Optimization of Output Spike Train Encoding for a Spiking Neuron Based on its Spatio–Temporal Input Pattern

Aboozar Taherkhani, *Member, IEEE*, Georgina Cosma, *Member, IEEE*, and Thomas Martin McGinnity, *Senior Member, IEEE*

***Abstract*—A common learning task for a spiking neuron is to map a spatio–temporal input pattern to a target output spike train. There is no prescribed method for selection of the target output spike train. However, the precise spiking pattern of the target output spike train (output encoding) can affect the learning performance of the spiking neuron. Therefore, systematic methods of finding the optimum spiking pattern for a target output spike train that can be learned by spiking neurons are needed. Here, a method is proposed to adaptively adjust an initial suboptimal output encoding during different learning epochs to find the optimal output encoding. A time varying value of a local event called a spike trace is used to calculate the amount of a required adjustment. The remote supervised method (ReSuMe) learning algorithm is used to train the weights, and the proposed method is used for finding optimized output encoding (optimized desired spikes). Experimental results show that optimizing the output encoding during the learning phase increases the accuracy. The proposed method was applied to find optimized output encoding in classification tasks and the results revealed improvements up to 16.5% in accuracy compared to when using the non-adapted method. It also increases the accuracy in a classification task from 90% to 100%.**

***Index Terms*—Encoding, learning, spatio–temporal patterns, spike trace, spike train, spiking neural network (SNN).**

I. Introduction

THE ABILITY of the brain to solve complex problems has inspired researchers to study its processing functions and learning procedures. Artificial neural networks (ANNs) are powerful engineering tools in many domains, such as pattern recognition, control, bioinformatics, and robotics. Despite the fact that rate-based coding is commonly used in traditional ANNs, it is unlikely that rate-based coding can convey all the information related to a rapid processing task such as color, odor, and sound processing [1], [2]. Spikes are an important part of information transmission between neurons in the brain, and there is biological evidence to show that information is encoded in the precise timing of the spikes [3], [4].

SpikeProp [5] was one of the first learning algorithms for spiking neural networks (SNNs). It is a gradient-based learning method and trains a neuron to fire a single desired output spike. In SpikeProp [5] each class is labeled by a desired output spike, i.e., each output neuron is fired at a desired time when an input from the class related to the desired output spike is applied to the network. The supervised multispike learning algorithm [6] is another gradient-based learning algorithm that can train an SNN to fire a desired output spike train with non-adapted spike times corresponding to each class. This method does not find the optimal desired spike times for different classes. Remote supervised method (ReSuMe) [7] is a biologically plausible learning algorithm that works based on spike timing dependent plasticity (STDP) and anti-STDP to train a neuron firing desired output spikes at non-adapted times. QuickProp [8], RProp [9], Chronotron [10], SPAN [11], EMPD [12], BPSL [13], EDL [14], and the supervised method proposed in [15] are other examples of learning algorithms for training spiking neurons to fire at non-adapted desired output times. The times of desired output spikes are usually set randomly, and the random target spikes might not be an appropriate choice for a classification task, which can result in a reduction in learning efficiency. For example, a neuron cannot learn to fire a target spike which is randomly set at a time that there are no or an insufficient number of input spikes in a time window around the target time.

Tempotron [16] is a biologically plausible supervised learning method that does not force a neuron to fire at non-adapted predefined times. It can train a neuron to fire an output spike, however, the spike time is not restricted. A dynamic evolving SNN (deSNN) was proposed by Kasabov *et al.* [17] for classification tasks, and it is a semi-supervised learning method to capture the temporal dynamics of input patterns. deSNN does not set predefined constant times for firing neurons. Yu *et al.* [18] designed a scheme to make decisions on output spikes of a feedforward network. They used N on/off neurons to encode the output of the network to 2^N classes. In the scheme proposed by Yu *et al.* [18], if

Manuscript received November 5, 2018; revised February 12, 2019 and March 21, 2019; accepted March 29, 2019. Date of publication April 11, 2019; date of current version September 9, 2020. This work was supported by the Leverhulme Trust Research Project "Novel Approaches for Constructing Optimised Multimodal Data Spaces" under Grant RPG-2016-252. *(Corresponding author: Aboozar Taherkhani.)*

A. Taherkhani and G. Cosma are with the Computational Neuroscience and Cognitive Robotics Research Group, Nottingham Trent University, Nottingham NG11 8NS, U.K. (e-mail: aboozar.taherkhani@ntu.ac.uk; georgina.cosma@ntu.ac.uk).

T. M. McGinnity is with the Computational Neuroscience and Cognitive Robotics Research Group, Nottingham Trent University, Nottingham NG11 8NS, U.K., and also with the Intelligent Systems Research Centre, Ulster University, Londonderry, U.K. (e-mail: martin.mcginnity@ntu.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/TCDS.2019.2909355

one neuron acts incorrectly, it completely changes the output of the network. Pham *et al.* [19] proposed a learning method for a self-organizing SNN for pattern clustering. In this method, each spiking neuron acts as a coincident detector (CD). A Hebbian-based rule is applied to shift the synaptic delays. The neuron threshold level is set to a small value at the beginning of the learning phase and it is then increased during different learning epochs. Similar to other methods such as Tempotron [16], this method has no predefined constant times for firing neurons.

The temporal spiking pattern of a desired output spike train has a significant effect on the performance of a learning algorithm for SNNs. It is hypothesized that finding an optimum desired output spike train encoding can improve the performance of the learning algorithm, by making the task of the learning algorithm easier. Mohemmed *et al.* [4] have shown that the spiking patterns of desired output spike trains, which are used for labeling different classes, affect the performance of SNN learning algorithms. They used spatio–temporal input patterns which were applied to a neuron with 200 input synapses. Spikes in the input pattern were generated using a uniform random distribution in the interval [0, 200] ms. They have shown that shifting the firing time of a desired output spike from a very early time (at 33 ms) to a late time (165 ms), can significantly improve classification performance. In particular, classification accuracy increased from 51% to 99% when shifting the firing time of a spike to a later time [4].

Legenstein *et al.* [20] constructed a desired output spike train for a spatio–temporal input pattern, when synaptic weights were chosen randomly. Xu *et al.* [6] have investigated the effect of the number and the times of desired output spikes on the performance of an SNN learning algorithm for a pattern classification task. In the classification task, a desired spike train is assigned to each class. A teacher signal is used in each learning epoch to train a spiking neuron. The teacher signal is called desired spike train and it contains a number of desired spikes. Xu *et al.* [6] have shown that appropriate selection of the time of a desired output spike can improve the performance of the learning algorithm. They set the time of the first desired output spike to an arbitrary value, then extracted the time of the second desired output spike based on the distribution of actual output spikes after the first run of learning on the first desired spike. Note that the term "actual output spikes" describes those spikes generated by a spiking neuron in response to an applied input spatio–temporal pattern. Then the SNN was trained with the new desired output spike train (composed of the first spike and the newly extracted desired spike). The method proposed by Xu *et al.* [6] does not combine the desired output spike extraction with the learning procedure of an SNN, and it does not provide any mechanism to adjust the time of the first desired output spike. In general, there is a need for methods which can identify optimum output spike patterns to encode outputs for training spatio–temporal input patterns in spiking neurons, and hence to improve the learning capabilities of SNNs [21].

