a limitless rate, and a refractory period will ensure an upper bound on the neuron firing rate. The excitatory hidden neuron weight potentiation at the time of a desired spike is also stopped when an actual spike is generated at the time of the desired spike by the output neuron. In addition, the feedback triggers an inhibitory hidden neuron to try to remove its output spikes fired a specific time interval before the desired time by using the long-term depression (LTD) of anti-STDP. The time interval is equal to the delay between the inhibitory hidden neuron and the readout neuron. The hidden neuron output spikes before the time interval affects the PSP of the readout neuron at the desired time, i.e., the hidden spikes generate delayed PSPs at the desired time. The reduction of the inhibitory hidden spikes helps the readout neuron to increase its total PSP at the desired time to hit the threshold level.

Second, similar to ReSuMe [25] the output neuron potentiates its weights that have a spike shortly before the desired time based on STDP to increase its PSP at the desired time to fire.

The third activity at the time of a desired spike where there are not any actual output spikes of the readout neuron is the adjustment of delays of the readout neuron to increase the PSP of the readout neuron at the desired time, based on EDL [43]. All the abovementioned activities are repeated at the time of other desired spikes in a multispike coding scheme.

At the time of an undesired output spike of the readout neuron (i.e., where there is an actual output spike and there are not any desired spikes), the learning algorithm should reduce the total PSP of the readout neuron at the time of the undesired output spike to remove it by applying the following three processes in parallel. First, the readout neuron sends a feedback to excitatory hidden neurons to instruct them to remove their output spikes. Each excitatory hidden neuron removes its spike fired at a precise time interval before the time of the undesired spike by using LTD based on anti-STDP and reduces its weights. The time interval for the hidden neuron is equal to the delay between the hidden neuron and the readout neuron. Consequently, the reduction of the excitatory hidden neuron weights can help the readout neuron to reduce its total PSP and to remove the undesired output spike. It is clear that the weight reduction should be applied to the excitatory neurons that have a number of output spikes. Therefore, the LTD is applied to the excitatory neurons when their firing rates are higher than a threshold rate. The threshold rate is set by trial and error. In addition, the feedback triggers each inhibitory hidden neuron to potentiate its weights based on the long-term potentiation of STDP. The weight potentiation increases inhibitory hidden spikes before a precise time interval (the time interval is equal to the delay between the hidden neuron and the readout neuron) before the undesired spike time to help the readout neuron to reduce its total PSP at the undesired output spike time. The second process is applied at the time of the undesired output spike and consists of a reduction of the readout neuron weights that have spikes at the undesired output spike time or shortly before it by using anti-STDP similar to ReSuMe [25]. The third process reduces the readout neuron total PSP at the time of the undesired spike by adjusting the delays of the readout neuron based on EDL [43].

The hidden layer spikes play an important role in the generation of the network output spikes (both at desired and undesired times). Generated spikes by different hidden neurons cooperatively increase the PSP of the output neuron at a desired time and help it to fire at the desired time. In addition, when the complexity of a learning task is increased by increasing the number of desired spikes and also by increasing the number of different training patterns for each class, it becomes difficult or impossible to train a single neuron to fire at all the desired times for all the training patterns. Different groups of hidden neurons can contribute in generating different desired spikes and cooperatively drive a readout neuron to fire at all the desired times for all the training patterns.

In Sections III-B and III-C, first the training rule of the output neurons is explained and then the training of the hidden neurons weights is described in detail.

B. Training the Output Neurons

The weights and delays of each output neuron are trained by EDL, as described in [43]. The delay adjustments in cooperation with the weight adjustments train an output neuron to increase its total PSP at a desired time to generate an actual output spike, and also the adjustments help the output neuron to reduce its PSP at undesired spike times and to remove undesired actual output spikes. The weights are trained by the following equation:

$$\frac{dw_{\text{oh}}(t)}{dt} = \left[s_o^d(t) - s_o^a(t)\right]\left[a + \int_0^{+\infty} \Psi(s) s_h(t - d_{\text{oh}} - s)\text{ds}\right] \tag{1}$$

where w_{oh} and d_{oh} are the weight and delay related to the connection between the hth hidden neuron and the oth output neuron, respectively. $s_o^d(t)$ and $s_o^a(t)$ are desired and actual output spike trains of the oth output neuron, respectively. $s_h(t)$ is the spike train fired by hth hidden neuron. a is a non-Hebbian parameter that can speed up the learning. $\Psi(s)$ is a learning window similar to that of STDP and has an exponential function as described by

$$\Psi(s) = \begin{cases} Ae^{-s/\tau}, & s \geq 0 \\ 0, & s < 0 \end{cases} \tag{2}$$

where τ and A are the exponential decay time constant and the amplitude of the learning window, respectively.

$x_{\text{oh}}(t)$, a local variable called spike trace, is used to train the delay related to the synapse that connect hth excitatory hidden neuron to oth output neuron. $x_{\text{oh}}(t)$ is governed by

$$x_{\text{oh}}(t) = \begin{cases} Ae^{-(t-t_h^f-\varepsilon_{\text{oh}})/\tau}, & t_h^f < t < t_h^{f+1} \\ A, & t = t_h^f \end{cases} \tag{3}$$

where t_h^f is the firing time of the fth spike of the hth excitatory hidden neuron, τ is the time constant of the exponential function, ε_{oh} is the delay between the hth excitatory

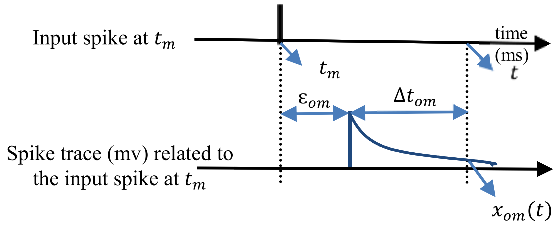

Fig. 1. Trace x_{om} related to input spike at t_m jumps to a maximum value after the delay $\varepsilon_{\mathrm{om}}$. Then it decays exponentially through time.

hidden neuron and the oth output neuron, and A is a constant value which are equal to their counterparts in (2). $x_{\mathrm{oh}}(t)$ is used to obtain appropriate value for delay adjustment. The adjustment $\Delta\varepsilon_{\mathrm{oh}}$ is calculated by (4) similar to EDL [43]

$$\Delta\varepsilon_{\mathrm{oh}}(t) = \begin{cases} +\Delta t_{\mathrm{om}}(t)(x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t))^4, & t = \hat{t}_o^f \\ -\Delta t_{\mathrm{om}}(t)(x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t))^4, & t = t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{4}$$

where $\hat{t}_o^f$ is the time of the fth desired spike, t_o^f is the time of the fth actual output spike of the oth output neuron, and $x_{\mathrm{om}}(t)$ is the maximum trace between the traces of the excitatory hidden neurons connected to the oth output neuron at the current time t. $x_{\mathrm{om}}(t)$ is corresponding to the connection between the mth excitatory hidden neuron (that has the closest spike before the current time t) and the oth output neuron. Δt_{om} is a delay shift which is necessary to be added to the delay between the mth excitatory hidden neuron and the oth output neuron to bring the effect of the closest spike fired by mth excitatory hidden neuron to the current time t. It is derived from (3) and calculated by

$$\Delta t_{\mathrm{om}} = t - t_m - \varepsilon_{\mathrm{om}} = -\tau_x \ln\left(x_{\mathrm{om}}(t)/A\right) \tag{5}$$

where t_m is the firing time of the mth excitatory hidden neuron before current time t. The mth excitatory hidden neuron has the closest spike before the current time t. It has the maximum trace at time $t x_{\mathrm{om}}(t)$ out of all excitatory input synapses of the oth output neuron. $x_{\mathrm{om}}(t)$ should be less than A, because the spike should occur before the current time. $\varepsilon_{\mathrm{om}}$ is the delay between the mth excitatory hidden neuron and the oth output neuron. Fig. 1 illustrates the relationship between the different parameters used in (5).

The delay adjustment in (4) tries to increase the total PSP of the oth output neuron at $t = \hat{t}_o^f$ and to reduce the total PSP at $t = t_o^f$. The delay increment in (4) shifts the positive PSPs generated by excitatory inputs to the desired times to generate an output spike. The delay reduction shifts the positive PSPs away from the actual output spikes times to remove undesired spikes. When an actual output spike is generated at the time of a desired spike, the positive delay adjustment cancels out the negative delay adjustment and the delays are stabilized. In (4), we have $[x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t)] \le 1$. The use of the fourth power in (4) reduces the amount of delay adjustment related to a far input spike. A far input spike corresponds to a low value of $[x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t)]$ and consequently a lower value of the fourth power of $[x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t)] \le 1$, and only the delays related to the close input spikes which have a high effect on the PSP is adjusted by a high value to prevent unnecessary change of the delays in the network.

The adjustment of delay between the hth inhibitory hidden neuron and the oth output neuron $\Delta\mu_{\mathrm{oh}}$ is governed by

$$\Delta\mu_{\mathrm{oh}}(t) = \begin{cases} -\Delta\bar{t}_{\mathrm{om}}(t)(\bar{x}_{\mathrm{oh}}(t)/(\bar{x}_{\mathrm{om}}(t))^4, & t = \hat{t}_o^f \\ +\Delta\bar{t}_{\mathrm{om}}(t)(\bar{x}_{\mathrm{oh}}(t)/\bar{x}_{\mathrm{om}}(t))^4, & t = t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{6}$$

where $\bar{x}_{\mathrm{oh}}(t)$ is the spike trace related to the connection between hth inhibitory hidden neuron and the oth output neuron. $\bar{x}_{\mathrm{om}}(t)$ is the maximum trace between the inhibitory hidden neurons that are connected to the oth output neuron. It should be less than A. $\Delta\bar{t}_{\mathrm{om}}(t)$ is calculated by putting $\bar{x}_{\mathrm{om}}(t)$ in (5). The decrement of delays in the first expression of (6) at the desired times shifts away the negative PSPs generated by inhibitory inputs (from the desired times) and increases the total PSP of the output neuron accordingly. This might increase the total PSP to hit the threshold level and generate an actual output at the desired times. The delay increment in the second expression relates to the inhibitory input spikes before the actual outputs shifts the negative PSP of the inhibitory inputs toward the actual output spikes to remove undesired output spikes. When an actual output spike is generated at the time of a desired spike, the delay decrement and increment in (6) are equal and the net adjustment becomes zero.

C. Training the Hidden Neurons

This section introduces the learning algorithms for both excitatory and inhibitory hidden neurons.

1) Weight Learning of Excitatory Hidden Neurons: The synaptic weight between the ith input neuron and the hth excitatory hidden neuron is denoted by w_{hi} and all the delays in the network are neglected in this stage. The synaptic weight adjustment is governed by

$$\Delta w_{\mathrm{hi}}(t) = \begin{cases} +\sum_o [\Psi(t-t_i)(1-\Psi(t-t_h)/A)](w_{\mathrm{oh}}/A), & t = \hat{t}_o^f \\ -\sum_o [\Psi(t-t_i)(\Psi(t-t_h)/A)](w_{\mathrm{oh}}/A), & t = t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{7}$$

where t_i is the last firing time of the ith input spike at or before the current time t. Equation (7) shows that the algorithm adjusts the weight at the time of the fth desired spike of the oth output neuron, $t = \hat{t}_o^f$, and at the time of the fth actual output spike of the oth output neuron, $t = t_o^f$. The sigma ($\sum$) collects the weight adjustment on all the output neurons. At the time of the desired spike, the weight is potentiated in proportion to the STDP time window ($\Psi(t - t_i)$) to generate hidden neuron spike at the desired time or shortly before it to increase the total PSP of the oth output neuron and help the output neuron to generate an actual output spike at the desired time (Fig. 2). Different hidden neurons correspond to different desired spikes, and they cooperatively force the output neuron to fire at all desired times.

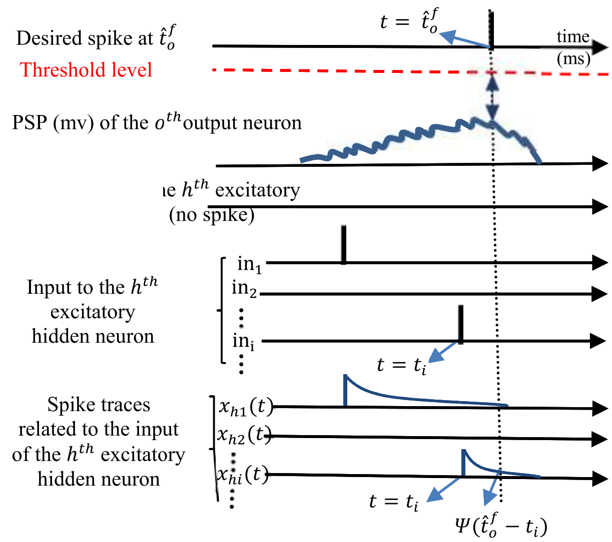

Fig. 2. Synaptic weight between ith input neuron and the hth excitatory hidden neuron w_{hi} is potentiated in proportion to the value of STDP time window $[\Psi(t - t_i)]$ at $t = \hat{t}_o^f$ to generate hidden spike at the desired time $t = \hat{t}_o^f$. The generated excitatory input will be fed to the oth output neuron, and it increases the total PSP of the neuron at the desired time.

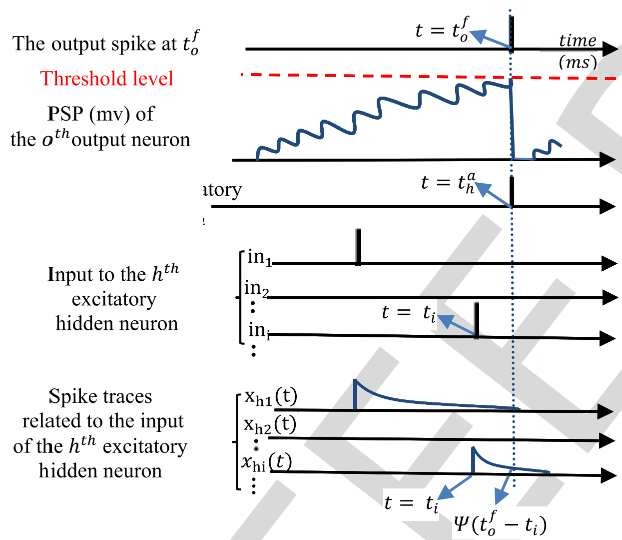

Fig. 3. w_{hi}, the synaptic weight between ith input neuron and the hth excitatory hidden neuron, is reduced in proportion to $\Psi(t - t_i)$, at $t = t_o^f$ (the time of the fth actual output spike of the oth output neuron). The reduction might lead to the cancelation of the hidden spike at t_h and consequently the reduction of the total PSP of the oth output neuron generated at $t = t_o^f$ and remove the actual output at $t = t_o^f$.

At the time of an actual output, $t = t_o^f$, $\Delta w_{\text{hi}}(t)$ is reduced in proportion to the STDP time window $\Psi(t - t_i)$. It depends on the time difference of its input spike t_i, and the current time $t = t_o^f$, $(t_o^f - t_i)$. The reduction might lead to the cancellation of the hidden spike at t_h shortly before $t = t_o^f$ or at t_o^f, and consequently reduces the total PSP of the oth output neuron generated at $t = t_o^f$ and remove the actual output at $t = t_o^f$ (Fig. 3). When the actual output spikes at $t = t_o^f$, it becomes close to the desired spike at $t = \hat{t}_o^f$, the positive weight adjustment related to the desired spike cancels out the negative weight adjustment at the actual output. Consequently, the net weight adjustment becomes small.

The excitatory hidden neuron weight is adjusted based on the three spikes shown in Fig. 3 by (7). In a triplet-STDP, which is a more accurate model of synaptic plasticity in a biological neuron than a standard pair-based STDP [1], three spikes also affect a weight adjustment. A triplet-STDP described in [1] uses a single presynaptic and two postsynaptic spikes. There are different models for triplet-STDP [1].

The term $[(1 - \Psi(t - t_h)/A)]$ in (7) prevents the weight change of an excitatory hidden neuron that already has an actual output at the desired time, $t = \hat{t}_o^f$ as in this situation $\Psi(\hat{t}_o^f - t_h) = A$, consequently, $[(1 - \Psi(\hat{t}_o^f - t_h)/A) = 0]$. Therefore, the weight increment related to the hidden w_{hi} is 0, because the hidden neuron already has a spike at this desired time and it does not need more weight adjustment. Different hidden neurons contribute to firing of the output neuron at different desired times and cooperatively help the output neuron to fire at all the desired spikes in a multispike coding scheme. The term also causes a smaller increment of the weight w_{hi} that has output spike closely before the desired spike $[\Psi(\hat{t}_o^f - t_h) \cong A$, consequently, $(1 - \Psi(\hat{t}_o^f - t_h)/A) \cong 0]$. An unnecessary high adjustment might shift the hidden spike close to $\hat{t}_o^f$ beyond the desired time and reduce the total PSP of the oth output neuron at the desired time. In addition, the term $(1 - \Psi(t - t_h)/A)$ causes a comparatively high increment of w_{hi} when a hidden neuron does not have spike before $t = \hat{t}_o^f$ [because $(1 - \Psi(\hat{t}_o^f - t_h)/A) = 1$], or the actual output of the hth hidden neuron is far from the desired time at $t = \hat{t}_o^f$ $[(1 - \Psi(\hat{t}_o^f - t_h)/A) \cong 1]$. The high increment might force the hth hidden neuron to fire at the desired time $t = \hat{t}_o^f$, and consequently increase the total PSP of the oth output neuron at the desired times $t = \hat{t}_o^f$.

The term $[\Psi(t - t_h)/A]$ in (7) when $t = t_o^f$ prevents the reduction of w_{hi} if the hth excitatory hidden neuron does not have any actual output spikes before the actual output of the oth output neuron at $t = t_o^f$ $[(\Psi(t_o^f - t_h)/A) = 0]$. Because, w_{hi} does not have any roles in the generation of the output spike at $t = t_o^f$. If an excitatory hidden neuron has output spike before and close to an actual output spike at $t = t_o^f$, the term has comparatively a high value $[(\Psi(t_o^f - t_h)/A) \cong 1]$, and consequently, w_{hi} is adjusted with a higher value, because the excitatory hidden neuron has a strong contribution in the generation of the actual output spike at $t = t_o^f$ and the weight reduction might lead to the removal of the output from the excitatory hidden neuron and consequently reduce the total PSP of the output neuron.