Biological evidence shows that the firing times of a biological neuron are dynamically changed, and irregular behavior has been reported with respect to the firing activity of neurons in *in-vivo* and *in-vitro* experiments [22]. Stochastic behavior

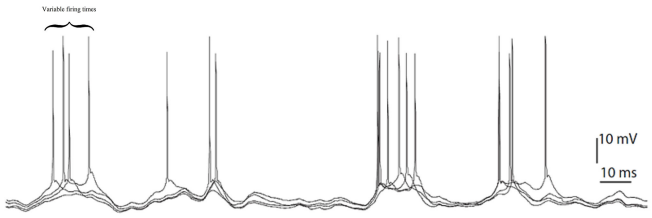


Fig. 1. Variable spike times in various trials with same time dependent input stimulus (adapted from [22]).

of cortical neurons has been reported in many biological experiments. These experiments show that spikes may shift from trial to trial, even though the same trial is performed several times [22]. In biological experiments, a time-dependent current is injected into a neuron several times in a controlled situation and typically results in output traces as shown in Fig. 1 from repeated experimental runs. The firing times of the biological neuron are not constant, even if exactly the same input is applied, and there is a fluctuation around the time of a spike from trial to trial. In the method proposed here, this biological property was used to design a method to dynamically change the times of desired output spikes to find an appropriate spiking pattern for the desired spikes that minimizes the challenge to the learning algorithm in learning the spatio–temporal input.

More specifically, this paper proposes a method for finding an optimal desired output spike train (optimal output encoding) for a spiking neuron. An initial suboptimal desired spike train composed of a number of spikes is generated randomly, and then the suboptimal spikes are adjusted in time to create an optimized set of spikes via controlled shifts in the initial spike times. The spiking neuron can be trained to map the spatio–temporal input pattern to the shifted desired spikes with high accuracy. In the proposed method, the times of initial suboptimal desired output spikes evolve to reach a pattern of spiking activity that the spiking neuron can learn with a high accuracy. The method can be applied to different learning algorithms for SNNs. Section II discusses the proposed method for finding desired output spikes; Section III provides the experiment results; and a discussion and conclusion are presented in Sections IV and V, respectively.

II. Proposed Method for Finding Desired Output Spikes

The aim of the learning task is to map a spatio–temporal input pattern composed of a number of spike trains to a desired spike train. This section proposes a method to find compatible desired output spikes with the spatio–temporal input pattern. In the proposed method, ReSuMe is used to adjust the learning parameters (i.e., synaptic weights) of a spiking neuron to train the neuron although any similar learning algorithm could be employed. In addition, to utilizing the ReSuMe learning algorithm to adjust the synaptic weights of a spiking neuron, a new method is proposed to gradually shift the desired spikes toward the nearest actual output spikes, thus easing the task of the learning algorithm. The proposed mechanism gives flexibility to the learning task to increase the accuracy of the learning procedure.

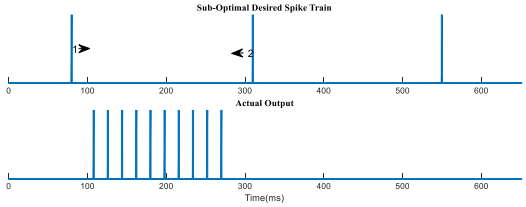


Fig. 2. Desired spike train and its corresponding actual output spike train before learning.

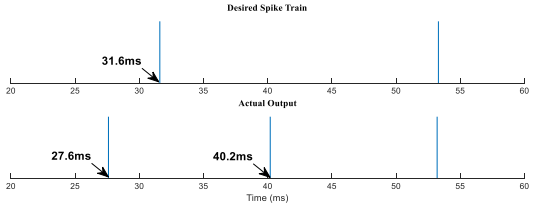


Fig. 3. Desired spike and its neighbor actual output spikes. The desired spike is close to the actual output spike on the left side.

For example, Fig. 2 shows an initial suboptimal desired spike train, and the actual output spike train of a spiking neuron before training. The initial desired spike train is far from the actual output of the spiking neuron. There are three desired spikes in the suboptimal desired spike train, and the proposed method shifts each desired spike in the train to find an optimum time for each spike in different learning epochs. For the situation shown in Fig. 2 in addition to weight learning, we propose to shift the desired output spikes toward the nearest existing actual output spikes. The first and the second desired spikes are shifted to the firing times of their nearest actual output spikes. The shift directions of the two spikes are shown by two arrows in Fig. 2. The third desired spike does not shift in the current learning epoch because there are no output spikes close enough for the third desired spike in the current learning epoch. The shifting of the desired output spike can improve the learning by removing the extra weight adjustment related to weight enhancement at the time of the first and second desired output spikes. The shifting of the desired spike can also prevent the weight adjustment required for removing the two nearest actual output spikes (as shown in Fig. 2) to the first and the second desired spikes. In this case, the ReSuMe algorithm adjusts the weights to remove only the eight extra spikes instead of ten spikes.

Each desired output spike shifts according to a heuristic rule proposed in Section II-A. In each learning epoch, each suboptimal desired output spike is compared to its surrounding actual output spikes, and it shifts right or left to reach the nearest actual output spike. Fig. 3 shows a desired output spike at $t = 31.6$ ms. There are two actual output spikes around the desired output spike. The first actual output spike is at $t = 27.6$ ms and the second one is at $t = 40.2$ ms. The desired spike is close to the left actual output spike, and therefore, it is shifted slightly to the left in the current learning epoch. The shift is small. Additionally, the learning algorithm adjusts the learning parameters of the neuron to generate an actual output spike at the current desired spike. It takes a number of learning epochs to fire output spikes at the desired time. During the learning phase, weight learning occurs and this may change the sequence of the actual output spikes and, consequently, this may also change the direction of the shift of the desired spike. If there is an actual output spike at a desired time, then the desired spike shift is stopped.

A. Proposed Method for Finding the Direction and Amount of Desired Spike Shift

The proposed method uses local variables, called spike traces, to find the nearest actual output spike around a desired spike, and to calculate the distance between the nearest actual output spike and the desired spike. First, the time distance of the nearest output spike after the desired spike and the nearest output spike before the desired spike are found. Then these two time intervals are compared to find the nearest actual output spike to the desired spike. In this paper, a method is proposed to calculate the time intervals in an online manner, i.e., this method uses the momentary value of local variables (spike traces) to calculate the time intervals and consequently the nearest actual output spike to the desired spike. This method does not save the previous spike times and it works based on current events which are generated by the previous spiking activities.

Similar to Morrison *et al.* [23], the proposed method uses a saturation method to model the spike traces, and we adopted a similar approach to that proposed in our previous work, DL-ReSuMe [1], to calculate time intervals using spike traces. $x_d(t)$ is defined as the trace of a desired spike train. Each desired spike causes a jump to a constant value A and then it decays exponentially according to

$$x_d(t) = \begin{cases} Ae^{-\left(t-t_d^f\right)/\tau}, & \text{for } t_d^f < t < t_d^{f+1} \\ A, & \text{for } t = t_d^1, \ldots, t_d^{f-1}, t_d^f, t_d^{f+1}, \ldots \end{cases} \quad (1)$$

where τ is the exponentially decay time constant, and amplitude A is a constant value where the trace jumps at the time of a desired spike. t_d^f is the time of the fth spike in the desired spike train. Each actual output spike, after a desired spike, resets the desired spike trace to zero [Fig. 4(b)]. The time of the first actual output spike after a desired spike is of interest. Similarly, $x_a(t)$ is defined as the trace of an actual output spike train, as follows:

$$x_a(t) = \begin{cases} Ae^{-\left(t-t_a^f\right)/\tau}, & \text{for } t_a^f < t < t_a^{f+1} \\ A, & \text{for } t = t_a^1, \ldots, t_a^{f-1}, t_a^f, t_a^{f+1}, \ldots \end{cases} \quad (2)$$

where t_a^f is the firing time of the fth actual output spike [Fig. 4(c)].