In a network with nonzero delays, the proposed method trains the excitatory hidden neuron to fire at a time interval (equal to the corresponding delay connecting the hidden neuron to the output neuron) before a desired time. The early firing of the excitatory hidden neuron increases the total PSP of its successor output neuron at the desired time by the delayed effect of the excitatory hidden spike. However, in the previous situation, where the connections do not have any delays, an excitatory hidden neuron is trained to fire at the same time as the desired time. Correspondingly, (8) is used to adjust w_{hi}, the synaptic weights between the ith input neuron and the hth excitatory hidden neuron, at time t

$$\Delta w_{\mathrm{hi}}(t) = \begin{cases} +\sum_o [x_{\mathrm{hi}}(t-\varepsilon_{\mathrm{oh}})(1-x_{\mathrm{oh}}(t)/A)](w_{\mathrm{oh}}/A), & t=\hat{t}_o^f \\ -\sum_o [x_{\mathrm{hi}}(t-\varepsilon_{\mathrm{oh}})(x_{\mathrm{oh}}(t)/A)](w_{\mathrm{oh}}/A), & t=t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{8}$$

where $x_{\mathrm{hi}}(t)$ is the spike trace corresponding to the connection between the ith input neuron and the hth excitatory hidden neuron. Each spike in the ith input spike train causes a delayed ($\varepsilon_{\mathrm{hi}}$) jump in the trace then it decays exponentially by a time constant similar to (3). $x_{\mathrm{oh}}(t)$ is the trace corresponding to the connection between the hth excitatory hidden neuron and the oth output neuron. Each output spike of the hth excitatory hidden neuron results in a delayed ($\varepsilon_{\mathrm{oh}}$) jump in the trace which decays exponentially by a time constant τ similar to (3). $\varepsilon_{\mathrm{hi}}$ is the delay between the ith input neuron and the hth excitatory hidden neuron, and $\varepsilon_{\mathrm{oh}}$ is the delay between the hth excitatory hidden neuron and the oth output neuron. The traces have same amplitude A and time constant τ as the STDP time window in (2).

The update of w_{hi} at $t=\hat{t}_o^f$ in (8) based on the delayed $x_{\mathrm{hi}}(t)$ increases w_{hi} by a high value if it has spike shortly before $(\hat{t}_o^f-\varepsilon_{\mathrm{oh}})$, because in this case $x_{\mathrm{hi}}(\hat{t}_o^f-\varepsilon_{\mathrm{oh}})$ has a high value. The high increase can lead to the generation of an output spike of the hth excitatory hidden neuron at $(\hat{t}_o^f-\varepsilon_{\mathrm{oh}})$. The effect of the generated hidden spike is shifted to the time of the desired spike in the oth output neuron after the delay of the connection between the hth excitatory hidden neuron and the oth output neuron $\varepsilon_{\mathrm{oh}}$. This helps the output neuron to generate output spike at the desired time.

The decrement in the second expression of (8) is high if the ith input neuron has spike shortly before $(t_o^f-\varepsilon_{\mathrm{oh}})$. Consequently, this decrement tries to remove the actual output of the hth excitatory hidden neuron at $(t_o^f-\varepsilon_{\mathrm{oh}})$ and helps the oth output neuron to reduce its PSP at the time t_o^f (by considering the delay $\varepsilon_{\mathrm{oh}}$).

2) Weight Learning of the Inhibitory Hidden Neurons: The connection weight between the hth inhibitory hidden neuron and the ith input neuron $\bar{w}_{\mathrm{hi}}$ is updated similar to (8) by multiplying it with a negative sign as shown in

$$\Delta \bar{w}_{\mathrm{hi}}(t) = \begin{cases} -\sum_o [\bar{x}_{\mathrm{hi}}(t-\mu_{\mathrm{oh}})(\bar{x}_{\mathrm{oh}}(t)/A)]|w_{\mathrm{oh}}/A|, & t=\hat{t}_o^f \\ +\sum_o [\bar{x}_{\mathrm{hi}}(t-\mu_{\mathrm{oh}})(1-\bar{x}_{\mathrm{oh}}(t)/A)]|w_{\mathrm{oh}}/A|, & t=t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{9}$$

where μ_{oh} is the delay between the hth inhibitory hidden neuron and the oth output neuron, and $\bar{x}_{\mathrm{hi}}(t)$ is the spike trace corresponding to the connection between the ith input neuron and the hth inhibitory hidden neuron. $\bar{x}_{\mathrm{oh}}(t)$ is the spike trace related to the connection between the hth inhibitory hidden neuron and the oth output neuron. The delay related the connection between the ith input neuron and the hth inhibitory hidden neuron is μ_{hi}. According to (9), the weight is reduced if the ith input neuron has a delayed (μ_{hi}) spike shortly before $(\hat{t}_o^f-\mu_{\mathrm{oh}})$ to increase the total PSP of the oth output neuron at the desired time $\hat{t}_o^f$ by removing hidden inhibitory spike at or before $(\hat{t}_o^f-\mu_{\mathrm{oh}})$. In addition, (9) increases the weight $\bar{w}_{\mathrm{hi}}$ to generate hidden inhibitory spike at $(t_o^f-\mu_{\mathrm{oh}})$ to reduce the total PSP of the oth output neuron at $t=t_o^f$. The reduction of the total PSP removes the actual output spike of the oth output neuron at t_o^f.

It is proposed that hidden neurons receive biofeedback from the readout neurons. Through this biofeedback, the times of desired spikes and actual outputs related to the neurons in the next layer are made available at the hidden layer neurons which use them to adjust their weights appropriately. In this paper, we did not describe the basis of the biofeedback or model it in detail. The training of the network is stopped when it reaches its goal, i.e., the readout neuron generates actual output spikes at the desired times and all the undesired output spikes of the readout are removed.

D. Classification Ability of the Proposed Method

The weight and delay learning characteristics of the proposed method enable it to train a neuron to fire at desired spike times related to an applied input pattern. In a classification task, an input pattern is assigned to the class whose desired spike train is most similar to the actual output of the network. Therefore, the classification ability of the proposed method can be improved if an output neuron is also trained not to fire close to the desired spikes of other classes in addition to firing at the desired times representing to the current class of the input pattern. As a result, the proposed method introduces an additional learning mechanism when a misclassification occurs.

The learning algorithm considers two desired spike trains after a misclassification. The first one is related to the class of the applied input spatiotemporal pattern, i.e., the desired spikes of the correct class, and the second one is related to the class that causes the misclassification (incorrect class). Thus, the learning adjusts the readout neurons and hidden neurons learning parameters at the time of each desired spike related to the class that causes the misclassification. It reduces the weights of the readout neuron that have a spike before the desired time. To force the oth output neuron to not fire at the fth desired spike of class j $(t=\hat{t}_o^{f(j)})$ the weights of the $o$thoutput neuron are adjusted by the following equation at $t=\hat{t}_o^{f(j)}$:

$$\Delta w_{\mathrm{oh}}(t) = -\Psi(t-t_h-d_{\mathrm{oh}}). \tag{10}$$

The proposed classification learning method adjusts an excitatory hidden neuron weight at the desired spike times $(t=\hat{t}_o^{f(j)})$ related to the class that causes the misclassification by the following equation similar to (8):

$$\Delta w_{\mathrm{hi}}(t) = -\sum_o [x_{\mathrm{hi}}(t-\varepsilon_{\mathrm{oh}})(x_{\mathrm{oh}}(t)/A)](w_{\mathrm{oh}}/A). \tag{11}$$

An inhibitory hidden neuron weight at $t=\hat{t}_o^{f(j)}$ is adjusted similar to (9) by the following equation:

$$\Delta \bar{w}_{\mathrm{hi}}(t) = +\sum_o [\bar{x}_{\mathrm{hi}}(t-\mu_{\mathrm{oh}})(1-\bar{x}_{\mathrm{oh}}(t)/A)]|w_{\mathrm{oh}}/A|. \tag{12}$$

The delay related to an excitatory input of a readout neuron is adjusted by (13) at $t = \hat{t}_o^{f(j)}$. The following equation is similar to (4):

$$\Delta\varepsilon_{\mathrm{oh}}(t) = -\Delta t_{\mathrm{om}}(t)(x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t))^4 \tag{13}$$

The delay related to an inhibitory input of the readout at $t = \hat{t}_o^{f(j)}$ is adjusted through the following equation which is similar to (6):

$$\Delta\mu_{\mathrm{oh}}(t) = +\Delta\bar{t}_{\mathrm{om}}(t)(\bar{x}_{\mathrm{oh}}(t)/\bar{x}_{\mathrm{om}}(t))^4. \tag{14}$$

The proposed method uses a criterion to control the learning level of every pattern and manage the misclassifications during training and adjust the network learning parameters to increase the inter class separability of the network.

Consider a pattern from class i is applied to the network and an actual output of the network is generated. The correlation between the actual output and the corresponding desired spike train of the class i is called c_i which is calculated by the method used in [41] as in

$$c_i = \frac{\upsilon_d \cdot \upsilon_o}{|\upsilon_d||\upsilon_o|} \tag{15}$$

where "$\upsilon_d \cdot \upsilon_o$" denotes the inner product of the two vectors υ_d and υ_o. υ_d and υ_o are two vectors with real value components which are generated from spike trains. A desired spike train is convolved with a symmetric Gaussian function to generate υ_d. Similarly, υ_o is generated by convolving an actual output spike train with the symmetric Gaussian function. $|\upsilon|$ is the length of a vector υ.

A maximum value p and a threshold level Δc for c_i are considered to control the learning. If the correlation metric c_i is less than Δc, the network learning parameters are updated based on the applied training pattern and their desired spike train without considering any extra criteria. In this situation, the network adjusts its learning parameters to increase its knowledge about the applied training pattern inside the class i. The low value of the correlation related to the applied training pattern $c_i < \Delta c$ means that the similarity of the training pattern with the previous trained patterns from the same class i is low and the learning parameters of the network should be adjusted to increase the ability of the network to recognize the patterns inside the class i.

If c_i reaches the value of p, the learning related to the pattern is not applied to the network in the current learning epoch, because the high value of the correlation shows that the knowledge of the presented training pattern is already in the network and it is not necessary to adjust the learning parameters for the current value of c_i. It means that the network has learned the overall distribution of the data from the class i and it is not necessary to memorize all the details of the presented training pattern. It also prevents over training of the network.

If c_i has a value between Δc and p, i.e., $(\Delta c < c_i < p)$, and c_i is appropriately higher than the correlation metric related to the other classes to prevent misclassification, then the learning related to the applied pattern is stopped in the current epoch. Therefore, if $\Delta c < c_i < p$ and $c_i > c_j + \Delta c$ (where $j = \mathrm{argmax}_{\{k\in\{1,2,\ldots,N\}\&k\neq i\}} c_k$, c_k is the correlation metric of the actual output with the kth desired spike train, and N is the number of all the classes), the learning adjustment related to the applied pattern from class i is not applied to the network in the current epoch. The $c_i > c_j + \Delta c$ denotes that the network can distinguish the class of the applied pattern correctly with an appropriate margin (Δc), therefore it is not necessary to have more training for the current value of c_i in the learning epoch.

TABLE I

PROPOSED CLASSIFICATION LEARNING METHOD

when a training pattern from class 'i' is presented in a learning epoch:

If $c_i \leq \Delta c$,

The corresponding weights and delays are adjusted to increase c_i i.e. train the network to generate the i^{th} class desired spike.

if $\Delta c < c_i < p$ and $c_i < c_j + \Delta c$

- $c_i < c_j + \Delta c$ implies that c_i has a low value and it could cause a misclassification. So c_i needs to be increased by learning the i^{th} desired spike train (using (1) for training output neurons' weights, (4) and (6) for training output neurons' delays, and (8) and (9) for training hidden Neurons' weight)
- c_j needs to be reduced by training the network to not fire close to the j^{th} class desired spike train (using (10) for training output neurons' weights, (11) and (12) for training hidden neurons' weights, (13) and (14) for training Output neurons delays).

if $(\Delta c < c_i < p$ and $c_i > c_j + \Delta c)$ or $(c_i \geq p)$

The learning parameter adjustment related to the training pattern is not applied to the network in the current epoch. Because c_i has reached an acceptable level in this epoch.

End

If c_i has a value between Δc and p, and $c_i < c_j + \Delta c$, it suggests that a misclassification has occurred. In this situation, the network learning parameters are updated to enhance the interclass separability of the network by training it to not fire close to the desired spike train of the class that causes this misclassification and to reduce c_j. The learning parameters are also updated to increase the ability of the network to generate the desired spike related to the applied pattern from the class i to increase c_i. The reduction of c_j and the increment of c_i may change the situation $c_i < c_j + \Delta c$ to $c_i > c_j + \Delta c$ and prevent the misclassification. The training is continued until the maximum number of learning epochs is reached or if the stopping criteria noted in Table I apply.

A c_i greater than p shows that the network is trained to fire appropriately close to the corresponding desired spike train. Therefore, similar to the situation where $(\Delta c < c_i \leq p$ and $c_i > c_j + \Delta c)$ the related learning adjustment is not applied to the network. The p value is chosen high enough depending on the desired spike trains related to the different classes to guarantee that when $c_i > p$, c_i is appropriately higher than c_j $(c_i > c_j + \Delta c)$. Desired spike trains related to different classes (related to c_i and c_j) should be chosen in a such a way that the correlation between the desired spike trains are low enough to support the point that if an actual spike train is very similar to the desired spike related to c_i, $(c_i > p)$ then it is appropriately dissimilar to the other classes $(c_j < c_i - \Delta c)$. The values of p and Δc are determined by trial and error. In this paper, the method used in [44] is employed to choose the desired spikes. A sequence of numbers starting from 10 to 100 ms with 10-ms time interval is generated. Then a number of firing times are extracted randomly from the sequence to assign each desired spike train corresponding to a class. In this situation, every two spikes have at least 10-ms interval. The parameter p is set based on the level of precision that the desired spikes should be learned. In this paper, when an actual output spike train reaches 90% of accuracy compared to its corresponding desired spike train the learning is stopped, so the learning parameter p is set 0.9. The parameter Δc should be higher than the maximum correlation between the desired spike trains related to different classes. Δc is set 0.45 to implement the proposed method.

After training, each testing pattern is applied to the network and the readout actual output spike train is calculated. The correlations between the actual output spike train and the desired spike trains corresponding to all classes are obtained. The input pattern is assigned to the class whose corresponding desired spike train has the maximum correlation value with the actual output spike train.

IV. Results

A. *Effect of Network Setups on the Learning Performance*

First, the effects of the different maximum allowable delays and the number of desired output spikes in each class on the performance of the learning method are explored. Then, the running time for the proposed method is reported. In the following simulation, the performance of the network is first evaluated on the Fisher IRIS data set. The IRIS data features are converted to spike times using population coding, as described in [23], where each feature value is encoded by M identically shaped overlapping Gaussian functions where M is set to 40. The IRIS data have four features for each pattern so there are $4 \times M = 160$ input spikes obtained which are then applied to 160 input synapses. The high number of input synapses increases the number of input spikes, and consequently reduces the length of silent windows inside a spatiotemporal input pattern and helps the neuron to fire at multiple desired times. In addition, there are nine extra input synapses with input spikes at fixed times for all patterns. The fixed times are the same as the times of desired spikes corresponding to all classes. These inputs act as bias inputs [21] and act as the reference start times in a multispike coding scheme. There are 360 hidden neurons in the hidden layer. The total time duration of the input spatiotemporal pattern is set to 100 ms, $T = 100$ ms.

1) Effect of Maximum Allowable Delays: Similar to [24], 50% of the IRIS data were selected randomly and used as training data and the remaining used for testing. The accuracy of the proposed method on the testing data reaches its highest value, 95.1%, when the maximum allowable delay D is 3 ms and there is a single readout neuron.

In Table II, the accuracies of the proposed method for different delays when there are three readout neurons (each corresponding to a class) in the network are shown. The accuracy of the method on the testing data reaches its maximum value when $D = 3$ ms (Table II). The accuracy of the proposed method on the testing data is increased from 95.1% to 95.7% when the number of readout neurons is increased from one to three when $D = 3$ ms. In Fig. 4, the accuracy of the learning algorithm on the training data is shown when a single readout neuron and three readout neurons are used. All these procedures are repeated independently for 40 different runs, and the mean value of the 40 results are reported. Different random initial weights and different random selections of the training and testing data are used for the different runs. When the number of readout neurons is increased, the number of learning parameters is also increased. Therefore, the readout neurons learn a lower number of training patterns compared to the situation where a single readout neuron is used, where the readout neuron should learn patterns related to all classes. Subsequently, they can learn the input patterns better compared to the situation that a single readout neuron is used. For higher values of maximum allowable delays, the cooperation between weight adjustment and delay adjustment is reduced and it leads to a lower accuracy. A higher delay adjustment causes a higher shift in the delayed effect of input spikes, and this higher shift might destroy previous weight training that was based on the previous value of the delay.

TABLE II

Effect of the Different Maximum Allowable Delays on IRIS Data Recognition. 50% of the Data Are Used as Training Data

Max-Delays (ms)	Training Accuracy (%)	Testing Accuracy (%)
1	99.8	95.3
3	99.8	95.7
4	99.6	95.7
5	99.6	95.3
7	99.6	94.6
10	98.9	94.5

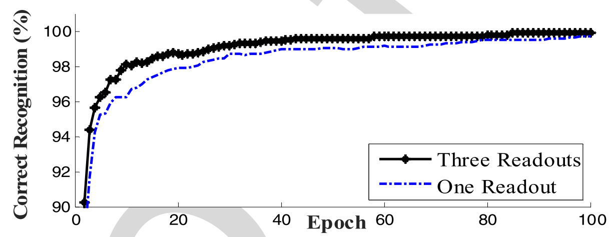

Fig. 4. Comparison of the learning method accuracy on the IRIS data training set when one and three readout neurons are used.