The time interval between the desired spike and the first previous actual output spike, dt_{ad}, is calculated and compared to dt_{da} (the time interval between a desired spike and the first actual output spike after the desired spike) to find the closest actual output spike to the desired spike [as shown in Fig. 4(a)]. In the proposed method, calculations of dt_{ad} and dt_{da} are performed at the time of the first actual output spike after the desired spike, t_a in Fig. 4(d). In the first step, $dt_{aa}(t_a)$, i.e., the time interval between the actual output spikes before and after the desired spike, and $dt_{\mathrm{da}}(t_a)$ are calculated at time t_a.

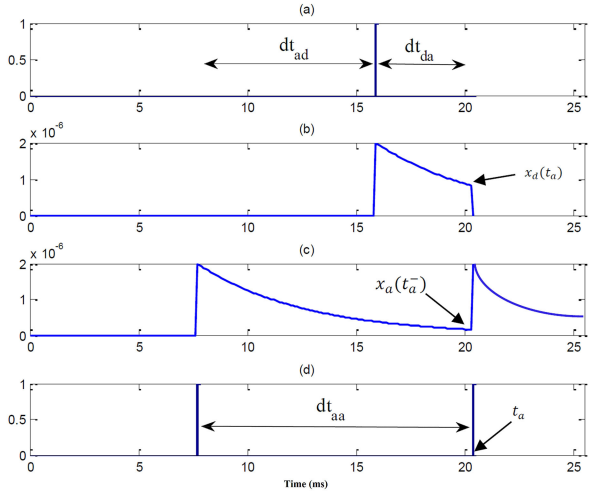


Fig. 4. (a) Desired spike. (b) Trace of a desired spike, $x_d(t)$: there is a jump to a constant value $A = 2 \times 10^{-6}$ at the time of a desired spike, then it decays exponentially. The value of x_d(t) is reset to zero at the time of an actual output spike. (c) Trace of an actual output spike train, x_a(t). (d) Actual output spike train.

Then $dt_{ad}(t_a)$ is calculated by $dt_{ad}(t_a) = dt_{aa}(t_a) - dt_{\mathrm{da}}(t_a)$. This procedure can be performed for each desired spike in the desired spike train.

The time interval of an actual output spike at time t_a and its previous desired spike, $dt_{\mathrm{da}}(t_a)$, can be calculated by the value of the nearest previous desired spike trace, $x_d(t_a)$, at the time of the actual output spike, t_a. The trace $x_d(t)$ starts from the value of A at the time of the previous desired spike and it decays exponentially by the time constant τ as shown in (1). By considering (1), the time interval, $dt_{\mathrm{da}}(t_a)$, between an actual output spike and its previous desired spike can be calculated by

$$dt_{\mathrm{da}}(t_a) = -\tau \ln\left(\frac{x_d(t_a)}{A}\right) \tag{3}$$

where $dt_{\mathrm{da}}(t_a)$ is the time interval between an actual output spike at time t_a and its previous desired spike. In other words, $dt_{\mathrm{da}}(t_a)$ is the time interval of the desired spike and the first actual output spike after the desired spike. $x_d(t_a)$ is the value of the desired spike trace at the time of the actual output t_a. The time interval $dt_{aa}(t_a)$ is calculated by

$$dt_{aa}(t_a) = -\tau \ln\left(\frac{x_a(t_a^-)}{A}\right) \tag{4}$$

where $x_a(t_a^-)$ is the momentary value of the actual output trace at the time of t_a just before jumping to the saturation value A. Thus, $dt_{ad}(t_a)$, which is the time interval of the nearest actual output spike before the desired spike, can be calculated by

$$\begin{aligned} dt_{ad}(t_a) &= dt_{aa}(t_a) - dt_{\mathrm{da}}(t_a) \\ &= -\tau Ln\left(\frac{x_a(t_a^-)}{A}\right) + \tau Ln\left(\frac{x_d(t_a)}{A}\right) \\ &= \tau Ln\big(x_d(t_a)/x_a(t_a^-)\big). \end{aligned} \tag{5}$$

At the time of the first spike after a desired spike, t_a, first $dt_{\mathrm{da}}(t_a)$, and $dt_{ad}(t_a)$ are calculated by (3) and (5), respectively. Then they are compared and the nearest actual spike time interval to the desired spike is determined as $dt_{\min}$. If the time interval is smaller than a maximum shift, dt_M, the desired spike is shifted toward the nearest actual output spike. Therefore, the shift, $dt_{\mathrm{shift}}(f)$, applied to the fth desired spike can be calculated by

$$dt_{\mathrm{shift}}(f) = \begin{cases} dt_{\min}(f), & dt_{\min}(f) \le dt_M \\ 0, & dt_{\min}(f) > dt_M \end{cases} \tag{6}$$

where $dt_{\min}(f)$ is the time interval of the closest actual output spike to the fth desired spike. If the nearest actual spike train is far from the desired spike, i.e., its time distance from the desired spike is larger than dt_M, the desired spike does not shift, and it waits for a weight learning procedure to generate an actual output spike close to the desired spike.

The desired spike train shift, $dt_{\mathrm{shift}}(f)$, is scaled to a small value at the start of the learning and it grows during the learning. At the beginning of the learning phase, because the weights are not well trained, the actual output spike train may be far from the desired spike train. Therefore, during the early epochs of the learning phase the desired spike shift is restricted to small values. The weight learning algorithm has higher impact on learning than the desired spike shift, when the desired spike shift is limited in the earlier epochs. The weight adjustment trains the neuron to fire at the desired time or a time close to the desired times. At the last epochs of the learning some of the desired spikes are trained and it might be difficult for the neuron to fire at the other desired times. In this situation the limitation on the desired output spike shift should be reduced to move the desired (target) spike toward the existing actual output spikes. To achieve this aim after each learning epoch and calculation of $dt_{\mathrm{shift}}(f)$, the shift is scaled by the epoch number as shown in

$$dt_a(f, e) = dt_{\mathrm{shift}}(f) \times \frac{e}{50} \tag{7}$$

where e is the number of the current epoch, $dt_a(f, e)$ is the time shift that is, applied to the fth desired spike at epoch number e. The denominator of (7) is set to 50 because the learning is continued until epoch 50. Equation (7) can be refined for higher number of learning epochs by setting the denominator of (7) with the new higher number of learning epochs. Thus, in epoch 50 the exact calculated shift from (6), $dt_{\mathrm{shift}}(f)$, is applied to the desired spikes.

The main strategy of learning in the proposed method is to place the focus on weight learning during the beginning of the training process. The effect of the desired spike shift increases during the final epochs of the learning process when the weight learning has stabilized. In the proposed method, first, learning of spatio–temporal pattern is performed by weight adjustment and the impact of a desired spike shift is gradually increased as the number of epochs increases.

At the start of the learning process, when $e = 1$, the applied shift, $dt_a(f, e)$, has a small value which is one fiftieth of $dt_{\mathrm{shift}}(f)$, i.e., $dt_a(f, e) = dt_{\mathrm{shift}}(f) \times (1/50)$ [see (7)]. When the epoch number, e, increases, the weights are updated, and the neuron learns to spike closer to the desired spikes.

In the last learning epoch, where $e = 50$, all weight learning is completed and there are no more learning epochs to train the neuron weights to fire at that desired times. In this stage, during the last epoch of the learning process, and after the weight learning procedure, the learning is completed by applying all the required desired spike shifts using (7), and $dt_a(f, e) = dt_{\text{shift}}(f) \times (50/50) = dt_{\text{shift}}(f)$.