Synaptic delays at chemical synapses usually take values from 1 to 5 ms. The minimum value of a synaptic delay is 0.3 ms. Synaptic delay also can take a value higher than 5 ms [45]. Different researchers use different maximum values for range [1, 16] ms. The results in this section show that for this configuration, 3 ms is an optimal value for the maximum synaptic delay. In the following simulations, Max Delays are set to 3 ms.

2) Effect of the Number of Desired Spikes: In the following experiment, the accuracy of the proposed method is obtained for different numbers of desired spikes corresponding to each class (Table III).

The network reaches its maximum testing accuracy, 95.7%, when three desired spikes are used in each desired spike train.

TABLE III

EFFECT OF THE NUMBER OF DESIRED SPIKES ON LEARNING ACCURACY USING THE IRIS DATA SET WITH THREE READOUT NEURONS

# Desired Spikes for each Class	Training Accuracy (%)	Testing Accuracy (%)
4	96.60	91.5
3	99.80	95.7
2	99.97	95.5
1	99.90	95.1

Fig. 5. Recognition accuracy for different numbers of desired spikes.

A very high number of desired spikes in each desired spike train (i.e., for a desired spike train with 100-ms duration and 10-ms minimum interspike interval, the highest number of desired spikes is 10) reduce the performance of the learning method as this increases the complexity of the learning task and the network should be trained to fire at a higher number of desired instances with a limited number of learning parameters. For instance, the testing accuracy of the proposed method is reduced from 95.7% to 81% when the number of desired spikes is increased from 3 to 7 (Fig. 5).

The time distances between desired spikes of different classes are reduced when there is a high increase in the numbers of desired spikes. Therefore, a small deviation in the times of output spikes can cause a switching from one class to the other one and reduces the accuracy. On the other hand, a lower number of desired spikes reduce the complexity of the learning task, therefore the training accuracy will be increased. However, a very low number of desired spikes lead to a low testing accuracy. For example, when the number of desired spikes is reduced from three to one, the testing accuracy is reduced from 95.7% to 95.1%. It shows that a single spike cannot capture enough information from training data, and consequently, it reduces the testing accuracy despite of a comparably high training accuracy of 99.9%. Moreover, the distributions of spikes in the spatiotemporal input patterns compared to desired spikes also affect the accuracy and the relation between the number of desired spikes, and the accuracy is not a simple linear function (Fig. 5).

3) Evaluation of the Running Time: MATLAB simulations were carried out on a quad core PC with 3 GHz and 16 GB of RAM. The running times required for each learning epoch of the proposed method are reported in Table IV. The running time related to a learning epoch is measured 10 times, and the mean value is reported for each number of input synapses. The running time is increased by increasing the maximum allowable delays D. For instance, the method needs 5.2 s to execute a learning epoch when $D = 1$ ms. However, the running time is increased to 15.9 s when D is increased to 7 ms. Because, at each time step, the learning algorithm should check the events at the previous time steps depending on the delays. A higher number of previous time steps should be considered for a higher value of delays. Therefore, the computational complexity of the method and consequently the running time is increased when the delay is increased.

TABLE IV

EFFECT OF THE MAXIMUM ALLOWABLE DELAY (d) ON THE RUNNING TIME OF THE PROPOSED METHOD USING THE IRIS DATA SET

Max-Delays (ms)	Running Time (sec)	Max-Delays (ms)	Running Time (sec)
1	5.2	5	12.4
3	9.2	7	15.9

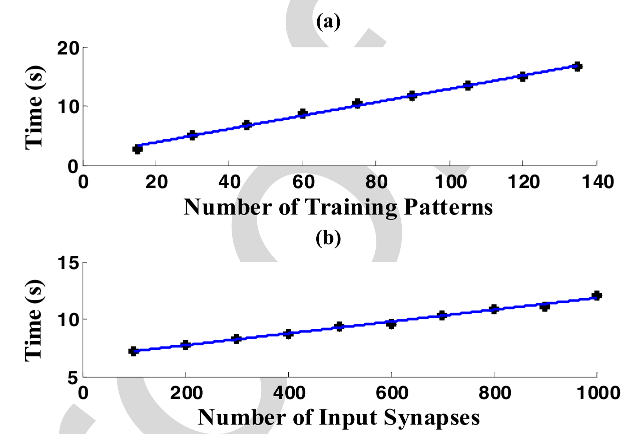

Fig. 6. Runing time of a learning epoch is increased linearly as a function of (a) number of training patterns and (b) number of input synapses.

The running times of a learning epoch of the proposed method are measured for different numbers of training patterns. The number of training patterns is increased from 15 to 135. IRIS data set is used to train the algorithm. Fig. 6(a) shows the relationship between the running times and the number of training patterns. The fit line shown in Fig. 6(a) is obtained by fitting the data points to a 1-D polynomial. The line is described by the equation $T(n) = 0.1128n + 1.593$. The time complexity of the process related to the equation is linear, i.e., it is $O(n)$ using the big O notation. It shows that the running time increases linearly with the number of training samples.

Random spatiotemporal input patterns with different numbers of inputs are used to analyze the complexity of the learning algorithm as a function of the number of input synapses. There are three classes similar to IRIS data in the randomly generated data. A spike train composed of three spikes is considered as desired spike train for each class like the desired spike used for IRIS data. The spike times in each input spatiotemporal pattern are generated by a uniform distribution. The values of spike times are extracted randomly from (0, 100) interval. The number of input synapses is changed from 100 to 1000, and an input spike is considered for each input synapse. Then, the running time for each

TABLE V
COMPARISON WITH THE MULTILAYER SNN PROPOSED IN [28] ON THE IRIS DATA SET

Method	Training Accuracy (%)	Testing Accuracy (%)
Sporea et. al.[28]	*96*	*94*
The proposed method	*99.3*	*95.8*

TABLE VI
COMPARISON WITH OTHER METHODS ON THE IRIS DATA SET

Method	Testing Accuracy (%)
Spiking Methods	
RBF [50]	*92.6*
SWAT [51]	*95.3*
SpikeProp [24]	*95*
QuickProp [23]	*92.3*
RProp [23]	*93.2*
RBF [53]	*89*
SNN (Bako) [52]	*83.4*
Proposed Method	*95.7*
Non-Spiking Methods	
K-Means [50]	*88.6*
SOM [50]	*85.33*
Matlab BP [51]	*95.5*
Matlab LM [51]	*95.7*
TEST [55]	*91.7*

TABLE VII
COMPARISON WITH OTHER METHODS ON THE WBCD DATA SET

Method	Testing Accuracy (%)
Spiking Methods	
SWAT [51]	*95.3*
SpikeProp [24]	*97*
SNN (Bako) [52]	*89.5*
Proposed Method	*96.4*
Non-Spiking Methods	
MATLAB Autoencoder	*96.2*
Matlab BP [51]	*96.3*
Matlab LM [51]	*96.7*
DBN [54]	*96.8*

learning epoch is calculated to analyze the complexity of the learning method. In this experiment, there are a fixed number of 75 training patterns. Fig. 6(b) shows the evolution of the running time in terms of the number of input synapses. In addition, a line fit with the obtained data points is plotted. The dependence between running time and the number of inputs indicates a linear time complexity, i.e., O(n).

B. Comparison With State-of-the-Art Methods

In the following simulation, first the proposed method is compared with the method proposed by Sporea and Grüning [28]. In this case, 75% of the total IRIS data for each class are considered as a training set and the remaining 25% are used for testing, as in [28]. The results are shown in Table V. The accuracy of the proposed method on the training is 99% which is higher than the method proposed in [28], 96%. The proposed method also achieved a higher testing accuracy of 96% (compared to 94% achieved by [28]).

Similar to the biologically plausible structure used in [18], each of the 169 input neurons is connected randomly to a limited number of neurons (40 neurons) in the hidden layer which consists of a population of 360 neurons. There are no subconnections, and every two neurons in two subsequent layers are connected by a single connection similar to the biologically plausible neural network in Izhikevich's work [18]. The proposed learning algorithm is designed to manage the training of a large number of SNs by local events such as spike trace which takes place at the location of each synapsis. There are three output neurons in the output layer and all the hidden neurons are connected to the three output neurons. The network proposed in [28] uses the timing of a single spike of an input neuron for each feature. The four input neurons are fully connected to ten neurons in the hidden layer. Every two neurons in two subsequent layers are connected by 12 subconnections with different delays from 1 to 12 ms. All the neurons in the hidden layer are fully connected to an output neuron. The performance of the method in [28] on the IRIS data is shown in Table V.

In order to compare the accuracy of the proposed method with that achieved by other existing methods, 50% of the data samples from the IRIS data set are selected randomly to construct training data and the remaining 50% are used for testing. The testing results are summarized in Table VI. The accuracies of the proposed method on the training and testing data are 99.7% and 95.7%, respectively. The testing accuracy of the proposed method, 95.7%, is comparable with the best result achieved for the state-of-the-art methods on IRIS data set. The proposed method has a high training accuracy, 99.7%.

The proposed method converges for all trials because it does not have the silent neuron problem. It has remote supervised spikes. In addition, it solves the problem of silent windows in a spatiotemporal input pattern by delay learning. A silent window can prevent generation of desired spikes and consequently it can cause learning convergence problem. These characteristics of the proposed method make it appropriate for learning multiple spikes. The accuracies of the proposed method are calculated for all trials, and there are not any rejected results. In contrast, the convergence rate of SpikeProp is investigated in [24] and as it has a problem with silent neurons it cannot converge for all trials, and as a result, those trials with low accuracies are removed from the reported results [24].

The Breast Cancer Wisconsin (Diagnostic) data set (WBCD) from the UCI machine learning repository is used as the second data set to evaluate the proposed method and to compare it with the other state-of-the-art methods, as shown in Table VII. WBCD contains 699 samples. The samples belong to two different classes (malignant and benign categories) where 458 samples are from the first category and 241 samples are from the second category. A total of 120 samples are selected

TABLE VIII

PERFORMANCE COMPARISON WITH SRESN AND GPSNN ON THE BUPA LIVER DISORDERS DATA SET

Method	Testing (Training) (%)
SRESN [46]	*59.7 (60.4)*
GPSNN [47]	*59.8 (61.18)*
Proposed Method	*61.8(69.9)*

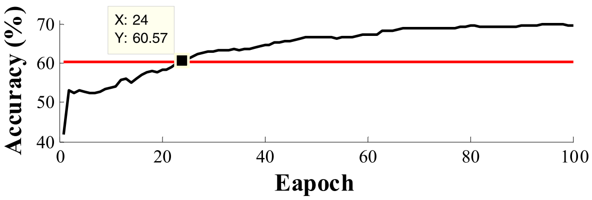

Fig. 7. Evolution of the accuracy of the proposed method over different learning epochs on BUPA liver disorders data. It needs 24 learning epochs to pass the accuracy level of 60%. SRESN [46] needs 715 epochs to reach about to the same level of accuracy.

TABLE IX

COMPARISON WITH SRESN ON DIFFERENT DATA SETS

Data Set	Testing (Training) (%)	Max # epochs	# LP [c]
	Proposed Method		
Pima diabetes [a]	*70.6 (72.1)*	*100*	*14640-1440*
BUPA	*61.8 (69.9)*	*100*	*9840-1440*
Ionosphere [b]	*90.5 (96.0)*	*100*	*54640-1440*
Iris	*95.7 (99.8)*	*100*	*6760-1440*
WBCD	*96.4 (98.2)*	*100*	*14640-1440*
	SRESN [46]		
Pima diabetes [a]	*69.9 (70.5)*	*254*	*486-756*
BUPA	*59.7 (60.4)*	*715*	*216-324*
Ionosphere [b]	*88.6(91.9)*	*1018*	*3264-4692*
Iris	*97.3(96.9)*	*102*	*120-200*
WBCD	*97.2(97.7)*	*306*	*432-648*

[a] Pima diabetes data from the UCI machine learning repository contains768 samples in which 500 and 268 samples are in two classes.
[b] Ionosphere data from the UCI machine learning repository contains 351 samples in which 225 and 126 samples are in two classes.
[c] # LP: Number of Learning parameters

randomly from each category to construct the training set, and the remaining data is used for testing. The proposed method has an accuracy comparable with the best accuracy achieved by the other state-of-the-art methods (Table VII).

One advantage of SNNs is that they use spikes to communicate between neurons. However, in the classical neural networks, real values are used to transfer data between neurons. Each spike can be encoded by a binary bit; however, a real value needs a high number of bits to be transferred between neurons depending on the precision that is required for the values. As shown in Tables VI and VII, the proposed method using spikes for communication between neurons and can achieve better or comparable accuracies with the state-of-the-art rate-based models including deep belief network (DBN) and autoencoders.

One more data set which is used to evaluate the proposed method is the BUPA liver disorders data from the UCI machine learning repository. There are 345 samples in this data set in which 145 samples are from the first class and 200 samples are from the second class. A total of 70 data samples are selected randomly from each class to construct the training set, and the remaining data is used for testing. Each sample has six attributes. The performance of the proposed method is shown in Table VIII. The testing accuracy of the proposed method ~~is higher than SRESN [46] and GPSNN [47]. SRESN [46]~~ uses a 30-2 architecture, and the proposed method uses a 246-360-2 architecture where there are 246 input neurons, 360 hidden neurons, and two output neurons. The evolution of the training accuracy of the proposed method over different learning epochs is shown in Fig. 7. The proposed method needs 24 learning epochs to pass the training accuracy of 60.4%; however, SRESN [46] needs 715 learning epochs to reach the same accuracy level. The proposed method can reach the accuracy level of 66.9% in less than 100 epochs.

The performance of the proposed method on different data sets is compared with SRESN [46] in Table IX. The number of learning parameters in SRESN [46] is lower than that of the parameters in the proposed method (see Table IX). A lower number of learning parameters can reduce the simulation time required for each learning epoch. However, the proposed method achieved high accuracies in a lower number of learning epochs compared to the method with a single layer of learning neurons on Pima diabetes, BUPA liver disorder, and ionosphere data sets. The proposed learning method achieves this improvement through appropriate interaction between different layers of SNs in a multilayer structure.

V. CONCLUSION

This paper proposed a BPSL for multilayer SNNs. It uses the precise timing of multiple spikes, which is a biologically plausible information coding scheme. The learning parameters of neurons in the hidden layer and output layer are learned in parallel using STDP, anti-STDP, and delay learning.

The simulation results show that the proposed method has improved the performance of the first fully supervised algorithm that learns multiple spikes in all layers proposed in [28].The improvement of the proposed method can be attributed to a number of properties of the proposed method. First, it has used the firing times of spikes fired by the hidden neurons to train the weights of the hidden neurons unlike the method in [28] where the firing time of hidden neurons is not considered and the weights of a hidden neuron are adjusted by the same values irrespective of the neuron firing at the desired times or not firing at all. In the proposed method, weight learning, based on the firing times of the hidden neurons, helps adjust the weights appropriately and prevents unnecessary weight adjustments. Another property of the proposed method is the appropriate use of the EPSP and the IPSP produced by the hidden excitatory and inhibitory neurons to effectively adjust their weights, unlike the approach in [28] where equal weight updates are applied to both excitatory and inhibitory neurons, which can reduce the learning performance. Another property of the proposed method that improves its performance compared to the learning method in [28] is the appropriate consideration of the effect of delays on the weight learning. It was shown that the delay after a hidden neuron has an essential effect on the output of the spiking network, hence it should be considered during the training of the weights of the hidden neuron. For example, an excitatory hidden neuron should fire earlier than a desired output spike depending on the delay after the hidden neuron, as described in ~~the previous sections~~. The produced PSP by the fired hidden spike is shifted to the desired time by the delay. The effect of the delay on the weight adjustments of hidden neurons is not considered in [28], and it was shown that this resulted in a lower accuracy compared to the proposed method on the IRIS data set.

The performance of the proposed method was also compared with other algorithms on different data sets. The results showed that the proposed method can achieve a higher accuracy compared to a single-layer SNN. In addition, the method has comparable accuracy with the best result achieved by state-of-the-art rate-based neural models including autoencoders and DBNs.

The results also showed that a very high number of desired spikes can reduce the accuracy of the method by increasing the complexity of the learning task, and a very low number of desired spikes cannot capture all the temporal information of input data. Although the delay learning increases the complexity of the learning method and consequently the running time, it was shown that delays can increase the learning performance of the proposed method. In addition, delays are a biologically plausible property of SNNs. Another property of the proposed method is its multilayer structure that increases the computational cost of each learning epoch. However, the results showed that it can also reduce the number of learning epochs and can improve its accuracy compared to the similar multilayer spiking network proposed by Sporea and Grüning [28]. The ablity of the proposed method to effectively learn multiple desired spikes suggests that this approach may be suitable for neuroprosthetic applications.

In a biologically plausible neuron model, the output of a neuron depends not only on synaptic inputs, but also on the internal dynamics of the neuron [48]. Therefore, a potential direction for future work is to incorporate the neuron internal dynamics in the proposed method, additionally to the effect of the synaptic weight and delays, which may lead to a new learning algorithm with potentially higher performance. For instance, Zhang *et al.* [49] have proposed a dynamic firing threshold to make the spiking network learning robust to noise. A similar method can be applied to the multilayer spiking network proposed in this paper to further improve its performance.

It is possible to extend the learning algorithm to more layers (deep SNNs). However, more layers may reduce the effect of training of earlier layers on the network output. Designing effective learning methods for deep spiking networks will be investigated in the future work.

REFERENCES

[1] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biol. Cybern.*, vol. 98, no. 6, pp. 459–478, 2008.

[2] V. Ramaswamy and A. Banerjee, "Connectomic constraints on computation in feedforward networks of spiking neurons," *J. Comput. Neurosci.*, vol. 37, no. 2, pp. 209–228, 2014.