The neuron weights are adjusted according to the ReSuMe [7] learning method as follows:

$$\frac{dw_i(t)}{dt} = \left[s_{\tilde{d}}(t) - s_o(t)\right]\left[a + \int_0^{+\infty} T_w(s) s_i(t-s) ds\right] \quad (8)$$

where w_i is the synaptic weight of the ith synapse, $s_{\tilde{d}}(t)$ is an adapted desired spike train that is, updated after each learning epoch (note that an adapted desired spike train is a shifted version of the initial suboptimal desired spike), $s_o(t)$ is actual output desired spike train, the parameter a is the ReSuMe non-Hebbian term and is a constant value, and $s_i(t)$ is the ith input spike train in the spatio–temporal input pattern. $T_w(s)$ is the ReSuMe learning window, and it is an exponential function as shown in

$$T_w(s) = \begin{cases} Ae^{-s/\tau}, & \text{for } s \geq 0 \\ 0, & \text{for } s < 0 \end{cases} \quad (9)$$

where A is a constant parameter and is the amplitude of the learning window. It has same value as the parameter A used in (1) $T_w(s)$, similar to STDP learning window has an exponential function that decays with a time constant τ.

B. Correlation-Based Metric

The correlation-based metric (C) proposed in [24] is used to evaluate the similarity between two spike trains to assess the performance of the proposed method. The metric is used to evaluate the similarity between the actual output of a neuron and its desired spike train. Additionally, it is used in a classification task. In the classification task a spiking neuron is trained to map different input patterns to their corresponding desired spike trains. Each spatio–temporal input pattern is assigned to the class with the desired spike train that has the highest correlation with the actual output of the neuron compared to the other classes. Then the classification accuracy is calculated based on the number of correct assignments.

The similarity coefficient C is equal to one for two identical spike trains, and it is zero for two completely uncorrelated spike trains. Therefore, the closer the value of C to one, the greater the similarity between two spike trains. C is calculated as follows:

$$C = \frac{v_d \cdot v_o}{|v_d||v_o|} \quad (10)$$

where v_d and v_o are two vectors and they are the convolution of a desired spike train, $s_d(t)$, and an actual output spike train, $s_o(t)$, by a symmetric Gaussian filter, respectively. The spike trains which consist of series of Dirac delta functions are converted to continuous functions. "$v_d.v_o$," the numerator of (10), is the inner product of the two vectors, and $|v_d|$ and $|v_o|$ represent the length of the vectors v_d and v_o, correspondingly.

TABLE I
HYPER-PARAMETERS OF THE PROPOSED ADAPTED DESIRED SPIKE TRAIN

Hyper–parameter	Equation	Value
τ	(1), (2), (3), (4), (5), (9)	1 ms
A	(1), (2), (3), (4), (5), (9)	2×10^{-6}
dt_M	(6)	[2, 10] ms
dt	(8)	0.1 ms
δ	(11)	2 ms

The symmetric Gaussian function which is convolved with the spike trains is given by

$$f(t, \delta) = e^{\frac{-t^2}{2\delta^2}} \quad (11)$$

where the parameter δ is a constant value and determines the width of the symmetric Gaussian function.

III. RESULTS

Experiments were performed to investigate the effect of the proposed method for shifting desired spikes and optimizing the output encoding. ReSuMe is used to train synaptic weights. Two learning tasks are considered. The first learning task is to map a random spatio–temporal input pattern to a desired spike train. The spatio–temporal input pattern and the desired spike train was produced by a random Poissonian process. The spatio–temporal input pattern contains 200 spike trains with 20 HZ (or 15 Hz where mentioned) mean spiking frequency, and the frequency of the desired spike train is set to 100 Hz. The time duration of the spike trains is 650 ms. The experiment results for the first learning task are described in Sections III-A and III-B. For the first learning task, the correlation between actual output of a trained neuron and its desired spike train is calculated using (10) and reported to determine the accuracy of the proposed SNN. The second learning task concerns the classification of spatio–temporal input patterns, and the results of the classification task are described in Section III-C. In the classification a spiking neuron is trained to map different input patterns to their corresponding desired spike trains. In this case, each spatio–temporal input pattern is assigned to the class with the desired spike that has the highest correlation with the actual output of the neuron compared to the other classes. Then the classification accuracy is calculated based on the number of correct assignments.

The proposed learning algorithm has four hyper-parameters that should be set before learning. The four hyper-parameters are shown in Table I. The second column in Table I shows the equations that use the hyper-parameters. In this paper, the simulation time step is 0.1 ms, $dt = 1$ ms. The other variables in the equations are automatically generated during usual activity of the spiking neuron when it fires actual output spikes in response to an applied spatio–temporal input pattern and they also depend on the desired spike train. Note that the hyper-parameter A is dimensionless, and it shows the ratio of two weight adjustment. This section discusses the results of three

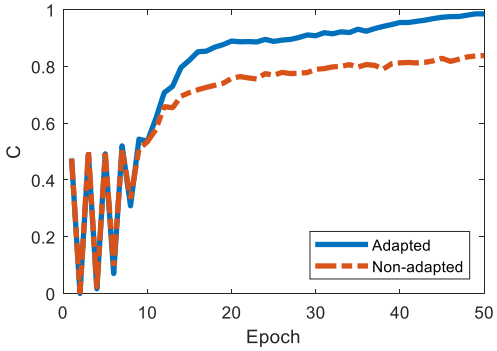


Fig. 5. Comparison of the performance of ReSuMe during various learning epochs when non-adapted desired spike and adapted desired spike are used ($T = 650$; Fin $= 20$ Hz, Fo $= 100$ Hz, and $dt_M = 10$ ms).

sets of experiments. In the first experiment, the effect of the shifting of the desired spikes on the accuracy of the learning method is investigated for the first learning task, i.e., mapping a spatio–temporal input pattern to a desired spike train. Then the effect of the maximum allowable shift, dt_M, is presented. Finally, the classification accuracy of the proposed method for the second learning task is reported.

A. Effect of the Proposed Spike Shifting Method on the Learning Efficiency of SNNs

In this section, the proposed method is used for mapping a spatio–temporal input pattern to a desired spike train. Two experiments were carried out to determine the effect of the proposed spike shifting method on the accuracy of SNNs. In the first experiment, a neuron was trained, as a benchmark comparison, with a non-adapted desired spike train. In the second experiment, the suboptimal desired spikes are shifted during the learning process such that each spike in the desired spike train is shifted toward the nearest actual output spike using the proposed method.

The experimental results are shown in Fig. 5. In the plot presented in Fig. 5, the y-axis shows the correlations between the neuron's actual output spike trains and the desired spike trains for the first learning task. The desired spike shift causes the performance of ReSuMe to reach a correlation level of 0.99 at epoch 50. However, the correlation is 0.84 at epoch 50 when a non-adapted desired spike train is used. The non-adapted method cannot reach the correlation level of the adapted method for a comparatively high number of training epochs. In the experiments, the non-adapted method reached its maximum correlation level of 0.86 at 500 learning epochs, whereas the adapted method reached a correlation value close to 1.00 in only 50 learning epochs. The experimental results show that the adapted desired spike method increases the performance of the learning and changes the pattern of the desired spike train. Fig. 6 illustrates the desired spike trains before and after 15 learning epochs. The similarity between these two spike trains (the initial suboptimal desired spike train and the shifted desired spike train) can be calculated by the correlation-based metric method, C. The calculated C is 0.80.

Fig. 7 shows the similarity between the adapted desired spike and the initial suboptimal desired spike train

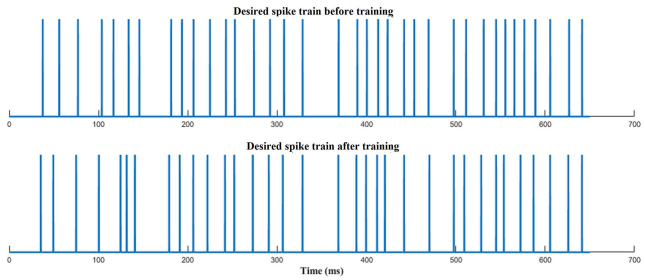


Fig. 6. Desired spike train before and after 15 learning epochs. C value that shows the similarity between the two spike trains is 0.80414.

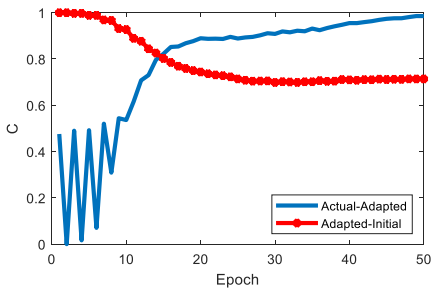


Fig. 7. Actual-Adapted: the correlation between the actual output spike train and the adapted desired spike train during various learning epochs. Adapted-Initial: the correlation between adapted desired spike train and the initial suboptimal desired spike train during various learning epochs.

(Adapted-Initial) during various learning epochs. The experiment is run 20 times and the mean values are reported. At the start of learning, similarity is 1.00 (i.e., the adapted desired spike train is the same as the initial suboptimal desired spike train), and at each learning epoch the spikes in the adapted desired spike train are shifted. During the shift, the similarity between the adapted desired spike train and the initial suboptimal desired spike train changes. Additionally, Fig. 7 shows the similarity between the neuron actual output spike train and the adapted desired spike train during various epochs.

B. Effect of the Maximum Allowable Shift

In the next experiment, the maximum allowable shift for the desired spikes, dt_M, is reduced from 10 to 1 ms. The result is shown in Fig. 8. The correlation between the actual output and the adapted desired spike train, which is shown by the legend [Adapted (1 ms)] in Fig. 8, is reduced compared to the correlation value which is shown in Fig. 5 where dt_M was 10 ms. The reduction of dt_M prevents the high modification of desired spike times. When $dt_M = 1$ ms, the weight adjustment has more contribution in the generation of actual output spikes at the desired times, compared to the previous situation where dt_M was 10 ms. The low dt_M is closer to the situation that the desired spike train is non-adapted and the generation of actual output spikes relies more on weight adjustment, and similarly it has lower accuracy. The maximum allowable shift controlled by dt_M gives a freedom to the neuron to stop its weight adjustment and consequently the learning is stabilized, and it prevents the learning algorithm to make more weight adjustment which consequently increases the accuracy of the method. The adapted desired spikes stay close to the initial suboptimal desired spike train when dt_M is reduced to 1 ms.

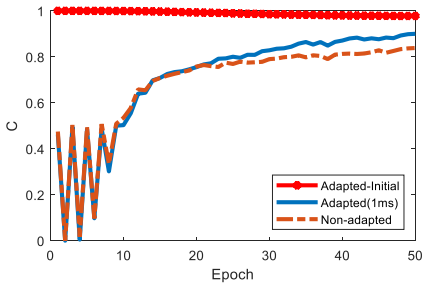


Fig. 8. Adapted-Initial: the correlation between the adapted desired spike train and initial suboptimal desired spike train when the desired spike train evolves in various learning epochs. Adapted (1 m): The correlation metric, C, between the adapted desired spike train and the neuron actual output spike train across various learning epochs when the maximum allowable shift for the desired spike is reduced to 1 ms. Non-adapted: is the C value of the neuron actual output when non-adapted desired spike train is used. ($T = 650$; Fin $= 20$ Hz, Fo $= 100$ Hz, and $dt_M = 1$ ms.)

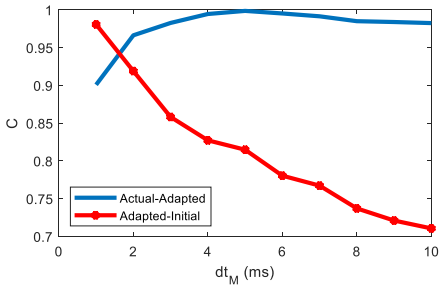


Fig. 9. Correlation between the actual output and adapted desired spike trains for different values of dt_M.

In Fig. 8, the curve with the Adapted-Initial legend shows that the correlation between the learned adapted spike train and the initial suboptimal desired spike train is higher than the situation that $dt_M = 10$ ms (Fig. 7). Therefore, there is a smaller change in the learned desired spike compared to the initial suboptimal desired spike train.

Fig. 9 shows the correlation between the actual output and adapted desired spike trains for different values of dt_M in [1, 10] interval. The figure shows that the correlation is increased until $dt_M = 5$ ms. Any further increase to dt_M than 5 ms reduces the correlation between the neuron actual output and the adapted desired spike trains. Because a high value for dt_M causes a high desired spike shift and consequently reduces the correlation between the adapted desired spike train and the initial suboptimal desired spike train. The high shift in the desired spike requires new weight values which are different from the previously learned weights, and it reduces the correlation. Fig. 9 also shows the correlation between the adapted desired spike train and the initial suboptimal desired spike (Adapted-Initial) for different values of dt_M. The correlation is reduced when dt_M is increased.

The histogram of the initial synaptic weights before learning is shown in Fig. 10(a), the initial weights are normal random number with the mean value and standard deviation of 1.2×10^{-5}. Fig. 10(a) shows that a lower portion of the weights are negative (inhibitory inputs) and a higher portion of weights are positive (excitatory input). Fig. 10(b) shows the histogram of the synaptic weights after training when dt_M is set to 5 ms. Fig. 10(b) shows that the weights after training

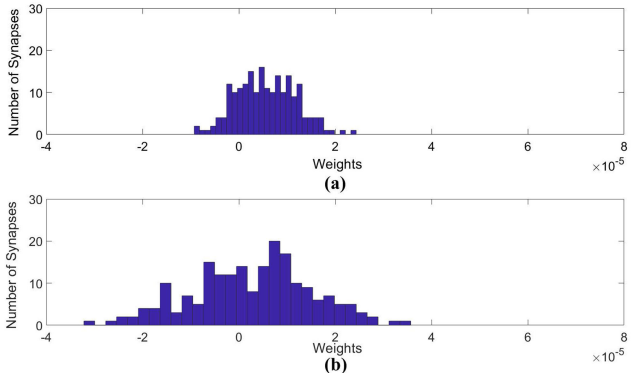


Fig. 10. Histogram of the initial synaptic weights when dt_M is set 5 ms. (a) Before learning. (b) After learning.

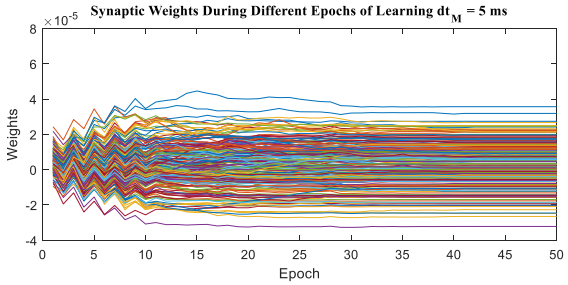


Fig. 11. Evolution of 200 synaptic weights during different learning epochs when $dt_M = 5$ ms.

are distributed around the initial weigh values as demonstrated in Fig. 10(a). Fig. 11 shows the evolution of weights during 50 learning epochs. Fig. 11 shows that almost all the weights are trained after about 45 learning epochs, and hence the weights stabilized at 45 epochs. In order to investigate the effect of dt_M on weight learning, the histogram of weights before and after learning when the desired spikes are non-adapted, i.e., $dt_M = 0$, are shown in Fig. 12. The weights after learning are in the range $[-3.88 \times 10^{-5}, 6.99 \times 10^{-5}]$ when $dt_M = 0$, whereas the weights are in the range $[-3.22 \times 10^{-5}, 3.57 \times 10^{-5}]$ when $dt_M = 5$ms. These results revealed that the weights are in a wider interval when $dt_M = 0$, when the desired spikes are non-adapted. However, the weights during training of an optimal desired spike train obtained by $dt_M = 5$ ms needs less weight adjustment. Fig. 13 shows the evolution of weights during different learning epochs when the desired spikes are non-adapted, i.e., $dt_M = 0$. The figure shows that quite a number of the weights keep adapting even after the 50th epoch.