[3] P. A. Cariani, "Temporal codes and computations for sensory representation and scene analysis," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1100–1111, Sep. 2004.

[4] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, no. 6535, pp. 33–36, 1995.

[5] A. Mohemmed and S. Schliebs, "Training spiking neural networks to associate spatio-temporal input-output spike patterns," *Neurocomputing*, vol. 107, pp. 3–10, May 2013.

[6] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," in *Handbook Natural Computing Anonymous*. Springer, 2012, pp. 335–376.

[7] S. B. Laughlin, R. R. de Ruyter van Steveninck, and J. C. Anderson, "The metabolic cost of neural information," *Nature Neurosci.*, vol. 1, no. 1, pp. 36–41, 1998.

[8] S. B. Laughlin, "Energy as a constraint on the coding and processing of sensory information," *Current Opinion Neurobiol.*, vol. 11, no. 4, pp. 475–480, 2001.

[9] J. Hu, H. Tang, K. C. Tan, H. Li, and L. Shi, "A spike-timing-based integrated model for pattern recognition," *Neural Comput.*, vol. 25, no. 2, pp. 450–472, 2013.

[10] A. Kasiński and F. Ponulak, "Comparison of supervised learning methods for spike time coding in spiking neural networks," *Int. J. Appl. Math. Comput. Sci.*, vol. 16, no. 1, pp. 101–113, 2006.

[11] R. V. Florian, "The chronotron: A neuron that learns to fire temporally precise spike patterns," *PLoS ONE*, vol. 7, no. 8, p. e40233, 2012.

[12] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.

[13] D. T. Pham, M. S. Packianather, and E. Y. A. Charles, "Control chart pattern clustering using a new self-organizing spiking neural network," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 222, no. 10, pp. 1201–1211, 2008.

[14] R. Brette *et al.*, "Simulation of networks of spiking neurons: A review of tools and strategies," *J. Comput. Neurosci.*, vol. 23, no. 3, pp. 349–398, Jul. 2007.

[15] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[16] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952.

[17] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[18] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Comput.*, vol. 18, no. 2, pp. 245–282, 2006.

[19] W. Maass and A. Zador, "Computing and learning with dynamic synapses," *Pulsed Neural Netw.*, vol. 6, pp. 321–336, May 1999.

[20] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition," *Neural Netw.*, vol. 41, pp. 188–201, May 2013.

[21] S. M. Bohte, J. N. Kok, and H. L. Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, 2002.

[22] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the spikeprop algorithm," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2006, pp. 3970–3977.

[23] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for EEG classification and epilepsy and seizure detection," *Integr. Comput.-Aided Eng.*, vol. 14, no. 3, pp. 187–212, 2007.

[24] S. B. Shrestha and Q. Song, "Adaptive learning rate of spikeprop based on weight convergence analysis," *Neural Netw.*, vol. 63, pp. 185–198, Mar. 2015.

[25] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, 2010.

[26] A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature Neurosci.*, vol. 2, no. 11, pp. 947–957, 1999.

[27] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Netw.*, vol. 43, pp. 99–113, Jul. 2013.

[28] I. Sporea and A. Grüning, "Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 25, no. 2, pp. 473–509, 2013.

[29] R. Memmesheimer, R. Rubin, B. P. Ölveczky, and H. Sompolinsky, "Learning precisely timed spikes," *Neuron*, vol. 82, no. 4, pp. 925–938, 2014.

[30] S. Hong, S. Ratté, S. A. Prescott, and E. De Schutter, "Single neuron firing properties impact correlation-based population coding," *J. Neurosci.*, vol. 32, no. 4, pp. 1413–1428, 2012.

[31] J. L. Rosselló, V. Canals, A. Oliver, and A. Morro, "Studying the role of synchronized and chaotic spiking neural ensembles in neural information processing," *Int. J. Neural Syst.*, vol. 24, no. 5, p. 1430003, 2014.

[32] J. P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning," *Neural Comput.*, vol. 18, no. 6, pp. 1318–1348, 2006.

[33] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "SPAN: Spike pattern association neuron for learning spatio-temporal spike patterns," *Int. J. Neural Syst.*, vol. 22, no. 4, p. 1250012, 2012.

[34] Y. Xu, X. Zeng, and S. Zhong, "A new supervised learning algorithm for spiking neurons," *Neural Comput.*, vol. 25, no. 6, pp. 1472–1511, 2013.

[35] A. Taherkhani, A. Belatreche, Y. Li, and L. Maguire, "A new biologically plausible supervised learning method for spiking neurons," in *Proc. ESANN*, Bruges, Belgium, 2014, pp. 1–6.

[36] M. Zhang *et al.*, "EMPD: An efficient membrane potential driven supervised learning algorithm for spiking neurons," *IEEE Trans. Cogn. Develop. Syst.*, ~~to be published~~.

[37] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural Netw.*, vol. 22, no. 10, pp. 1419–1431, 2009.

[38] O. Booij and H. tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Inf. Process. Lett.*, vol. 95, no. 6, pp. 552–558, 2005.

[39] H. Adeli and S. L. Hung, "An adaptive conjugate gradient learning algorithm for efficient training of neural networks," *Appl. Math. Comput.*, vol. 62, no. 1, pp. 81–102, 1994.

[40] S. L. Hung and H. Adeli, "Object-oriented backpropagation and its application to structural design," *Neurocomputing*, vol. 6, no. 1, pp. 45–55, 1994.

[41] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "DL-ReSuMe: A delay learning-based remote supervised method for spiking neurons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3137–3149, Dec. 2015.

[42] B. Glackin, J. A. Wall, T. M. McGinnity, L. P. Maguire, and L. J. McDaid, "A spiking neural network model of the medial superior olive using spike timing dependent plasticity for sound localization," *Frontiers Comput. Neurosci.*, vol. 4, p. 18, Aug. 2010.

[43] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "EDL: An extended delay learning based remote supervised method for spiking neurons," in *Neural Information Processing*, S. Arik, Ed. Springer, 2015, pp. 190–197.

[44] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "Multi-DL-ReSuMe: Multiple neurons delay learning remote supervised method," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Killarney, Ireland, Jul. 2015, pp. 1–7.

[45] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, and A. J. Hudspeth, *Principles of Neural Science*. New York, NY, USA: McGraw-Hill, 2004.

[46] S. Dora, K. Subramanian, S. Suresh, and N. Sundararajan, "Development of a self-regulating evolving spiking neural network for classification problem," *Neurocomputing*, vol. 171, pp. 1216–1229, Jan. 2016.

[47] S. Dora, S. Sundaram, and N. Sundararajan, "A two stage learning algorithm for a growing-pruning spiking neural network for pattern classification problems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–7.

[48] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[49] M. Zhang, H. Qu, X. Xie, and J. Kurths, "Supervised learning in spiking neural networks with noise-threshold," *Neurocomputing*, vol. 219, pp. 333–349, Jan. 2017.

[50] S. M. Bohte, H. La Poutré, and J. N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 426–435, Mar. 2002.

[51] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A spiking neural network training algorithm for classification problems," *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.

[52] L. Bako, "Real-time classification of datasets with hardware embedded neuromorphic neural networks," *Briefings Bioinf.*, vol. 11, no. 3, pp. 348–363, 2010.

[53] N. Gueorguieva, I. Valova, and G. Georgiev, "Learning and data clustering with an RBF-based spiking neuron network," *J. Experim. Theor. Artif. Intell.*, vol. 18, no. 1, pp. 73–86, 2006.

[54] M. Tanaka and M. Okutomi, "A novel inference of a restricted Boltzmann machine," in *Proc. 22nd Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2014, pp. 1526–1531.

[55] S. Yoon and S.-Y. Lee, "Training algorithm with incomplete data for feed-forward neural networks," *Neural Process Lett.*, vol. 10, no. 3, pp. 171–179, 1999.

Aboozar Taherkhani received the B.Sc. degree in electrical and electronic engineering from Shahid Beheshti University, Tehran, Iran, the M.Sc. degree in biomedical engineering from the Amirkabir University of Technology, Tehran, and the Ph.D. degree from Ulster University, Londonderry, U.K., in 2017.

He is currently a Research Fellow with the Computational Neuroscience and Cognitive Robotics Laboratory, Nottingham Trent University, Nottingham, U.K. His current research interests include artificial intelligence, spiking neural network, deep neural network, complex system theory, and nonlinear signal processing.

Ammar Belatreche (M'09) received the Ph.D. degree in computer science from Ulster University, Londonderry, U.K.

He was a Lecturer in computer science with the School of Computing and Intelligent Systems, Ulster University. He is currently a Senior Lecturer in computer science with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. His current research interests include bioinspired adaptive systems, machine learning, pattern recognition, data analytics, capital market engineering, and image processing and understanding.

Dr. Belatreche is a fellow of the Higher Education Academy, an Associate Editor of *Neurocomputing*, and has served as a Program Committee Member and a reviewer for several international conferences and journals.

Yuhua Li (SM'11) received the Ph.D. degree in general engineering from the University of Leicester, Leicester, U.K.

He was a Senior Research Fellow with Manchester Metropolitan University, Manchester, U.K., and a Research Associate with the University of Manchester, Manchester, from 2000 to 2005. He was a Lecturer with the School of Computing and Intelligent Systems, Ulster University, Londonderry, U.K., from 2005 to 2014, and with the School of Computing, Science and Engineering, University of Salford, Greater Manchester, U.K, from 2014 to 2017. He is currently a Senior Lecturer with the School of Computer Science and Informatics, Cardiff University, Cardiff, U.K. His current research interests include machine learning, pattern recognition, data science, knowledge-based systems, and condition monitoring and fault diagnosis.

Liam P. Maguire (M'15) received the M.Eng. and Ph.D. degrees in electrical and electronic engineering from the Queen's University of Belfast, Belfast, U.K., in 1988 and 1991, respectively.

He is currently a Professor and the Executive Dean of the Faculty of Computing, Engineering and the Built Environment, Ulster University, Londonderry, U.K. He has authored over 250 research papers, including 80 journal papers. He has an established track record of securing research funding and has also supervised 18 Ph.D. and 3 M.Phil. students to completion. His current research interests include cognitive data analytics, the development of new bioinspired intelligent systems, and the application of existing intelligent techniques.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ:1 = Please provide the appropriate reference number for McCulloch and Pitts (1943).
AQ:2 = Please provide the expansion for the acronyms “ReSuMe, EDL, SRESN, and GPSNN.”
AQ:3 = Please provide the section numbers to replace the phrase “the previous sections.”
AQ:4 = Please provide the publisher location for refs. [6] and [43].
AQ:5 = Please provide the volume no., issue no., page range, month, and year for ref. [36].

A Supervised Learning Algorithm for Learning Precise Timing of Multiple Spikes in Multilayer Spiking Neural Networks

Aboozar Taherkhani, Ammar Belatreche, *Member, IEEE*, Yuhua Li, *Senior Member, IEEE*, and Liam P. Maguire, *Member, IEEE*

***Abstract*— There is a biological evidence to prove information is coded through precise timing of spikes in the brain. However, training a population of spiking neurons in a multilayer network to fire at multiple precise times remains a challenging task. Delay learning and the effect of a delay on weight learning in a spiking neural network (SNN) have not been investigated thoroughly. This paper proposes a novel biologically plausible supervised learning algorithm for learning precisely timed multiple spikes in a multilayer SNNs. Based on the spike-timing-dependent plasticity learning rule, the proposed learning method trains an SNN through the synergy between weight and delay learning. The weights of the hidden and output neurons are adjusted in parallel. The proposed learning method captures the contribution of synaptic delays to the learning of synaptic weights. Interaction between different layers of the network is realized through biofeedback signals sent by the output neurons. The trained SNN is used for the classification of spatiotemporal input patterns. The proposed learning method also trains the spiking network not to fire spikes at undesired times which contribute to misclassification. Experimental evaluation on benchmark data sets from the UCI machine learning repository shows that the proposed method has comparable results with classical rate-based methods such as deep belief network and the autoencoder models. Moreover, the proposed method can achieve higher classification accuracies than single layer and a similar multilayer SNN.**

***Index Terms*— Multilayer neural network, spiking neural network (SNN), supervised learning, synaptic delay.**

I. Introduction

SPIKE-timing-dependent plasticity (STDP) plays a prominent role in learning biological neurons, and it represents one form of synaptic plasticity which underpins synaptic weight changes based on the precise times of pre and postsynaptic spikes [1]. STDP highlights the important role of precise spike times in information processing in the brain [2]. In addition, the rapid sensory processing observed in the visual, auditory, and olfactory systems supports the assumption that information is encoded in the precise timing of the spikes [3]–[5]. Moreover, using precise timing of spikes results in a higher information encoding capacity compared with rate-based coding [6], and it can also convey the information related to rate of spikes in a multispike coding scheme [2]. Furthermore, as neural activity is metabolically expensive, the high number of spikes involved in rate coding scheme demands a significant amount of energy and resources [7], [8]. Despite the existing evidence supporting information encoding using the precise timing of spikes, the exact neuronal mechanisms that underlie learning to fire at precise times are still not clear and remain as one of the challenging problems in the field of spiking neural networks (SNNs) [2], [9]–[11].

In this paper, a novel supervised learning algorithm inspired by STDP is proposed to train an SNN to fire multiple spikes at precise desired times. Local synaptic biochemical events, produced by incoming spikes, are used to adjust weights and delays appropriately. In addition, neurons in the output and hidden layers interact with each other through a biofeedback signal sent by the output neurons to train the network. The main novelty of the proposed method consists in: 1) capturing the effect of synaptic delays on the learning of neuronal connection weights in an SNN, which has not been considered in previous works and 2) learning the spiking network synaptic delays. In addition, the proposed approach introduces an additional training mechanism to prevent the occurrence of undesired spikes which contribute to the misclassification of spatiotemporal input patterns. The proposed approach is validated using benchmark classification data sets and is compared against both spiking and rate-based neural models including state-of-the-art deep learning and autoencoder models. The experimental results show an improvement in learning accuracy over existing competitive SNN architectures and comparable performance to state-of-the-art rate-based neural models.

The remainder of this paper is structured as follows. A brief review of background and related work on SNNs is presented in Section II. Section III introduces the proposed method

Manuscript received April 4, 2017; revised October 27, 2017 and January 11, 2018; accepted January 16, 2018. This work was supported by the Ulster University's Vice Chancellor's Research Scholarship Scheme.
(Corresponding author: Aboozar Taherkhani.)

A. Taherkhani is with the Computational Neuroscience and Cognitive Robotics Laboratory, Nottingham Trent University, Nottingham NG1 4FQ, U.K. (e-mail: aboozar.taherkhani@ntu.ac.uk).

A. Belatreche is with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: ammar.belatreche@northumbria.ac.uk).

Y. Li is with the School of Computer Science and Informatics, Cardiff University, Cardiff CF10 3AT, U.K. (e-mail: yl7_99@yahoo.com).

L. P. Maguire is with the Faculty of Computing and Engineering, Ulster University, Londonderry BT48 7JL, U.K. (e-mail: lp.maguire@ulster.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2797801

in detail. The simulation results are then provided in Section IV. Finally, Section V concludes this paper.

II. Background and Related Work

Different artificial neural networks (ANNs) have been devised based on the working principle of their biological AQ:1 counterparts. McCulloch and Pitts (1943) developed the first ANN where the neuron model is a logic unit which can be in an active or inactive (binary) mode depending on the weighted sum of their binary inputs. Later, a continuous transfer function (e.g., sigmoid function) is applied to the weighted sum of continuous inputs to generate continuous output [12]. The continuous values represent the biological neuron spiking rates. ANNs are inspired by the biological nervous system and are successfully used in various applications. However, their high abstraction compared to their biological counterparts [13] and their inability to capture the complex temporal dynamics of biological neurons have resulted in a new area of ANNs where the focus is placed on more biologically plausible neuronal models known as SNNs. Thanks to their ability to capture the rich dynamics of biological neurons and to represent and integrate different information dimensions such as time, frequency, and phase, SNNs offer a promising computing paradigm and are potentially capable of modeling complex information processing in the brain [14]–[20].

In 1952, Hodgkin and Huxley [16] built a 4-D detailed conductance-based neuron model which can reproduce electrophysiological measurements to a high degree of accuracy. However, because of its intrinsic computational complexity, this model has a high computational cost. For this reason, simple phenomenological spiking neuron (SN) models are employed for simulating large-scale SNNs [15]. The leaky integrate-and-fire (LIF) model is a popular 1-D spiking neural model with low computational cost, but it offers relatively poor biological plausibility compared with the Hodgkin and Huxley model. Simple phenomenological SN models with low computational cost are highly popular for studies of neural coding, memory, and network dynamics [12].

The first supervised learning algorithms for multilayer SNNs using the precise timing of spikes could train only a single spike for each neuron. Bohte *et al.* [21] proposed the multilayer SNN called SpikeProp (inspired by the classical back-propagation algorithm) as one of the first supervised learning methods for feedforward multilayer SNNs. Backpropagation with momentum [22], QuickProp [22], resilient propagation [22], [23], and the SpikeProp based on adaptive learning rate [24] were proposed to improve the performance of SpikeProp. In all these methods, neurons in the input, output, and hidden layers can only fire a single spike.

Despite the capability of a single-spike learning method, single-spike coding schemes limit the diversity and capacity of information transmission in a network of SNs. In contrast, multiple spikes significantly increase the richness of the neural information representation [25], [26]. In addition, training a neuron to fire multiple spikes is more biologically plausible compared to single-spike learning methods [27], [28]. Temporal encoding through multiple spikes transfers important information which cannot be expressed by a single-spike coding scheme or a rate coding scheme. Although the exact mechanism of information coding in the brain is not clear, biological evidence shows that multiple spikes have a pivotal role in the brain. For instance, mapping between spatiotemporal spiking sensory inputs composed of spike trains to precise timing of spikes is an essential characteristic of neuronal circuits of the zebra finch brain to execute well-timed motor sequences [29]. In the mixed approaches proposed in [30] and [31], it is suggested that using both spike timing and spike rate increases processing speed. These methods use a combination of both correlated and uncorrelated spiking signals. So, there is useful information in the spike rate that cannot be captured by the precise timing of single spikes. Encoding information in the precise timing of multiple spikes which are used in this paper can capture not only the information in the spike rate but also the information in inter spike intervals.