Fig. 14 illustrates the correlation between the actual output spike train of the trained neuron and the initial suboptimal desired spike train (Actual-Initial) during various learning epochs when the adapted desired spike method is used. Fig. 14 also shows the correlation between the actual output spike train and the initial suboptimal desired spike train when non-adapted desired spike train is used. The experimental result shows that when dt_M is set to 1 ms, the adapted desired spike increases the performance of ReSuMe, to learn initial suboptimal desired spike train. It increases the correlation metric from $C = 0.84$ to $C = 0.88$. The output spike optimization method brings a desired spike to the time of an actual output

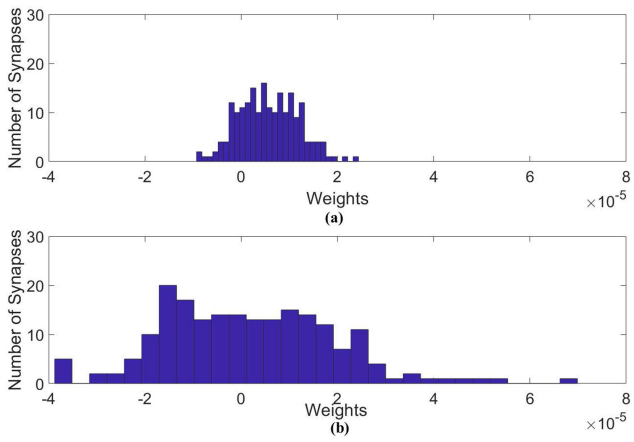


Fig. 12. Histogram of the initial synaptic weights when non-adapted desired spikes are used, i.e., $dt_M = 0$. (a) Before learning. (b) After learning.

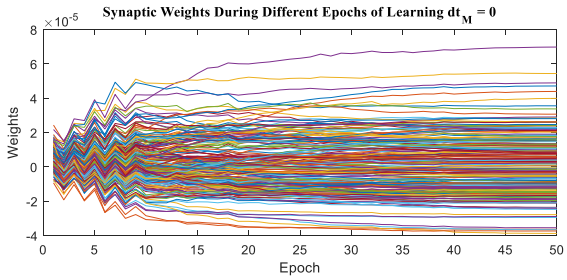


Fig. 13. Evolution of 200 synaptic weights during different learning epochs when the non-adapted desired spikes are used, i.e., $dt_M = 0$.

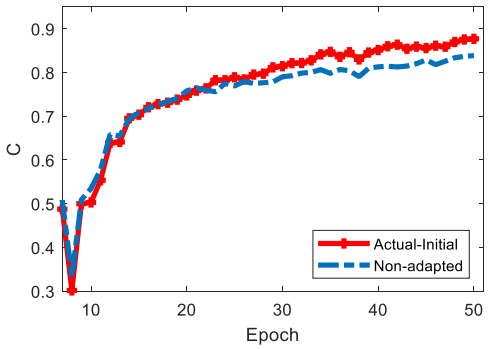


Fig. 14. Actual-Initial: correlation, C, between actual output spike train and initial suboptimal desired spike train when ReSuMe uses the adapted desired spike train method. Non-adapted: correlation, C, between actual output spike train and the non-adapted desired spike train when ReSuMe is used to train a neuron by the non-adapted desired spike train.

spike which is close to the desired spike. This prevents the weight adjustment related to generation an output spike at the time of the desired spike through weight learning, and it prevents the weight adjustment related to the cancelation of the nearby actual output spike. Consequently, the desired spike shift settles the weight adjustment and prevents extra adjustment of weights that may interfere in the learning of the other desired spikes, at the cost of acceptance of a small error.

C. Classification of Spatio–Temporal Input Patterns

In this section, the performance of the proposed method is investigated within a classification task. In the classification

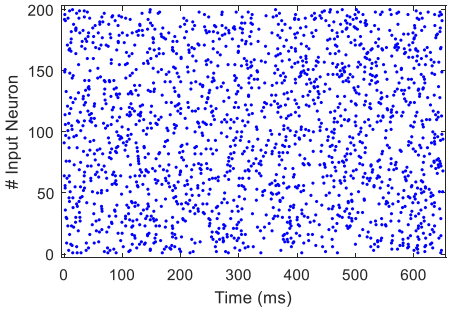


Fig. 15. Raster plot of a spatio–temporal input pattern from the first class. There are 200 spike trains in the spatio–temporal pattern generated by 200 input neurons.

task, two sets of synthetic spatio–temporal spiking input patterns are used. Each spatio–temporal input pattern composed of several spike trains. The two sets of spatio–temporal patterns belong to two different classes. The two sets of spatio–temporal input patterns are generated by Poissonian process with two different spike rates. The spatio–temporal input patterns of the first class has the same property of the spatio–temporal input patterns which are used in the previous experiment described in the beginning of Section III-A random Poissonian process with the mean spike rate of 20 Hz is used to generate the spatio–temporal input patterns of the first class. The input patterns of the second class are random patterns with the rate of 15 Hz. Fig. 15 shows the raster plot of a spatio–temporal input pattern from the second class. The spatio–temporal input pattern is composed of 200 spike trains. The spike trains are generated by 200 input neurons. Additionally, two random desired spike trains are assigned as the label of the two classes of spatio–temporal input patterns. A spiking neuron is trained to assign each input pattern to its corresponding desired spike train.

In the experiments, described in this section, classification performance was initially evaluated when the neuron was trained on non-adapted desired spike trains. Then the neuron was trained on the proposed adapted desired spike trains. The proposed method is used to shift desired spikes for the different classes to find optimum times for the desired spikes for the two classes, and classification performance is compared when using the proposed adapted desired spikes versus the non-adapted desired spikes.

1) Experiment Results on Two Random Spatio–Temporal Input Patterns Per Class: In the first experiment, each class has a set of spatio–temporal input patterns composed of two spiking patterns. An SNN is trained to assign each pattern to its corresponding desired spike train. The results achieved by the non-adapted desired spike train and the proposed adapted desired spike train method are shown in Table II. The results show that the proposed adapted desired spike method can shift the initial suboptimal desired spikes to appropriate desired times with higher accuracy than the method using non-adapted desired spike trains, and consequently it increases classification accuracy by 10%.

2) Experimental Results on Spatio–Temporal Input Patterns Generated by Time Jitters: A similar method used in [25] is used to generate a higher number of training and testing

TABLE II
COMPARISON OF THE PROPOSED ADAPTED DESIRED SPIKE METHOD AGAINST THE METHOD WHICH USES A NON-ADAPTED DESIRED SPIKE TRAIN[A]

Method	Classification Accuracy
Proposed adapted desired spike train method	100.00%
non-adapted desired spike train method	90.00%

[A]Four spatiotemporal input patterns are generated by the Poissonian process

TABLE III
TRAINING AND TESTING CLASSIFICATION ACCURACY ON THE DATA THAT IS GENERATED BY ADDING NOISY JITTERS TO THE BASE PATTERNS

Method	Training accuracy	Testing accuracy
Proposed adapted desired spike train method	96.00%	96.75%
Non-adapted desired spike train method	88.25%	84.50%

samples. The four spatio–temporal input patterns generated by the random Poissonian process with frequencies of 20 and 15 Hz in Section III-C1 are considered as base patterns. Then a number of testing and training spatio–temporal patterns are generated by adding random jitters to the spike times of the base patterns. Each spike in a base spatio–temporal input pattern is moved by a random time jitter extracted from a uniform distribution on $[-2, 2]$ ms interval. In total the resulted training set contains 20 spatio–temporal input patterns, composed of the 4 base patterns and 16 noisy patterns. The test set is composed of other 20 noisy spatio–temporal patterns that have been generated by adding random jitter to the four base patterns. The task concerns training a spiking neuron to generate desired spikes corresponding to the class of an applied spatio–temporal input pattern. The simulation is continued for 50 learning epochs on the training dataset. The results are reported in Table III. The results reveal that the proposed method can improve the testing accuracy more than 12.25%.

3) Experimental Results on Different Numbers of Random Spatio–Temporal Input Patterns: In this set of experiments, more challenging tasks are considered. Instead of generating training and testing data by adding a noisy jitter to the base patterns, a number of training and testing patterns are directly generated by the Poissonian process with the firing rate of 15 and 20 Hz for the two different classes. Fig. 16 shows the mean value of classification accuracy when the number of spatio–temporal input patterns is increased from 4 to 20. Each classification task is repeated for 20 different runs and the mean values are reported in Fig. 16. The vertical error bars in Fig. 16 shows standard errors of the mean values. The results show that the proposed method has higher accuracy for the different numbers of spatio–temporal input patterns. When the number of the patterns increases, the difficulty of the learning task also increases. Consequently, the accuracy of the two methods shown in Fig. 16 is reduced when the number of spatio–temporal patterns increases. The improvement of the proposed method can reach more than 16% when the number

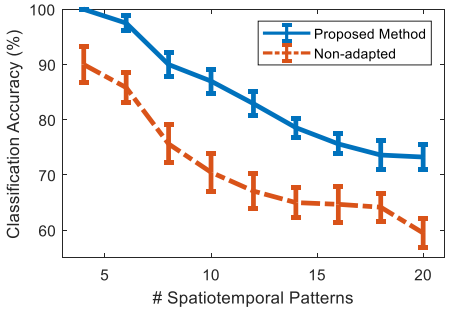


Fig. 16. Classification accuracy of the SNN when it uses the proposed method for finding the optimum desired spikes is higher than when it uses non-adapted desired spikes.

TABLE IV
NUMBER OF INPUT PATTERNS VERSUS IMPROVEMENT IN ACCURACY

Number of input patterns	Improvement in Accuracy (%) (Adapted)-(Non_adapted)
4	10.00
6	11.67
8	14.38
10	16.50
12	15.84
14	13.57
16	10.93
18	9.44
20	13.75

of the spatio–temporal patterns reaches 10 (see Table IV). The proposed method shifts the initial suboptimal desired spikes to appropriate times when different patterns are trained. At the end, the desired spikes can reach optimum times for all the trained input patterns depending on the distribution of spikes in the input patterns. The optimum times for desired spikes reduced required weight adjustment and consequently prevents distortion in the learning of previously trained patterns and increases the overall classification accuracy (see Fig. 16).

IV. DISCUSSION

The discontinuous nature of the activity of spiking neurons makes it difficult (or impossible) to use classical methods, such as the backpropagation learning algorithm, to train spiking neurons. One challenge in training a spiking neuron is that a spiking neuron cannot generate every possible encoding of desired output spike trains in response to a specific spatio–temporal input pattern. A spiking neuron can be trained to produce output spikes that are compatible with the distribution of its input spatio–temporal spikes. An SNN generates output spikes at specific times in response to an input pattern, provided the input spikes occur within a suitable time window. The input spikes in that time window cause a high level of post synaptic potential (PSP) and force the neuron to fire, ideally at the desired times. Training involves adjustment of the network parameters so as to ensure the output spikes occur at the correct times in response to such an input spike

train. However, if there are not enough input spikes in the relevant time window, the learning method has difficulties and the network parameter adjustment is increased without reaching the training aim. Comparison of Figs. 11 and 13 reveal that the weight adjustment stabilizes when the proposed method is used to find appropriate desired spikes. However, the neuron finds it difficult to learn the non-adapted desired spike train and the weights do not stabilize during all the training epochs (see Fig. 13). Additionally, the weight adjustments are constrained to be within a shorter interval when an appropriate desired spike is used to train the spiking neuron.

A learning algorithm for a spiking neuron adjusts learning parameters and forces the neuron to fire actual output spikes at desired times. On the other hand, training a neuron to fire in some specific times when there are no input spikes in the time windows around the desired times can reduce the performance of a learning algorithm. For instance, consider the situation where there is a large number of spikes in the input spatio–temporal pattern within a short time interval. If there are no desired output spikes corresponding to that time interval, an undesired output spike may be generated. If the learning algorithm adjusts the learning parameters to remove such an undesired spike, it may adversely interfere with learning of other desired output spikes. Fig. 13 shows that weights with negative value are continuously increased when the number of learning epochs is increased. This negative growth of the weights is the learning method's reaction to removing the undesired output spikes which are generated because of the high number of input spikes in the undesired time interval. Additionally, a neuron may have difficulty producing a desired output spike where there are not enough input spikes shortly before the desired spike. A very low number of input spikes leads to a low level of PSP at the desired time, which consequently causes difficulty in the generation of an actual output spike at the desired time. Fig. 13 shows that ReSuMe learning method continuously increases the weights to generate desired spikes that do not have input spike close to desired times without reaching the learning goal. Therefore, an inappropriate selection of a desired (target) output spike train (i.e., output encoding) for a spatio–temporal input pattern might lead to low performance or convergence problems for the SNN learning algorithm.

Considering Fig. 15, the total time length of the spatio–temporal spiking pattern is 650 ms, and the spatio–temporal input pattern has 200 spike trains. The distribution of spikes in the spatio–temporal input pattern shows that the concentration of the spikes is high in some parts and sparse in other parts. This distribution changes depending on the class to which the pattern belongs and can be considered as characteristic of the class. In other words, each class can be identified according to the distribution of spikes in its spatio–temporal input patterns. The critical point is that, based on the distribution of spikes in the spatio–temporal input pattern, the learning ability of a neuron is different for different desired output spike trains, i.e., how the output is encoded in spike times.

Spike times are the most crucial part of information transmission in SNNs. The proposed method optimizes the performance of an SNN through improving the learning process. The experimental results show that the original learning algorithm, ReSuMe, used for training the suboptimal desired spike train, could not learn all the desired spikes in the suboptimal desired spike train precisely. This was because weight adjustment could not be stabilized as less relevancy existed between the input and the suboptimal desired output spikes; which caused continuous weight change and consequently reduced the correlation of the actual output spikes of the trained neuron with the initial suboptimal desired spike train. However, the shift of the desired spikes around the times of the original desired spikes stabilizes the learning process of the neuron, and the shift prevents continuous changes of learning parameters, i.e., weights, and consequently increased the correlation between the actual output and the original desired spikes. Fig. 14 shows that the correlation between the actual output of the neuron with the initial desired spike train when the proposed method is used to shift the desired spikes is higher than when the proposed method is not used.

In the proposed method, the desired spike can be adjusted around the original desired spike train, and this shift provides flexibility to stabilize the learning and consequently to increase the accuracy. Multispike Tempotron [26] also provides flexibility by permitting a neuron to fire at any time in a specified window to enable fast conversion.

The proposed spike shift method does not rely on any specific weight learning algorithm, but instead has its separate procedure which performs in parallel with the weight learning method. In each learning epoch, a desired spike train is considered as a fixed desired spike train for the weight learning method, and therefore the learning method can work as usual. At the end of each learning epoch, updates are made for the desired spikes, then they become fixed for the next learning epoch for the weight learning algorithm. Therefore, any other learning rules like SPAN [11], or the linear algebraic method [27] which perform weight learning in different epochs are compatible with the proposed method.