Pfister *et al.* [32] designed a supervised learning algorithm for a single SN which updates synaptic weights to increase the likelihood of postsynaptic firing at several desired times. The algorithm is designed to train only a single neuron; however, it can train the neuron to fire multiple desired spikes. ReSuMe [25], spike pattern association neuron [33], AQ:2 perceptron-based SN learning rule [34], biologically plausible supervised learning method (BPSL) [35], and efficient membrane potential-driven supervised learning method [36] are other examples of learning methods that can train a single neuron to fire multiple desired spikes. Multispike learning methods focus on a single neuron or a single layer of neurons. It is difficult to design a multilayer SNN to fire multiple desired spikes because the complexity of the learning task is increased [27], [37]. In this situation, the learning algorithm should control several neurons to generate different desired spikes. However, a real biological nervous system is composed of a large number of interconnected neurons [27], [28], [37].

A multilayer neural network has a higher information processing ability than a single layer of neurons. Sporea and Grüning [28] have shown that a multilayer SNN can perform a nonlinearly separable logical operation; however, the task cannot be accomplished without the hidden layer neurons.

Ghosh-Dastidar and Adeli [37] and Booij and tat Nguyen [38] extended the multilayer SpikeProp [21] to allow each neuron in the input and hidden layers to fire multiple spikes. However, each output neuron can fire only a single spike. Xu *et al.* [27] proposed the first supervised learning method based on the classical error back-propagation method that can train all the neurons in a multilayer SNN to fire multiple spikes. Gradient learning methods suffer from various known problems which can lead to learning failure such as sudden jumps (called surge) or discontinuities in the error function [24]. The problem becomes more severe when the output neurons are trained to fire more than a single spike. In addition, the construction of an error function becomes difficult when multiple desired spikes should be learned as the number of actual output spikes may differ from the number of desired spikes in each learning epoch [27]. After investigation of the gradient-based methods in [23], [39], and [40], it is concluded that the application of STDP is worth further investigation to implement a more biologically plausible learning algorithm for multilayer SNNs [37].

Sporea and Grüning [28] have used STDP and anti-STDP to devise the first biologically plausible supervised learning algorithm for the classification of real-world data by a multilayer SNN in which each neuron in the input, hidden, and output layers can fire multiple spikes. The authors did not consider the spikes fired by hidden neurons when training the hidden neurons parameters. However, in a biological neuron, STDP usually works on the pre- and postsynaptic spikes of the neuron. In addition, the output spikes of the hidden neurons have significant effects on a training task in a multilayer SNN. Another drawback of this method [28] is that it has used the same learning adjustment method for inhibitory and excitatory neurons in hidden layers. However, inhibitory and excitatory neurons have different effects in a network by generating positive and negative postsynaptic potentials (PSPs). In this paper, a method is proposed to use spikes fired by hidden neurons during learning, and excitatory and inhibitory neurons are trained appropriately.

Delays of spike propagation are an important characteristic of real biological neural systems, and they have a significant effect on the information processing ability of the nervous system [18], [41], [42]. In EDL [43], an extended delay learning-based remote supervised method for SNs, and in DL-ReSuMe [41], a delay learning-based remote supervised method for SNs, investigated the viability of adjusting the neuron synaptic weights and delays for training a single SN to map a given spatiotemporal input pattern into a desired output spike train. STDP and anti-STDP were used to adjust the synaptic weights, and a delay shift approach was used to adjust their delays. It is worth noting that constant synaptic delays have been employed in [28], hence neglecting the effect of a synaptic delay between a hidden neuron and an output neuron on the weight adjustment of the hidden neuron. It trains the hidden neuron to fire at the time of an output desired spike. However, the generated spike is shifted by the network synaptic delay and causes an error in the firing time of the output neuron. SpikeProp and its related gradient-based methods [21], [23], [37] have taken into account the effect of a delay between a hidden neuron and an output neuron on the input weight adjustment of the hidden neurons. However, the use of multiple connections with different delays after a hidden neuron causes each of the different delays to affect the adjustment of the hidden neuron weights in different and opposite directions. Because, different errors are propagated from an output neuron to a hidden neuron corresponding to the different subconnections between the two neurons. The different errors force the hidden neuron to fire at different times depending on the different delays related to the multiple connections, and it disturbs the learning procedure. This might be one reason for the huge sudden rise in learning error of SpikeProp, as reported in [24].

In this paper, a learning algorithm is proposed to train both weights and delays of a multilayer SNN to fire multiple desired spikes. In the proposed method, each neuron at input, hidden, and output layers can fire multiple spikes. Supervised training of SNs which fire multiple spikes in a multilayer SNN remains a challenge. Furthermore, the proposed approach trains the synaptic delays in the multilayer SNN and also takes into the effect of delays on weight adjustments which is not considered in [21]–[24] and [28]. In the proposed method, the effect of the delays between a hidden neuron and an output neuron is considered during weight adjustments of the hidden neuron. In addition, the proposed method trains the weights of the hidden neurons by using the spikes fired by hidden neurons during STDP and anti-STDP, which results in a more biologically plausible and a highly accurate learning. Moreover, different weight adjustment strategies are used to train excitatory and inhibitory hidden neurons based on the effect of the excitatory (positive) and inhibitory (negative) PSPs (EPSP and IPSP) produced by the trained hidden neurons. In Section II, the principle of the proposed method is described.

III. Materials and Methods

The aim of the proposed supervised learning algorithm is to train a multilayer SNN to map spatiotemporal input patterns to their corresponding desired spike trains which implements a classification of the spatiotemporal input patterns. The network is composed of an input, a hidden, and an output layer. An output neuron, called a readout neuron, is fully connected to the hidden neurons. A spatiotemporal input pattern is emitted by the neurons in the input layer. Each input neuron is randomly connected to a fraction number of hidden neurons as used in [18]. The LIF neuron model described in [41] is used. The proposed method trains the spiking network by adjusting the learning parameters of the hidden and output neurons in parallel.

A. Overview of the Proposed Learning Method

The proposed learning method aims to train the multilayer SNN to enable each readout (output) neuron to fire actual output spikes at desired times and to cancel out undesired output spikes. A remote supervising signal is considered for an output neuron similar to ReSuMe [25]. At the time of a desired spike where there are not any actual output spikes at the readout neuron, the network learning parameters are adjusted to increase the total PSP of the readout neuron to hit the threshold level and generate an actual output spike at the desired time by using biologically plausible local events. The output neuron does the following three activities in parallel at the desired spike time.

First, at the time of the desired spike, the output neuron sends back an instruction signal (biofeedback) that shows the time of desired spike to the hidden neurons. After receiving the instruction signal, an excitatory hidden neuron potentiates its weights based on STDP to fire an output spike (hidden spike) at a specific time interval before the desired time. The specific time interval is equal to the delay related to the connection between the excitatory hidden neuron and the output neuron. The effect of the generated hidden spike (i.e., the PSP generated by the hidden spike) is shifted to the desired spike time after the related delay between the hidden neuron and the output neuron. The potentiation of the excitatory hidden neuron weights is stopped when the hidden neuron firing rate reaches a certain value, because a biological neuron cannot fire with a limitless rate, and a refractory period will ensure an upper bound on the neuron firing rate. The excitatory hidden neuron weight potentiation at the time of a desired spike is also stopped when an actual spike is generated at the time of the desired spike by the output neuron. In addition, the feedback triggers an inhibitory hidden neuron to try to remove its output spikes fired a specific time interval before the desired time by using the long-term depression (LTD) of anti-STDP. The time interval is equal to the delay between the inhibitory hidden neuron and the readout neuron. The hidden neuron output spikes before the time interval affects the PSP of the readout neuron at the desired time, i.e., the hidden spikes generate delayed PSPs at the desired time. The reduction of the inhibitory hidden spikes helps the readout neuron to increase its total PSP at the desired time to hit the threshold level.

Second, similar to ReSuMe [25] the output neuron potentiates its weights that have a spike shortly before the desired time based on STDP to increase its PSP at the desired time to fire.

The third activity at the time of a desired spike where there are not any actual output spikes of the readout neuron is the adjustment of delays of the readout neuron to increase the PSP of the readout neuron at the desired time, based on EDL [43]. All the abovementioned activities are repeated at the time of other desired spikes in a multispike coding scheme.

At the time of an undesired output spike of the readout neuron (i.e., where there is an actual output spike and there are not any desired spikes), the learning algorithm should reduce the total PSP of the readout neuron at the time of the undesired output spike to remove it by applying the following three processes in parallel. First, the readout neuron sends a feedback to excitatory hidden neurons to instruct them to remove their output spikes. Each excitatory hidden neuron removes its spike fired at a precise time interval before the time of the undesired spike by using LTD based on anti-STDP and reduces its weights. The time interval for the hidden neuron is equal to the delay between the hidden neuron and the readout neuron. Consequently, the reduction of the excitatory hidden neuron weights can help the readout neuron to reduce its total PSP and to remove the undesired output spike. It is clear that the weight reduction should be applied to the excitatory neurons that have a number of output spikes. Therefore, the LTD is applied to the excitatory neurons when their firing rates are higher than a threshold rate. The threshold rate is set by trial and error. In addition, the feedback triggers each inhibitory hidden neuron to potentiate its weights based on the long-term potentiation of STDP. The weight potentiation increases inhibitory hidden spikes before a precise time interval (the time interval is equal to the delay between the hidden neuron and the readout neuron) before the undesired spike time to help the readout neuron to reduce its total PSP at the undesired output spike time. The second process is applied at the time of the undesired output spike and consists of a reduction of the readout neuron weights that have spikes at the undesired output spike time or shortly before it by using anti-STDP similar to ReSuMe [25]. The third process reduces the readout neuron total PSP at the time of the undesired spike by adjusting the delays of the readout neuron based on EDL [43].

The hidden layer spikes play an important role in the generation of the network output spikes (both at desired and undesired times). Generated spikes by different hidden neurons cooperatively increase the PSP of the output neuron at a desired time and help it to fire at the desired time. In addition, when the complexity of a learning task is increased by increasing the number of desired spikes and also by increasing the number of different training patterns for each class, it becomes difficult or impossible to train a single neuron to fire at all the desired times for all the training patterns. Different groups of hidden neurons can contribute in generating different desired spikes and cooperatively drive a readout neuron to fire at all the desired times for all the training patterns.

In Sections III-B and III-C, first the training rule of the output neurons is explained and then the training of the hidden neurons weights is described in detail.

B. Training the Output Neurons

The weights and delays of each output neuron are trained by EDL, as described in [43]. The delay adjustments in cooperation with the weight adjustments train an output neuron to increase its total PSP at a desired time to generate an actual output spike, and also the adjustments help the output neuron to reduce its PSP at undesired spike times and to remove undesired actual output spikes. The weights are trained by the following equation:

$$\frac{dw_{\mathrm{oh}}(t)}{dt} = \left[s_o^d(t) - s_o^a(t)\right]\left[a + \int_0^{+\infty} \Psi(s) s_h(t - d_{\mathrm{oh}} - s)\mathrm{d}s\right] \tag{1}$$

where w_{oh} and d_{oh} are the weight and delay related to the connection between the hth hidden neuron and the oth output neuron, respectively. $s_o^d(t)$ and $s_o^a(t)$ are desired and actual output spike trains of the oth output neuron, respectively. $s_h(t)$ is the spike train fired by hth hidden neuron. a is a non-Hebbian parameter that can speed up the learning. $\Psi(s)$ is a learning window similar to that of STDP and has an exponential function as described by

$$\Psi(s) = \begin{cases} Ae^{-s/\tau}, & s \geq 0 \\ 0, & s < 0 \end{cases} \tag{2}$$

where τ and A are the exponential decay time constant and the amplitude of the learning window, respectively.

$x_{\mathrm{oh}}(t)$, a local variable called spike trace, is used to train the delay related to the synapse that connect hth excitatory hidden neuron to oth output neuron. $x_{\mathrm{oh}}(t)$ is governed by

$$x_{\mathrm{oh}}(t) = \begin{cases} Ae^{-(t - t_h^f - \varepsilon_{\mathrm{oh}})/\tau}, & t_h^f < t < t_h^{f+1} \\ A, & t = t_h^f \end{cases} \tag{3}$$

where t_h^f is the firing time of the fth spike of the hth excitatory hidden neuron, τ is the time constant of the exponential function, $\varepsilon_{\mathrm{oh}}$ is the delay between the hth excitatory

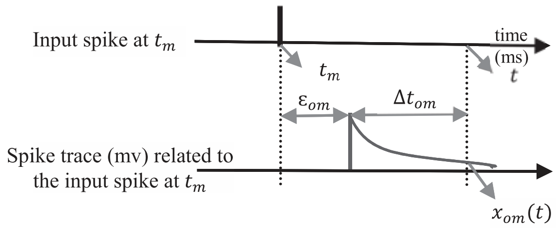

Fig. 1. Trace x_{om} related to input spike at t_m jumps to a maximum value after the delay $\varepsilon_{\mathrm{om}}$. Then it decays exponentially through time.

hidden neuron and the oth output neuron, and A is a constant value which are equal to their counterparts in (2). $x_{\mathrm{oh}}(t)$ is used to obtain appropriate value for delay adjustment. The adjustment $\Delta\varepsilon_{\mathrm{oh}}$ is calculated by (4) similar to EDL [43]

$$\Delta\varepsilon_{\mathrm{oh}}(t) = \begin{cases} +\Delta t_{\mathrm{om}}(t)(x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t))^4, & t = \hat{t}_o^f \\ -\Delta t_{\mathrm{om}}(t)(x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t))^4, & t = t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{4}$$

where $\hat{t}_o^f$ is the time of the fth desired spike, t_o^f is the time of the fth actual output spike of the oth output neuron, and $x_{\mathrm{om}}(t)$ is the maximum trace between the traces of the excitatory hidden neurons connected to the oth output neuron at the current time t. $x_{\mathrm{om}}(t)$ is corresponding to the connection between the mth excitatory hidden neuron (that has the closest spike before the current time t) and the oth output neuron. Δt_{om} is a delay shift which is necessary to be added to the delay between the mth excitatory hidden neuron and the oth output neuron to bring the effect of the closest spike fired by mth excitatory hidden neuron to the current time t. It is derived from (3) and calculated by

$$\Delta t_{\mathrm{om}} = t - t_m - \varepsilon_{\mathrm{om}} = -\tau_x \ln\left(x_{\mathrm{om}}(t)/A\right) \tag{5}$$

where t_m is the firing time of the mth excitatory hidden neuron before current time t. The mth excitatory hidden neuron has the closest spike before the current time t. It has the maximum trace at time $t x_{\mathrm{om}}(t)$ out of all excitatory input synapses of the oth output neuron. $x_{\mathrm{om}}(t)$ should be less than A, because the spike should occur before the current time. $\varepsilon_{\mathrm{om}}$ is the delay between the mth excitatory hidden neuron and the oth output neuron. Fig. 1 illustrates the relationship between the different parameters used in (5).

The delay adjustment in (4) tries to increase the total PSP of the oth output neuron at $t = \hat{t}_o^f$ and to reduce the total PSP at $t = t_o^f$. The delay increment in (4) shifts the positive PSPs generated by excitatory inputs to the desired times to generate an output spike. The delay reduction shifts the positive PSPs away from the actual output spikes times to remove undesired spikes. When an actual output spike is generated at the time of a desired spike, the positive delay adjustment cancels out the negative delay adjustment and the delays are stabilized. In (4), we have $[x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t)] \le 1$. The use of the fourth power in (4) reduces the amount of delay adjustment related to a far input spike. A far input spike corresponds to a low value of $[x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t)]$ and consequently a lower value of the fourth power of $[x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t)] \le 1$, and only the delays related to the close input spikes which have a high effect on the PSP is adjusted by a high value to prevent unnecessary change of the delays in the network.

The adjustment of delay between the hth inhibitory hidden neuron and the oth output neuron $\Delta\mu_{\mathrm{oh}}$ is governed by

$$\Delta\mu_{\mathrm{oh}}(t) = \begin{cases} -\Delta\bar{t}_{\mathrm{om}}(t)(\bar{x}_{\mathrm{oh}}(t)/(\bar{x}_{\mathrm{om}}(t))^4, & t = \hat{t}_o^f \\ +\Delta\bar{t}_{\mathrm{om}}(t)(\bar{x}_{\mathrm{oh}}(t)/\bar{x}_{\mathrm{om}}(t))^4, & t = t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{6}$$

where $\bar{x}_{\mathrm{oh}}(t)$ is the spike trace related to the connection between hth inhibitory hidden neuron and the oth output neuron. $\bar{x}_{\mathrm{om}}(t)$ is the maximum trace between the inhibitory hidden neurons that are connected to the oth output neuron. It should be less than A. $\Delta\bar{t}_{\mathrm{om}}(t)$ is calculated by putting $\bar{x}_{\mathrm{om}}(t)$ in (5). The decrement of delays in the first expression of (6) at the desired times shifts away the negative PSPs generated by inhibitory inputs (from the desired times) and increases the total PSP of the output neuron accordingly. This might increase the total PSP to hit the threshold level and generate an actual output at the desired times. The delay increment in the second expression relates to the inhibitory input spikes before the actual outputs shifts the negative PSP of the inhibitory inputs toward the actual output spikes to remove undesired output spikes. When an actual output spike is generated at the time of a desired spike, the delay decrement and increment in (6) are equal and the net adjustment becomes zero.

C. Training the Hidden Neurons

This section introduces the learning algorithms for both excitatory and inhibitory hidden neurons.