Noisy spike patterns are common in SNNs. A time jitter can be added to a base spike pattern to generate noisy spike patterns. In Section III-C2, a time jitter has been added to generate noisy testing data which was used to test the performance of the proposed method on noisy data. Table III shows that the proposed method can achieve higher accuracy for noisy data compared to the non-adapted desired spike train method.

V. Conclusion

In this paper, a method is proposed to adaptively adjust a desired spike train. The experimental results show that a spiking neuron can learn adjusted desired spikes with significantly high accuracy. For instance, it can increase the correlation level from 0.84 to 0.99. A biologically plausible local variable called spike trace is used to calculate the required shift for desired spikes in different epochs for different spikes. The desired spike trace and actual output spike trace are used to find the appropriate shift for each desired spike. The proposed method calculates the time interval of the nearest actual output spike before and after a desired spike and compares these time intervals and finds the nearest actual spike to a desired spike using time varying spike trace. Selection of

a low value for the maximum allowable shift for desired spikes improves the performance of the algorithm for learning of not only adapted desired spike train but also the initial suboptimal desired spike train. For instance, the proposed method can increase the accuracy of ReSuMe on a suboptimal desired spike train by 4%. Small shifts in the desired spike train help the neuron weight adjustment to settle and prevent unnecessary weight adjustment and distraction of previously trained weights. In this paper, the adapted desired spike train learning method is applied to the ReSuMe weight learning method, however, the proposed method can be applied to other learning methods for SNNs.

The higher performance of the adapted desired spike train method can be used to improve the classification ability of SNNs. For instance, in a classification task, output spike train encodings act as labels for different classes. The proposed method can be used to find the optimum output spiking pattern for different classes and to increase the performance of classification tasks. SNNs can learn optimal desired spike trains which are compatible with its spatio–temporal input patterns. The network can learn each spatio–temporal input pattern with less weight adjustment and it can achieve a higher accuracy with the proposed desired spike shift method. In a classification task there are not many restrictions because the encoding can be arbitrary provided the classes can be distinctly segregated.

This paper proposes a method to find appropriate desired spike trains based on a classification problem. Although finding appropriate desired spike trains for different classes increases the classification accuracy, it has not been investigated significantly in previous studies, with the result that arbitrary spikes at precise times or uniformly spaced spike trains are usually used in classification tasks which reduces their accuracy. The arbitrary or uniformly spaced spikes may not be compatible with the input spike trains, and it is not possible for a neuron to generate them. The proposed method adds a degree of flexibility to the desired spike times and it leads to faster convergence and improvement of the accuracy of the learning algorithm. The proposed method can be used to improve classification accuracies of different SNNs such as the SNNs proposed in [28]–[30].

REFERENCES

[1] 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.

[2] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds. Berlin, Germany: Springer, 2012, pp. 335–376. [Online]. Available: https://link.springer.com/referencework/10.1007/978-3-540-92910-9

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

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

[6] 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.

[7] 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.

[8] S. McKennoch, D. L. D. Liu, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm," in *Proc. IEEE Int. Joint. Conf. Neural Netw.*, 2006, pp. 3970–3977.

[9] 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, Aug. 2007.

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

[11] 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, 2012, Art. no. 1250012.

[12] M. Zhang, H. Qu, A. Belatreche, and X. Xie, "EMPD: An efficient membrane potential driven supervised learning algorithm for spiking neurons," *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 2, pp. 151–162, Jun. 2018.

[13] A. Taherkhani, A. Belatreche, Y. Li, and L. Maguire, "A new biologically plausible supervised learning method for spiking neurons," in *Proc. ESANN Eur. Symp. Aritif. Neural Netw. Comput. Intell. Mach. Learn.*, Apr. 2014, pp. 11–16.

[14] 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. ICONIP 2015* (LNCS 9490), S. Arik, T. Huang, W. Lai, and Q. Liu, Eds. Cham, Switzerland: Springer, 2015. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-26535-3_22#citeas

[15] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5394–5407, Nov. 2018.

[16] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nat. Neurosci.*, vol. 9, no. 3, pp. 420–428, Feb. 2006.

[17] 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, no. 1995, pp. 188–201, 2013.

[18] Q. Yu, H. Tang, J. Hu, and K. C. Tan, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *Intell. Syst. Ref. Lib.*, vol. 126, no. 10, pp. 19–41, 2013.

[19] 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.

[20] R. Legenstein, C. Naeger, and W. Maass, "What can a neuron learn with spike-timing-dependent plasticity?" *Neural Comput.*, vol. 17, no. 11, pp. 2337–2382, 2005.

[21] 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.

[22] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[23] 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.

[24] S. Schreiber, J. M. Fellous, D. Whitmer, P. Tiesinga, and T. J. Sejnowski, "A new correlation-based measure of spike timing reliability," *Neurocomputing*, vols. 52–54, pp. 925–931, Jun. 2003.

[25] 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.*, Jul. 2015, pp. 1–7.

[26] R. Gütig, "Spiking neurons can discover predictive features by aggregate-label learning," *Science*, vol. 351, no. 6277, pp. 272–283, 2016.

[27] A. Carnell and D. Richardson, "Linear algebra for time series of spikes," in *Proc. ESANN*, Apr. 2005, pp. 363–368.

[28] 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, Feb. 2013.

[29] N. Anwani and B. Rajendran, "NormAD—Normalized approximate descent based supervised learning rule for spiking neurons," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2015, pp. 1–8.
[30] S. R. Kulkarni and B. Rajendran, "Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization," *Neural Netw.*, vol. 103, pp. 118–127, Jul. 2018.

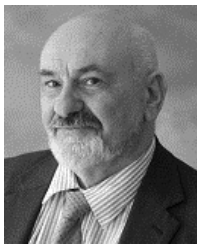
Aboozar Taherkhani (M'17) received the B.Sc. degree in electrical and electronic engineering from Shahid Beheshti University, Tehran, Iran, the M.Sc. degree in biomedical engineering from the Amirkabir University of Technology, Tehran, and the Ph.D. degree from Ulster University, Londonderry, U.K., in 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, deep neural network, spiking neural network, complex system theory, and nonlinear signal processing.

Georgina Cosma (M'14) received the B.Sc. degree (First Class Hons.) in computer science from Coventry University, Coventry, U.K., in 2003 and the Ph.D. degree in computer science from the University of Warwick, Coventry, in 2008.

She is currently an Associate Professor with Nottingham Trent University, Nottingham, U.K. Her current research interests include data science, computational intelligence, nature-inspired and deep feature selection, feature extraction, and conventional machine learning and deep learning algorithms.

Dr. Cosma is a member of the IEEE Computer Society with Computational Intelligence, Big Data Community, and Brain Community memberships. She is a Principal Investigator of the Leverhulme Trust project grant titled "Novel Approaches for Constructing Optimised Multimodal Data Spaces."



Thomas Martin McGinnity (SM'09) received the B.Sc. degree (First Class Hons.) in physics from the New University of Ulster, Coleraine, U.K., in 1975 and the Ph.D. degree from the University of Durham, Durham, U.K., in 1979.

He currently holds a part-time Professorship with the Department of Computing and Technology, Nottingham Trent University (NTU), U.K., and the School of Computing, Engineering and Intelligent Systems, Ulster University, Londonderry, U.K. Before taking semi-retirement, he was formerly the Pro Vice Chancellor and the Head of the College of Science and Technology with NTU. He has authored or coauthored over 330 research papers and has attracted over £25 million in research funding. His current research interests include computational intelligence, computational neuroscience, modeling of biological information processing in software and reconfigurable hardware and cognitive robotics.

Professor McGinnity is a fellow of IET.