1) Weight Learning of Excitatory Hidden Neurons: The synaptic weight between the ith input neuron and the hth excitatory hidden neuron is denoted by w_{hi} and all the delays in the network are neglected in this stage. The synaptic weight adjustment is governed by

$$\Delta w_{\mathrm{hi}}(t) = \begin{cases} +\sum_o [\Psi(t-t_i)(1-\Psi(t-t_h)/A)](w_{\mathrm{oh}}/A), & t = \hat{t}_o^f \\ -\sum_o [\Psi(t-t_i)(\Psi(t-t_h)/A)](w_{\mathrm{oh}}/A), & t = t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{7}$$

where t_i is the last firing time of the ith input spike at or before the current time t. Equation (7) shows that the algorithm adjusts the weight at the time of the fth desired spike of the oth output neuron, $t = \hat{t}_o^f$, and at the time of the fth actual output spike of the oth output neuron, $t = t_o^f$. The sigma ($\sum$) collects the weight adjustment on all the output neurons. At the time of the desired spike, the weight is potentiated in proportion to the STDP time window ($\Psi(t - t_i)$) to generate hidden neuron spike at the desired time or shortly before it to increase the total PSP of the oth output neuron and help the output neuron to generate an actual output spike at the desired time (Fig. 2). Different hidden neurons correspond to different desired spikes, and they cooperatively force the output neuron to fire at all desired times.

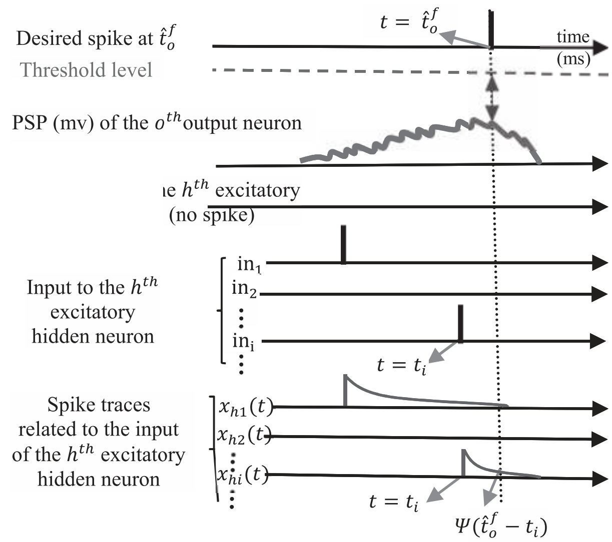

Fig. 2. Synaptic weight between ith input neuron and the hth excitatory hidden neuron w_{hi} is potentiated in proportion to the value of STDP time window $[\Psi(t - t_i)]$ at $t = \hat{t}_o^f$ to generate hidden spike at the desired time $t = \hat{t}_o^f$. The generated excitatory input will be fed to the oth output neuron, and it increases the total PSP of the neuron at the desired time.

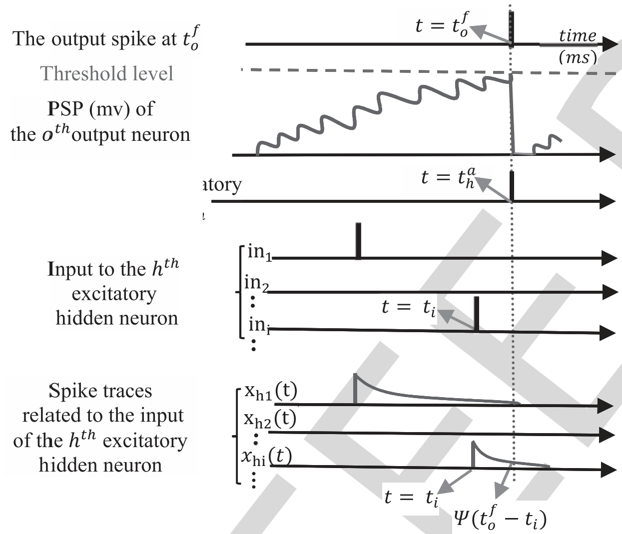

Fig. 3. w_{hi}, the synaptic weight between ith input neuron and the hth excitatory hidden neuron, is reduced in proportion to $\Psi(t - t_i)$, at $t = t_o^f$ (the time of the fth actual output spike of the oth output neuron). The reduction might lead to the cancelation of the hidden spike at t_h and consequently the reduction of the total PSP of the oth output neuron generated at $t = t_o^f$ and remove the actual output at $t = t_o^f$.

At the time of an actual output, $t = t_o^f$, $\Delta w_{\mathrm{hi}}(t)$ is reduced in proportion to the STDP time window $\Psi(t - t_i)$. It depends on the time difference of its input spike t_i, and the current time $t = t_o^f$, $(t_o^f - t_i)$. The reduction might lead to the cancellation of the hidden spike at t_h shortly before $t = t_o^f$ or at t_o^f, and consequently reduces the total PSP of the oth output neuron generated at $t = t_o^f$ and remove the actual output at $t = t_o^f$ (Fig. 3). When the actual output spikes at $t = t_o^f$, it becomes close to the desired spike at $t = \hat{t}_o^f$, the positive weight adjustment related to the desired spike cancels out the negative weight adjustment at the actual output. Consequently, the net weight adjustment becomes small.

The excitatory hidden neuron weight is adjusted based on the three spikes shown in Fig. 3 by (7). In a triplet-STDP, which is a more accurate model of synaptic plasticity in a biological neuron than a standard pair-based STDP [1], three spikes also affect a weight adjustment. A triplet-STDP described in [1] uses a single presynaptic and two postsynaptic spikes. There are different models for triplet-STDP [1].

The term $[(1 - \Psi(t - t_h)/A)]$ in (7) prevents the weight change of an excitatory hidden neuron that already has an actual output at the desired time, $t = \hat{t}_o^f$ as in this situation $\Psi(\hat{t}_o^f - t_h) = A$, consequently, $[(1 - \Psi(\hat{t}_o^f - t_h)/A) = 0]$. Therefore, the weight increment related to the hidden w_{hi} is 0, because the hidden neuron already has a spike at this desired time and it does not need more weight adjustment. Different hidden neurons contribute to firing of the output neuron at different desired times and cooperatively help the output neuron to fire at all the desired spikes in a multispike coding scheme. The term also causes a smaller increment of the weight w_{hi} that has output spike closely before the desired spike $[\Psi(\hat{t}_o^f - t_h) \cong A$, consequently, $(1 - \Psi(\hat{t}_o^f - t_h)/A) \cong 0]$. An unnecessary high adjustment might shift the hidden spike close to $\hat{t}_o^f$ beyond the desired time and reduce the total PSP of the oth output neuron at the desired time. In addition, the term $(1 - \Psi(t - t_h)/A)$ causes a comparatively high increment of w_{hi} when a hidden neuron does not have spike before $t = \hat{t}_o^f$ [because $(1 - \Psi(\hat{t}_o^f - t_h)/A) = 1$], or the actual output of the hth hidden neuron is far from the desired time at $t = \hat{t}_o^f$ $[(1 - \Psi(\hat{t}_o^f - t_h)/A) \cong 1]$. The high increment might force the hth hidden neuron to fire at the desired time $t = \hat{t}_o^f$, and consequently increase the total PSP of the oth output neuron at the desired times $t = \hat{t}_o^f$.

The term $[\Psi(t - t_h)/A]$ in (7) when $t = t_o^f$ prevents the reduction of w_{hi} if the hth excitatory hidden neuron does not have any actual output spikes before the actual output of the oth output neuron at $t = t_o^f$ $[(\Psi(t_o^f - t_h)/A) = 0]$. Because, w_{hi} does not have any roles in the generation of the output spike at $t = t_o^f$. If an excitatory hidden neuron has output spike before and close to an actual output spike at $t = t_o^f$, the term has comparatively a high value $[(\Psi(t_o^f - t_h)/A) \cong 1]$, and consequently, w_{hi} is adjusted with a higher value, because the excitatory hidden neuron has a strong contribution in the generation of the actual output spike at $t = t_o^f$ and the weight reduction might lead to the removal of the output from the excitatory hidden neuron and consequently reduce the total PSP of the output neuron.

In a network with nonzero delays, the proposed method trains the excitatory hidden neuron to fire at a time interval (equal to the corresponding delay connecting the hidden neuron to the output neuron) before a desired time. The early firing of the excitatory hidden neuron increases the total PSP of its successor output neuron at the desired time by the delayed effect of the excitatory hidden spike. However, in the previous situation, where the connections do not have any delays, an excitatory hidden neuron is trained to fire at the same time as the desired time. Correspondingly, (8) is used to adjust w_{hi}, the synaptic weights between the ith input neuron and the hth excitatory hidden neuron, at time t

$$\Delta w_{\mathrm{hi}}(t) = \begin{cases} +\sum_o [x_{\mathrm{hi}}(t-\varepsilon_{\mathrm{oh}})(1-x_{\mathrm{oh}}(t)/A)](w_{\mathrm{oh}}/A), & t=\hat{t}_o^f \\ -\sum_o [x_{\mathrm{hi}}(t-\varepsilon_{\mathrm{oh}})(x_{\mathrm{oh}}(t)/A)](w_{\mathrm{oh}}/A), & t=t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{8}$$

where $x_{\mathrm{hi}}(t)$ is the spike trace corresponding to the connection between the ith input neuron and the hth excitatory hidden neuron. Each spike in the ith input spike train causes a delayed ($\varepsilon_{\mathrm{hi}}$) jump in the trace then it decays exponentially by a time constant similar to (3). $x_{\mathrm{oh}}(t)$ is the trace corresponding to the connection between the hth excitatory hidden neuron and the oth output neuron. Each output spike of the hth excitatory hidden neuron results in a delayed ($\varepsilon_{\mathrm{oh}}$) jump in the trace which decays exponentially by a time constant τ similar to (3). $\varepsilon_{\mathrm{hi}}$ is the delay between the ith input neuron and the hth excitatory hidden neuron, and $\varepsilon_{\mathrm{oh}}$ is the delay between the hth excitatory hidden neuron and the oth output neuron. The traces have same amplitude A and time constant τ as the STDP time window in (2).

The update of w_{hi} at $t=\hat{t}_o^f$ in (8) based on the delayed $x_{\mathrm{hi}}(t)$ increases w_{hi} by a high value if it has spike shortly before $(\hat{t}_o^f-\varepsilon_{\mathrm{oh}})$, because in this case $x_{\mathrm{hi}}(\hat{t}_o^f-\varepsilon_{\mathrm{oh}})$ has a high value. The high increase can lead to the generation of an output spike of the hth excitatory hidden neuron at $(\hat{t}_o^f-\varepsilon_{\mathrm{oh}})$. The effect of the generated hidden spike is shifted to the time of the desired spike in the oth output neuron after the delay of the connection between the hth excitatory hidden neuron and the oth output neuron $\varepsilon_{\mathrm{oh}}$. This helps the output neuron to generate output spike at the desired time.

The decrement in the second expression of (8) is high if the ith input neuron has spike shortly before $(t_o^f-\varepsilon_{\mathrm{oh}})$. Consequently, this decrement tries to remove the actual output of the hth excitatory hidden neuron at $(t_o^f-\varepsilon_{\mathrm{oh}})$ and helps the oth output neuron to reduce its PSP at the time t_o^f (by considering the delay $\varepsilon_{\mathrm{oh}}$).

2) Weight Learning of the Inhibitory Hidden Neurons: The connection weight between the hth inhibitory hidden neuron and the ith input neuron $\bar{w}_{\mathrm{hi}}$ is updated similar to (8) by multiplying it with a negative sign as shown in

$$\Delta \bar{w}_{\mathrm{hi}}(t) = \begin{cases} -\sum_o [\bar{x}_{\mathrm{hi}}(t-\mu_{\mathrm{oh}})(\bar{x}_{\mathrm{oh}}(t)/A)]|w_{\mathrm{oh}}/A|, & t=\hat{t}_o^f \\ +\sum_o [\bar{x}_{\mathrm{hi}}(t-\mu_{\mathrm{oh}})(1-\bar{x}_{\mathrm{oh}}(t)/A)]|w_{\mathrm{oh}}/A|, & t=t_o^f \\ 0, & \text{Otherwise} \end{cases} \tag{9}$$

where μ_{oh} is the delay between the hth inhibitory hidden neuron and the oth output neuron, and $\bar{x}_{\mathrm{hi}}(t)$ is the spike trace corresponding to the connection between the ith input neuron and the hth inhibitory hidden neuron. $\bar{x}_{\mathrm{oh}}(t)$ is the spike trace related to the connection between the hth inhibitory hidden neuron and the oth output neuron. The delay related the connection between the ith input neuron and the hth inhibitory hidden neuron is μ_{hi}. According to (9), the weight is reduced if the ith input neuron has a delayed (μ_{hi}) spike shortly before $(\hat{t}_o^f-\mu_{\mathrm{oh}})$ to increase the total PSP of the oth output neuron at the desired time $\hat{t}_o^f$ by removing hidden inhibitory spike at or before $(\hat{t}_o^f-\mu_{\mathrm{oh}})$. In addition, (9) increases the weight $\bar{w}_{\mathrm{hi}}$ to generate hidden inhibitory spike at $(t_o^f-\mu_{\mathrm{oh}})$ to reduce the total PSP of the oth output neuron at $t=t_o^f$. The reduction of the total PSP removes the actual output spike of the oth output neuron at t_o^f.

It is proposed that hidden neurons receive biofeedback from the readout neurons. Through this biofeedback, the times of desired spikes and actual outputs related to the neurons in the next layer are made available at the hidden layer neurons which use them to adjust their weights appropriately. In this paper, we did not describe the basis of the biofeedback or model it in detail. The training of the network is stopped when it reaches its goal, i.e., the readout neuron generates actual output spikes at the desired times and all the undesired output spikes of the readout are removed.

D. Classification Ability of the Proposed Method

The weight and delay learning characteristics of the proposed method enable it to train a neuron to fire at desired spike times related to an applied input pattern. In a classification task, an input pattern is assigned to the class whose desired spike train is most similar to the actual output of the network. Therefore, the classification ability of the proposed method can be improved if an output neuron is also trained not to fire close to the desired spikes of other classes in addition to firing at the desired times representing to the current class of the input pattern. As a result, the proposed method introduces an additional learning mechanism when a misclassification occurs.

The learning algorithm considers two desired spike trains after a misclassification. The first one is related to the class of the applied input spatiotemporal pattern, i.e., the desired spikes of the correct class, and the second one is related to the class that causes the misclassification (incorrect class). Thus, the learning adjusts the readout neurons and hidden neurons learning parameters at the time of each desired spike related to the class that causes the misclassification. It reduces the weights of the readout neuron that have a spike before the desired time. To force the oth output neuron to not fire at the fth desired spike of class j ($t=\hat{t}_o^{f(j)}$) the weights of the $o$thoutput neuron are adjusted by the following equation at $t=\hat{t}_o^{f(j)}$:

$$\Delta w_{\mathrm{oh}}(t) = -\Psi(t-t_h-d_{\mathrm{oh}}). \tag{10}$$

The proposed classification learning method adjusts an excitatory hidden neuron weight at the desired spike times ($t=\hat{t}_o^{f(j)}$) related to the class that causes the misclassification by the following equation similar to (8):

$$\Delta w_{\mathrm{hi}}(t) = -\sum_o [x_{\mathrm{hi}}(t-\varepsilon_{\mathrm{oh}})(x_{\mathrm{oh}}(t)/A)](w_{\mathrm{oh}}/A). \tag{11}$$

An inhibitory hidden neuron weight at $t=\hat{t}_o^{f(j)}$ is adjusted similar to (9) by the following equation:

$$\Delta \bar{w}_{\mathrm{hi}}(t) = +\sum_o [\bar{x}_{\mathrm{hi}}(t-\mu_{\mathrm{oh}})(1-\bar{x}_{\mathrm{oh}}(t)/A)]|w_{\mathrm{oh}}/A|. \tag{12}$$

The delay related to an excitatory input of a readout neuron is adjusted by (13) at $t = \hat{t}_o^{f(j)}$. The following equation is similar to (4):

$$\Delta\varepsilon_{\mathrm{oh}}(t) = -\Delta t_{\mathrm{om}}(t)(x_{\mathrm{oh}}(t)/x_{\mathrm{om}}(t))^4 \tag{13}$$

The delay related to an inhibitory input of the readout at $t = \hat{t}_o^{f(j)}$ is adjusted through the following equation which is similar to (6):

$$\Delta\mu_{\mathrm{oh}}(t) = +\Delta\bar{t}_{\mathrm{om}}(t)(\bar{x}_{\mathrm{oh}}(t)/\bar{x}_{\mathrm{om}}(t))^4. \tag{14}$$

The proposed method uses a criterion to control the learning level of every pattern and manage the misclassifications during training and adjust the network learning parameters to increase the inter class separability of the network.

Consider a pattern from class i is applied to the network and an actual output of the network is generated. The correlation between the actual output and the corresponding desired spike train of the class i is called c_i which is calculated by the method used in [41] as in

$$c_i = \frac{\upsilon_d \cdot \upsilon_o}{|\upsilon_d||\upsilon_o|} \tag{15}$$

where "$\upsilon_d \cdot \upsilon_o$" denotes the inner product of the two vectors υ_d and υ_o. υ_d and υ_o are two vectors with real value components which are generated from spike trains. A desired spike train is convolved with a symmetric Gaussian function to generate υ_d. Similarly, υ_o is generated by convolving an actual output spike train with the symmetric Gaussian function. $|\upsilon|$ is the length of a vector υ.

A maximum value p and a threshold level Δc for c_i are considered to control the learning. If the correlation metric c_i is less than Δc, the network learning parameters are updated based on the applied training pattern and their desired spike train without considering any extra criteria. In this situation, the network adjusts its learning parameters to increase its knowledge about the applied training pattern inside the class i. The low value of the correlation related to the applied training pattern $c_i < \Delta c$ means that the similarity of the training pattern with the previous trained patterns from the same class i is low and the learning parameters of the network should be adjusted to increase the ability of the network to recognize the patterns inside the class i.

If c_i reaches the value of p, the learning related to the pattern is not applied to the network in the current learning epoch, because the high value of the correlation shows that the knowledge of the presented training pattern is already in the network and it is not necessary to adjust the learning parameters for the current value of c_i. It means that the network has learned the overall distribution of the data from the class i and it is not necessary to memorize all the details of the presented training pattern. It also prevents over training of the network.

If c_i has a value between Δc and p, i.e., $(\Delta c < c_i < p)$, and c_i is appropriately higher than the correlation metric related to the other classes to prevent misclassification, then the learning related to the applied pattern is stopped in the current epoch. Therefore, if $\Delta c < c_i < p$ and $c_i > c_j + \Delta c$ (where $j = \mathrm{argmax}_{\{k\in\{1,2,\ldots,N\}\&k\neq i\}} c_k$, c_k is the correlation metric of the actual output with the kth desired spike train, and N is the number of all the classes), the learning adjustment related to the applied pattern from class i is not applied to the network in the current epoch. The $c_i > c_j + \Delta c$ denotes that the network can distinguish the class of the applied pattern correctly with an appropriate margin (Δc), therefore it is not necessary to have more training for the current value of c_i in the learning epoch.

TABLE I

PROPOSED CLASSIFICATION LEARNING METHOD

when a training pattern from class 'i' is presented in a learning epoch:

If $c_i \le \Delta c$,

The corresponding weights and delays are adjusted to increase c_i i.e. train the network to generate the i^{th} class desired spike.

if $\Delta c < c_i < p$ and $c_i < c_j + \Delta c$

- $c_i < c_j + \Delta c$ implies that c_i has a low value and it could cause a misclassification. So c_i needs to be increased by learning the i^{th} desired spike train (using (1) for training output neurons' weights, (4) and (6) for training output neurons' delays, and (8) and (9) for training hidden Neurons' weight)
- c_j needs to be reduced by training the network to not fire close to the j^{th} class desired spike train (using (10) for training output neurons' weights, (11) and (12) for training hidden neurons' weights, (13) and (14) for training Output neurons delays).

if $(\Delta c < c_i < p$ and $c_i > c_j + \Delta c)$ or $(c_i \ge p)$

The learning parameter adjustment related to the training pattern is not applied to the network in the current epoch. Because c_i has reached an acceptable level in this epoch.

End

If c_i has a value between Δc and p, and $c_i < c_j + \Delta c$, it suggests that a misclassification has occurred. In this situation, the network learning parameters are updated to enhance the interclass separability of the network by training it to not fire close to the desired spike train of the class that causes this misclassification and to reduce c_j. The learning parameters are also updated to increase the ability of the network to generate the desired spike related to the applied pattern from the class i to increase c_i. The reduction of c_j and the increment of c_i may change the situation $c_i < c_j + \Delta c$ to $c_i > c_j + \Delta c$ and prevent the misclassification. The training is continued until the maximum number of learning epochs is reached or if the stopping criteria noted in Table I apply.

A c_i greater than p shows that the network is trained to fire appropriately close to the corresponding desired spike train. Therefore, similar to the situation where ($\Delta c < c_i \le p$ and $c_i > c_j + \Delta c$) the related learning adjustment is not applied to the network. The p value is chosen high enough depending on the desired spike trains related to the different classes to guarantee that when $c_i > p$, c_i is appropriately higher than c_j $(c_i > c_j + \Delta c)$. Desired spike trains related to different classes (related to c_i and c_j) should be chosen in a such a way that the correlation between the desired spike trains are low enough to support the point that if an actual spike train is very similar to the desired spike related to c_i, $(c_i > p)$ then it is appropriately dissimilar to the other classes $(c_j < c_i - \Delta c)$. The values of p and Δc are determined by trial and error. In this paper, the method used in [44] is employed to choose the desired spikes. A sequence of numbers starting from 10 to 100 ms with 10-ms time interval is generated. Then a number of firing times are extracted randomly from the sequence to assign each desired spike train corresponding to a class. In this situation, every two spikes have at least 10-ms interval. The parameter p is set based on the level of precision that the desired spikes should be learned. In this paper, when an actual output spike train reaches 90% of accuracy compared to its corresponding desired spike train the learning is stopped, so the learning parameter p is set 0.9. The parameter Δc should be higher than the maximum correlation between the desired spike trains related to different classes. Δc is set 0.45 to implement the proposed method.

After training, each testing pattern is applied to the network and the readout actual output spike train is calculated. The correlations between the actual output spike train and the desired spike trains corresponding to all classes are obtained. The input pattern is assigned to the class whose corresponding desired spike train has the maximum correlation value with the actual output spike train.

IV. Results

A. *Effect of Network Setups on the Learning Performance*

First, the effects of the different maximum allowable delays and the number of desired output spikes in each class on the performance of the learning method are explored. Then, the running time for the proposed method is reported. In the following simulation, the performance of the network is first evaluated on the Fisher IRIS data set. The IRIS data features are converted to spike times using population coding, as described in [23], where each feature value is encoded by M identically shaped overlapping Gaussian functions where M is set to 40. The IRIS data have four features for each pattern so there are $4 \times M = 160$ input spikes obtained which are then applied to 160 input synapses. The high number of input synapses increases the number of input spikes, and consequently reduces the length of silent windows inside a spatiotemporal input pattern and helps the neuron to fire at multiple desired times. In addition, there are nine extra input synapses with input spikes at fixed times for all patterns. The fixed times are the same as the times of desired spikes corresponding to all classes. These inputs act as bias inputs [21] and act as the reference start times in a multispike coding scheme. There are 360 hidden neurons in the hidden layer. The total time duration of the input spatiotemporal pattern is set to 100 ms, $T = 100$ ms.

1) Effect of Maximum Allowable Delays: Similar to [24], 50% of the IRIS data were selected randomly and used as training data and the remaining used for testing. The accuracy of the proposed method on the testing data reaches its highest value, 95.1%, when the maximum allowable delay D is 3 ms and there is a single readout neuron.

In Table II, the accuracies of the proposed method for different delays when there are three readout neurons (each corresponding to a class) in the network are shown. The accuracy of the method on the testing data reaches its maximum value when $D = 3$ ms (Table II). The accuracy of the proposed method on the testing data is increased from 95.1% to 95.7% when the number of readout neurons is increased from one to three when $D = 3$ ms. In Fig. 4, the accuracy of the learning algorithm on the training data is shown when a single readout neuron and three readout neurons are used. All these procedures are repeated independently for 40 different runs, and the mean value of the 40 results are reported. Different random initial weights and different random selections of the training and testing data are used for the different runs. When the number of readout neurons is increased, the number of learning parameters is also increased. Therefore, the readout neurons learn a lower number of training patterns compared to the situation where a single readout neuron is used, where the readout neuron should learn patterns related to all classes. Subsequently, they can learn the input patterns better compared to the situation that a single readout neuron is used. For higher values of maximum allowable delays, the cooperation between weight adjustment and delay adjustment is reduced and it leads to a lower accuracy. A higher delay adjustment causes a higher shift in the delayed effect of input spikes, and this higher shift might destroy previous weight training that was based on the previous value of the delay.

TABLE II

Effect of the Different Maximum Allowable Delays on IRIS Data Recognition. 50% of the Data Are Used as Training Data

Max-Delays (ms)	Training Accuracy (%)	Testing Accuracy (%)
1	99.8	95.3
3	99.8	95.7
4	99.6	95.7
5	99.6	95.3
7	99.6	94.6
10	98.9	94.5

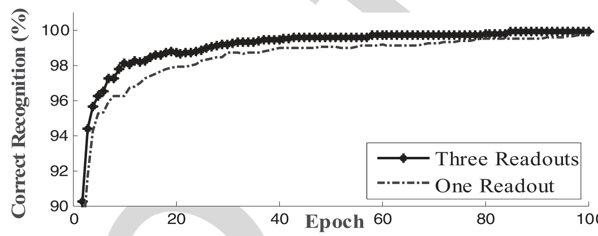

Fig. 4. Comparison of the learning method accuracy on the IRIS data training set when one and three readout neurons are used.

Synaptic delays at chemical synapses usually take values from 1 to 5 ms. The minimum value of a synaptic delay is 0.3 ms. Synaptic delay also can take a value higher than 5 ms [45]. Different researchers use different maximum values for range [1, 16] ms. The results in this section show that for this configuration, 3 ms is an optimal value for the maximum synaptic delay. In the following simulations, Max Delays are set to 3 ms.

2) Effect of the Number of Desired Spikes: In the following experiment, the accuracy of the proposed method is obtained for different numbers of desired spikes corresponding to each class (Table III).

The network reaches its maximum testing accuracy, 95.7%, when three desired spikes are used in each desired spike train.

TABLE III
EFFECT OF THE NUMBER OF DESIRED SPIKES ON LEARNING ACCURACY USING THE IRIS DATA SET WITH THREE READOUT NEURONS

# Desired Spikes for each Class	Training Accuracy (%)	Testing Accuracy (%)
4	96.60	91.5
3	99.80	95.7
2	99.97	95.5
1	99.90	95.1

Fig. 5. Recognition accuracy for different numbers of desired spikes.

A very high number of desired spikes in each desired spike train (i.e., for a desired spike train with 100-ms duration and 10-ms minimum interspike interval, the highest number of desired spikes is 10) reduce the performance of the learning method as this increases the complexity of the learning task and the network should be trained to fire at a higher number of desired instances with a limited number of learning parameters. For instance, the testing accuracy of the proposed method is reduced from 95.7% to 81% when the number of desired spikes is increased from 3 to 7 (Fig. 5).

The time distances between desired spikes of different classes are reduced when there is a high increase in the numbers of desired spikes. Therefore, a small deviation in the times of output spikes can cause a switching from one class to the other one and reduces the accuracy. On the other hand, a lower number of desired spikes reduce the complexity of the learning task, therefore the training accuracy will be increased. However, a very low number of desired spikes lead to a low testing accuracy. For example, when the number of desired spikes is reduced from three to one, the testing accuracy is reduced from 95.7% to 95.1%. It shows that a single spike cannot capture enough information from training data, and consequently, it reduces the testing accuracy despite of a comparably high training accuracy of 99.9%. Moreover, the distributions of spikes in the spatiotemporal input patterns compared to desired spikes also affect the accuracy and the relation between the number of desired spikes, and the accuracy is not a simple linear function (Fig. 5).

3) Evaluation of the Running Time: MATLAB simulations were carried out on a quad core PC with 3 GHz and 16 GB of RAM. The running times required for each learning epoch of the proposed method are reported in Table IV. The running time related to a learning epoch is measured 10 times, and the mean value is reported for each number of input synapses. The running time is increased by increasing the maximum allowable delays D. For instance, the method needs 5.2 s to execute a learning epoch when $D = 1$ ms. However, the running time is increased to 15.9 s when D is increased to 7 ms. Because, at each time step, the learning algorithm should check the events at the previous time steps depending on the delays. A higher number of previous time steps should be considered for a higher value of delays. Therefore, the computational complexity of the method and consequently the running time is increased when the delay is increased.

TABLE IV
EFFECT OF THE MAXIMUM ALLOWABLE DELAY (d) ON THE RUNNING TIME OF THE PROPOSED METHOD USING THE IRIS DATA SET

Max-Delays (ms)	Running Time (sec)	Max-Delays (ms)	Running Time (sec)
1	5.2	5	12.4
3	9.2	7	15.9

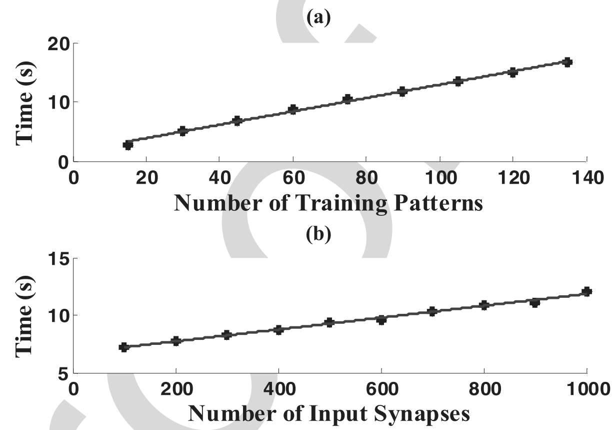

Fig. 6. Runing time of a learning epoch is increased linearly as a function of (a) number of training patterns and (b) number of input synapses.

The running times of a learning epoch of the proposed method are measured for different numbers of training patterns. The number of training patterns is increased from 15 to 135. IRIS data set is used to train the algorithm. Fig. 6(a) shows the relationship between the running times and the number of training patterns. The fit line shown in Fig. 6(a) is obtained by fitting the data points to a 1-D polynomial. The line is described by the equation $T(n) = 0.1128n + 1.593$. The time complexity of the process related to the equation is linear, i.e., it is O(n) using the big O notation. It shows that the running time increases linearly with the number of training samples.

Random spatiotemporal input patterns with different numbers of inputs are used to analyze the complexity of the learning algorithm as a function of the number of input synapses. There are three classes similar to IRIS data in the randomly generated data. A spike train composed of three spikes is considered as desired spike train for each class like the desired spike used for IRIS data. The spike times in each input spatiotemporal pattern are generated by a uniform distribution. The values of spike times are extracted randomly from (0, 100) interval. The number of input synapses is changed from 100 to 1000, and an input spike is considered for each input synapse. Then, the running time for each

TABLE V

COMPARISON WITH THE MULTILAYER SNN PROPOSED IN [28] ON THE IRIS DATA SET

Method	Training Accuracy (%)	Testing Accuracy (%)
Sporea et. al.[28]	*96*	*94*
The proposed method	*99.3*	*95.8*

learning epoch is calculated to analyze the complexity of the learning method. In this experiment, there are a fixed number of 75 training patterns. Fig. 6(b) shows the evolution of the running time in terms of the number of input synapses. In addition, a line fit with the obtained data points is plotted. The dependence between running time and the number of inputs indicates a linear time complexity, i.e., O(n).

B. Comparison With State-of-the-Art Methods

In the following simulation, first the proposed method is compared with the method proposed by Sporea and Grüning [28]. In this case, 75% of the total IRIS data for each class are considered as a training set and the remaining 25% are used for testing, as in [28]. The results are shown in Table V. The accuracy of the proposed method on the training is 99% which is higher than the method proposed in [28], 96%. The proposed method also achieved a higher testing accuracy of 96% (compared to 94% achieved by [28]).

Similar to the biologically plausible structure used in [18], each of the 169 input neurons is connected randomly to a limited number of neurons (40 neurons) in the hidden layer which consists of a population of 360 neurons. There are no subconnections, and every two neurons in two subsequent layers are connected by a single connection similar to the biologically plausible neural network in Izhikevich's work [18]. The proposed learning algorithm is designed to manage the training of a large number of SNs by local events such as spike trace which takes place at the location of each synapsis. There are three output neurons in the output layer and all the hidden neurons are connected to the three output neurons. The network proposed in [28] uses the timing of a single spike of an input neuron for each feature. The four input neurons are fully connected to ten neurons in the hidden layer. Every two neurons in two subsequent layers are connected by 12 subconnections with different delays from 1 to 12 ms. All the neurons in the hidden layer are fully connected to an output neuron. The performance of the method in [28] on the IRIS data is shown in Table V.

In order to compare the accuracy of the proposed method with that achieved by other existing methods, 50% of the data samples from the IRIS data set are selected randomly to construct training data and the remaining 50% are used for testing. The testing results are summarized in Table VI. The accuracies of the proposed method on the training and testing data are 99.7% and 95.7%, respectively. The testing accuracy of the proposed method, 95.7%, is comparable with the best result achieved for the state-of-the-art methods on IRIS data set. The proposed method has a high training accuracy, 99.7%.

TABLE VI

COMPARISON WITH OTHER METHODS ON THE IRIS DATA SET

Method	Testing Accuracy (%)
Spiking Methods	
RBF [50]	*92.6*
SWAT [51]	*95.3*
SpikeProp [24]	*95*
QuickProp [23]	*92.3*
RProp [23]	*93.2*
RBF [53]	*89*
SNN (Bako) [52]	*83.4*
Proposed Method	*95.7*
Non-Spiking Methods	
K-Means [50]	*88.6*
SOM [50]	*85.33*
Matlab BP [51]	*95.5*
Matlab LM [51]	*95.7*
TEST [55]	*91.7*

TABLE VII

COMPARISON WITH OTHER METHODS ON THE WBCD DATA SET

Method	Testing Accuracy (%)
Spiking Methods	
SWAT [51]	*95.3*
SpikeProp [24]	*97*
SNN (Bako) [52]	*89.5*
Proposed Method	*96.4*
Non-Spiking Methods	
MATLAB Autoencoder	*96.2*
Matlab BP [51]	*96.3*
Matlab LM [51]	*96.7*
DBN [54]	*96.8*

The proposed method converges for all trials because it does not have the silent neuron problem. It has remote supervised spikes. In addition, it solves the problem of silent windows in a spatiotemporal input pattern by delay learning. A silent window can prevent generation of desired spikes and consequently it can cause learning convergence problem. These characteristics of the proposed method make it appropriate for learning multiple spikes. The accuracies of the proposed method are calculated for all trials, and there are not any rejected results. In contrast, the convergence rate of SpikeProp is investigated in [24] and as it has a problem with silent neurons it cannot converge for all trials, and as a result, those trials with low accuracies are removed from the reported results [24].

The Breast Cancer Wisconsin (Diagnostic) data set (WBCD) from the UCI machine learning repository is used as the second data set to evaluate the proposed method and to compare it with the other state-of-the-art methods, as shown in Table VII. WBCD contains 699 samples. The samples belong to two different classes (malignant and benign categories) where 458 samples are from the first category and 241 samples are from the second category. A total of 120 samples are selected

TABLE VIII

PERFORMANCE COMPARISON WITH SRESN AND GPSNN ON THE BUPA LIVER DISORDERS DATA SET

Method	Testing (Training) (%)
SRESN [46]	*59.7 (60.4)*
GPSNN [47]	*59.8 (61.18)*
Proposed Method	*61.8(69.9)*

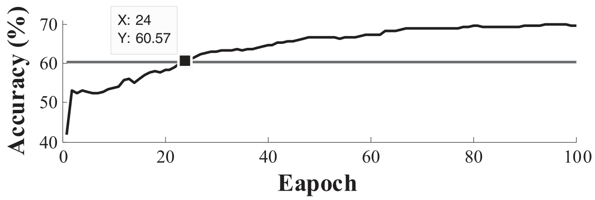

Fig. 7. Evolution of the accuracy of the proposed method over different learning epochs on BUPA liver disorders data. It needs 24 learning epochs to pass the accuracy level of 60%. SRESN [46] needs 715 epochs to reach about to the same level of accuracy.

TABLE IX

COMPARISON WITH SRESN ON DIFFERENT DATA SETS

Data Set	Testing (Training) (%)	Max # epochs	# LP [c]
	Proposed Method		
Pima diabetes [a]	*70.6 (72.1)*	*100*	*14640-1440*
BUPA	*61.8 (69.9)*	*100*	*9840-1440*
Ionosphere [b]	*90.5 (96.0)*	*100*	*54640-1440*
Iris	*95.7 (99.8)*	*100*	*6760-1440*
WBCD	*96.4 (98.2)*	*100*	*14640-1440*
	SRESN [46]		
Pima diabetes [a]	*69.9 (70.5)*	*254*	*486-756*
BUPA	*59.7 (60.4)*	*715*	*216-324*
Ionosphere [b]	*88.6(91.9)*	*1018*	*3264-4692*
Iris	*97.3(96.9)*	*102*	*120-200*
WBCD	*97.2(97.7)*	*306*	*432-648*

[a] Pima diabetes data from the UCI machine learning repository contains768 samples in which 500 and 268 samples are in two classes.
[b] Ionosphere data from the UCI machine learning repository contains 351 samples in which 225 and 126 samples are in two classes.
[c] # LP: Number of Learning parameters

randomly from each category to construct the training set, and the remaining data is used for testing. The proposed method has an accuracy comparable with the best accuracy achieved by the other state-of-the-art methods (Table VII).

One advantage of SNNs is that they use spikes to communicate between neurons. However, in the classical neural networks, real values are used to transfer data between neurons. Each spike can be encoded by a binary bit; however, a real value needs a high number of bits to be transferred between neurons depending on the precision that is required for the values. As shown in Tables VI and VII, the proposed method using spikes for communication between neurons and can achieve better or comparable accuracies with the state-of-the-art rate-based models including deep belief network (DBN) and autoencoders.

One more data set which is used to evaluate the proposed method is the BUPA liver disorders data from the UCI machine learning repository. There are 345 samples in this data set in which 145 samples are from the first class and 200 samples are from the second class. A total of 70 data samples are selected randomly from each class to construct the training set, and the remaining data is used for testing. Each sample has six attributes. The performance of the proposed method is shown in Table VIII. The testing accuracy of the proposed method is higher than SRESN [46] and GPSNN [47]. SRESN [46] uses a 30-2 architecture, and the proposed method uses a 246-360-2 architecture where there are 246 input neurons, 360 hidden neurons, and two output neurons. The evolution of the training accuracy of the proposed method over different learning epochs is shown in Fig. 7. The proposed method needs 24 learning epochs to pass the training accuracy of 60.4%; however, SRESN [46] needs 715 learning epochs to reach the same accuracy level. The proposed method can reach the accuracy level of 66.9% in less than 100 epochs.

The performance of the proposed method on different data sets is compared with SRESN [46] in Table IX. The number of learning parameters in SRESN [46] is lower than that of the parameters in the proposed method (see Table IX). A lower number of learning parameters can reduce the simulation time required for each learning epoch. However, the proposed method achieved high accuracies in a lower number of learning epochs compared to the method with a single layer of learning neurons on Pima diabetes, BUPA liver disorder, and ionosphere data sets. The proposed learning method achieves this improvement through appropriate interaction between different layers of SNs in a multilayer structure.

V. CONCLUSION

This paper proposed a BPSL for multilayer SNNs. It uses the precise timing of multiple spikes, which is a biologically plausible information coding scheme. The learning parameters of neurons in the hidden layer and output layer are learned in parallel using STDP, anti-STDP, and delay learning.

The simulation results show that the proposed method has improved the performance of the first fully supervised algorithm that learns multiple spikes in all layers proposed in [28].The improvement of the proposed method can be attributed to a number of properties of the proposed method. First, it has used the firing times of spikes fired by the hidden neurons to train the weights of the hidden neurons unlike the method in [28] where the firing time of hidden neurons is not considered and the weights of a hidden neuron are adjusted by the same values irrespective of the neuron firing at the desired times or not firing at all. In the proposed method, weight learning, based on the firing times of the hidden neurons, helps adjust the weights appropriately and prevents unnecessary weight adjustments. Another property of the proposed method is the appropriate use of the EPSP and the IPSP produced by the hidden excitatory and inhibitory neurons to effectively adjust their weights, unlike the approach in [28] where equal weight updates are applied to both excitatory and inhibitory neurons, which can reduce the learning performance. Another property of the proposed method that improves its performance compared to the learning method in [28] is the appropriate consideration of the effect of delays on the weight learning. It was shown that the delay after a hidden neuron has an essential effect on the output of the spiking network, hence it should be considered during the training of the weights of the hidden neuron. For example, an excitatory hidden neuron should fire earlier than a desired output spike depending on the delay after the hidden neuron, as described in the previous sections. The produced PSP by the fired hidden spike is shifted to the desired time by the delay. The effect of the delay on the weight adjustments of hidden neurons is not considered in [28], and it was shown that this resulted in a lower accuracy compared to the proposed method on the IRIS data set.

The performance of the proposed method was also compared with other algorithms on different data sets. The results showed that the proposed method can achieve a higher accuracy compared to a single-layer SNN. In addition, the method has comparable accuracy with the best result achieved by state-of-the-art rate-based neural models including autoencoders and DBNs.

The results also showed that a very high number of desired spikes can reduce the accuracy of the method by increasing the complexity of the learning task, and a very low number of desired spikes cannot capture all the temporal information of input data. Although the delay learning increases the complexity of the learning method and consequently the running time, it was shown that delays can increase the learning performance of the proposed method. In addition, delays are a biologically plausible property of SNNs. Another property of the proposed method is its multilayer structure that increases the computational cost of each learning epoch. However, the results showed that it can also reduce the number of learning epochs and can improve its accuracy compared to the similar multilayer spiking network proposed by Sporea and Grüning [28]. The ablity of the proposed method to effectively learn multiple desired spikes suggests that this approach may be suitable for neuroprosthetic applications.

In a biologically plausible neuron model, the output of a neuron depends not only on synaptic inputs, but also on the internal dynamics of the neuron [48]. Therefore, a potential direction for future work is to incorporate the neuron internal dynamics in the proposed method, additionally to the effect of the synaptic weight and delays, which may lead to a new learning algorithm with potentially higher performance. For instance, Zhang *et al.* [49] have proposed a dynamic firing threshold to make the spiking network learning robust to noise. A similar method can be applied to the multilayer spiking network proposed in this paper to further improve its performance.

It is possible to extend the learning algorithm to more layers (deep SNNs). However, more layers may reduce the effect of training of earlier layers on the network output. Designing effective learning methods for deep spiking networks will be investigated in the future work.

REFERENCES

[1] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biol. Cybern.*, vol. 98, no. 6, pp. 459–478, 2008.

[2] V. Ramaswamy and A. Banerjee, "Connectomic constraints on computation in feedforward networks of spiking neurons," *J. Comput. Neurosci.*, vol. 37, no. 2, pp. 209–228, 2014.

[3] P. A. Cariani, "Temporal codes and computations for sensory representation and scene analysis," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1100–1111, Sep. 2004.

[4] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, no. 6535, pp. 33–36, 1995.

[5] A. Mohemmed and S. Schliebs, "Training spiking neural networks to associate spatio-temporal input-output spike patterns," *Neurocomputing*, vol. 107, pp. 3–10, May 2013.

[6] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," in *Handbook Natural Computing Anonymous*. Springer, 2012, pp. 335–376.

[7] S. B. Laughlin, R. R. de Ruyter van Steveninck, and J. C. Anderson, "The metabolic cost of neural information," *Nature Neurosci.*, vol. 1, no. 1, pp. 36–41, 1998.

[8] S. B. Laughlin, "Energy as a constraint on the coding and processing of sensory information," *Current Opinion Neurobiol.*, vol. 11, no. 4, pp. 475–480, 2001.

[9] J. Hu, H. Tang, K. C. Tan, H. Li, and L. Shi, "A spike-timing-based integrated model for pattern recognition," *Neural Comput.*, vol. 25, no. 2, pp. 450–472, 2013.

[10] A. Kasiński and F. Ponulak, "Comparison of supervised learning methods for spike time coding in spiking neural networks," *Int. J. Appl. Math. Comput. Sci.*, vol. 16, no. 1, pp. 101–113, 2006.

[11] R. V. Florian, "The chronotron: A neuron that learns to fire temporally precise spike patterns," *PLoS ONE*, vol. 7, no. 8, p. e40233, 2012.

[12] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.

[13] D. T. Pham, M. S. Packianather, and E. Y. A. Charles, "Control chart pattern clustering using a new self-organizing spiking neural network," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 222, no. 10, pp. 1201–1211, 2008.

[14] R. Brette *et al.*, "Simulation of networks of spiking neurons: A review of tools and strategies," *J. Comput. Neurosci.*, vol. 23, no. 3, pp. 349–398, Jul. 2007.

[15] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[16] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, 1952.

[17] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[18] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Comput.*, vol. 18, no. 2, pp. 245–282, 2006.

[19] W. Maass and A. Zador, "Computing and learning with dynamic synapses," *Pulsed Neural Netw.*, vol. 6, pp. 321–336, May 1999.

[20] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition," *Neural Netw.*, vol. 41, pp. 188–201, May 2013.

[21] S. M. Bohte, J. N. Kok, and H. L. Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, 2002.

[22] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the spikeprop algorithm," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2006, pp. 3970–3977.

[23] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for EEG classification and epilepsy and seizure detection," *Integr. Comput.-Aided Eng.*, vol. 14, no. 3, pp. 187–212, 2007.

[24] S. B. Shrestha and Q. Song, "Adaptive learning rate of spikeprop based on weight convergence analysis," *Neural Netw.*, vol. 63, pp. 185–198, Mar. 2015.

[25] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, 2010.

[26] A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature Neurosci.*, vol. 2, no. 11, pp. 947–957, 1999.

[27] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Netw.*, vol. 43, pp. 99–113, Jul. 2013.

[28] I. Sporea and A. Grüning, "Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 25, no. 2, pp. 473–509, 2013.

[29] R. Memmesheimer, R. Rubin, B. P. Ölveczky, and H. Sompolinsky, "Learning precisely timed spikes," *Neuron*, vol. 82, no. 4, pp. 925–938, 2014.

[30] S. Hong, S. Ratté, S. A. Prescott, and E. De Schutter, "Single neuron firing properties impact correlation-based population coding," *J. Neurosci.*, vol. 32, no. 4, pp. 1413–1428, 2012.

[31] J. L. Rosselló, V. Canals, A. Oliver, and A. Morro, "Studying the role of synchronized and chaotic spiking neural ensembles in neural information processing," *Int. J. Neural Syst.*, vol. 24, no. 5, p. 1430003, 2014.

[32] J. P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning," *Neural Comput.*, vol. 18, no. 6, pp. 1318–1348, 2006.

[33] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "SPAN: Spike pattern association neuron for learning spatio-temporal spike patterns," *Int. J. Neural Syst.*, vol. 22, no. 4, p. 1250012, 2012.

[34] Y. Xu, X. Zeng, and S. Zhong, "A new supervised learning algorithm for spiking neurons," *Neural Comput.*, vol. 25, no. 6, pp. 1472–1511, 2013.

[35] A. Taherkhani, A. Belatreche, Y. Li, and L. Maguire, "A new biologically plausible supervised learning method for spiking neurons," in *Proc. ESANN*, Bruges, Belgium, 2014, pp. 1–6.

[36] M. Zhang *et al.*, "EMPD: An efficient membrane potential driven supervised learning algorithm for spiking neurons," *IEEE Trans. Cogn. Develop. Syst.*, to be published.

AQ:5

[37] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural Netw.*, vol. 22, no. 10, pp. 1419–1431, 2009.

[38] O. Booij and H. tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Inf. Process. Lett.*, vol. 95, no. 6, pp. 552–558, 2005.

[39] H. Adeli and S. L. Hung, "An adaptive conjugate gradient learning algorithm for efficient training of neural networks," *Appl. Math. Comput.*, vol. 62, no. 1, pp. 81–102, 1994.

[40] S. L. Hung and H. Adeli, "Object-oriented backpropagation and its application to structural design," *Neurocomputing*, vol. 6, no. 1, pp. 45–55, 1994.

[41] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "DL-ReSuMe: A delay learning-based remote supervised method for spiking neurons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3137–3149, Dec. 2015.

[42] B. Glackin, J. A. Wall, T. M. McGinnity, L. P. Maguire, and L. J. McDaid, "A spiking neural network model of the medial superior olive using spike timing dependent plasticity for sound localization," *Frontiers Comput. Neurosci.*, vol. 4, p. 18, Aug. 2010.

[43] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "EDL: An extended delay learning based remote supervised method for spiking neurons," in *Neural Information Processing*, S. Arik, Ed. Springer, 2015, pp. 190–197.

[44] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "Multi-DL-ReSuMe: Multiple neurons delay learning remote supervised method," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Killarney, Ireland, Jul. 2015, pp. 1–7.

[45] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, and A. J. Hudspeth, *Principles of Neural Science*. New York, NY, USA: McGraw-Hill, 2004.

[46] S. Dora, K. Subramanian, S. Suresh, and N. Sundararajan, "Development of a self-regulating evolving spiking neural network for classification problem," *Neurocomputing*, vol. 171, pp. 1216–1229, Jan. 2016.

[47] S. Dora, S. Sundaram, and N. Sundararajan, "A two stage learning algorithm for a growing-pruning spiking neural network for pattern classification problems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–7.

[48] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[49] M. Zhang, H. Qu, X. Xie, and J. Kurths, "Supervised learning in spiking neural networks with noise-threshold," *Neurocomputing*, vol. 219, pp. 333–349, Jan. 2017.

[50] S. M. Bohte, H. La Poutré, and J. N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 426–435, Mar. 2002.

[51] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A spiking neural network training algorithm for classification problems," *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.

[52] L. Bako, "Real-time classification of datasets with hardware embedded neuromorphic neural networks," *Briefings Bioinf.*, vol. 11, no. 3, pp. 348–363, 2010.

[53] N. Gueorguieva, I. Valova, and G. Georgiev, "Learning and data clustering with an RBF-based spiking neuron network," *J. Experim. Theor. Artif. Intell.*, vol. 18, no. 1, pp. 73–86, 2006.

[54] M. Tanaka and M. Okutomi, "A novel inference of a restricted Boltzmann machine," in *Proc. 22nd Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2014, pp. 1526–1531.

[55] S. Yoon and S.-Y. Lee, "Training algorithm with incomplete data for feed-forward neural networks," *Neural Process Lett.*, vol. 10, no. 3, pp. 171–179, 1999.

Aboozar Taherkhani received the B.Sc. degree in electrical and electronic engineering from Shahid Beheshti University, Tehran, Iran, the M.Sc. degree in biomedical engineering from the Amirkabir University of Technology, Tehran, and the Ph.D. degree from Ulster University, Londonderry, U.K., in 2017.

He is currently a Research Fellow with the Computational Neuroscience and Cognitive Robotics Laboratory, Nottingham Trent University, Nottingham, U.K. His current research interests include artificial intelligence, spiking neural network, deep neural network, complex system theory, and nonlinear signal processing.

Ammar Belatreche (M'09) received the Ph.D. degree in computer science from Ulster University, Londonderry, U.K.

He was a Lecturer in computer science with the School of Computing and Intelligent Systems, Ulster University. He is currently a Senior Lecturer in computer science with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. His current research interests include bioinspired adaptive systems, machine learning, pattern recognition, data analytics, capital market engineering, and image processing and understanding.

Dr. Belatreche is a fellow of the Higher Education Academy, an Associate Editor of *Neurocomputing*, and has served as a Program Committee Member and a reviewer for several international conferences and journals.

Yuhua Li (SM'11) received the Ph.D. degree in general engineering from the University of Leicester, Leicester, U.K.

He was a Senior Research Fellow with Manchester Metropolitan University, Manchester, U.K., and a Research Associate with the University of Manchester, Manchester, from 2000 to 2005. He was a Lecturer with the School of Computing and Intelligent Systems, Ulster University, Londonderry, U.K., from 2005 to 2014, and with the School of Computing, Science and Engineering, University of Salford, Greater Manchester, U.K, from 2014 to 2017. He is currently a Senior Lecturer with the School of Computer Science and Informatics, Cardiff University, Cardiff, U.K. His current research interests include machine learning, pattern recognition, data science, knowledge-based systems, and condition monitoring and fault diagnosis.

Liam P. Maguire (M'15) received the M.Eng. and Ph.D. degrees in electrical and electronic engineering from the Queen's University of Belfast, Belfast, U.K., in 1988 and 1991, respectively.

He is currently a Professor and the Executive Dean of the Faculty of Computing, Engineering and the Built Environment, Ulster University, Londonderry, U.K. He has authored over 250 research papers, including 80 journal papers. He has an established track record of securing research funding and has also supervised 18 Ph.D. and 3 M.Phil. students to completion. His current research interests include cognitive data analytics, the development of new bioinspired intelligent systems, and the application of existing intelligent techniques.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ:1 = Please provide the appropriate reference number for McCulloch and Pitts (1943).
AQ:2 = Please provide the expansion for the acronyms "ReSuMe, EDL, SRESN, and GPSNN."
AQ:3 = Please provide the section numbers to replace the phrase "the previous sections."
AQ:4 = Please provide the publisher location for refs. [6] and [43].
AQ:5 = Please provide the volume no., issue no., page range, month, and year for ref. [36].