SOLSA: Neuromorphic Spatiotemporal Online Learning for Synaptic Adaptation

Zhenhang Zhang, Jingang Jin, Haowen Fang, Qinru Qiu
Department of Electrical Engineering and Computer Science, Syracuse University
{zzhan281, jjin24, hfang02}@syr.edu, qinru.qiu@gmail.com

***Abstract*—Spiking neural networks (SNNs) are bio-plausible computing models with high energy efficiency. The temporal dynamics of neurons and synapses enable them to detect temporal patterns and generate sequences. While Backpropagation Through Time (BPTT) is traditionally used to train SNNs, it is not suitable for online learning of embedded applications due to its high computation and memory cost as well as extended latency. In this work, we present Spatiotemporal Online Learning for Synaptic Adaptation (SOLSA), which is specifically designed for online learning of SNNs composed of Leaky Integrate and Fire (LIF) neurons with exponentially decayed synapses and soft reset. The algorithm not only learns the synaptic weight but also adapts the temporal filters associated to the synapses. Compared to the BPTT algorithm, SOLSA has much lower memory requirement and achieves a more balanced temporal workload distribution. Moreover, SOLSA incorporates enhancement techniques such as scheduled weight update, early stop training and adaptive synapse filter, which speed up the convergence and enhance the learning performance. When compared to other non-BPTT based SNN learning, SOLSA demonstrates an average learning accuracy improvement of 14.2%. Furthermore, compared to BPTT, SOLSA achieves a 5% higher average learning accuracy with a 72% reduction in memory cost.**

Keywords*—*Spiking Neural Network, Spatiotemporal pattern learning, online learning.

I. INTRODUCTION

Unlike conventional artificial neural networks (ANNs), Spiking Neural Network (SNNs) [1][2] have the unique ability to retain past information within the membrane potential. The membrane potential is considered as the state of neurons and synapses. The changes of the membrane potential over time, which are influenced by both current and historical inputs, are referred to as temporal dynamics. Many SNN models, such as the widely used Leaky Integrate and Fire (LIF) model [3], calculate membrane potential as a leaky integration of the input with exponential decay. Once the membrane potential reaches a certain threshold, the neuron fires a spike and resets its membrane potential. In a more advanced model proposed by [4], called spatial temporal (ST) SNN, the temporal dynamics of synapses are also considered. It has been demonstrated both biologically [5] and computationally [6] that the temporal dynamics in neurons and synapses play a crucial role in enabling the network to perform spatiotemporal operations.

A basic method of SNN training is to update the weights based on the timing of pre- and post-synaptic neuron spiking events [7]. When a pre-synaptic spiking event happens before post-synaptic spiking event within a time window, this leads to a potentiation of the synaptic weight. A contrast condition leads to a depression of weight. This weight update rule is also called Hebbian Learning Rule [8]. Vanilla Hebbian Rule is usually slow and less effective compared to the backpropagation algorithm used in ANN training [1].

By modeling the temporal dynamics of neurons and synapses as filters with feedback connections [4], an SNN is seen as a recurrent neural network (RNN) and can be trained using Backpropagation Through Time (BPTT) [9][14]. This algorithm unrolls the network along the temporal axis and applies conventional backpropagation to the unrolled network. The gradients need to propagate from the output layer all the way back to the input layer and from present time back to the time when the input sequence begins. Therefore, BPTT requires a significant amount of memory to store the historical activities of neurons. Additionally, the learning process cannot start until the entire input sequence has been received. This either leads to an extended processing time or a high workload surge at the end of each input sequence. These limitations make BPTT unsuitable to be applied for online learning on edge devices with limited hardware resources.

The limitations discussed above regarding BPTT have spurred a set of online training algorithms for SNN, including eligibility trace propagation (i.e., E-prop) [10], Online Training Through Time (OTTT) [11] and Deep Continuous Local Learning (DECOLLE) [12]. However, these algorithms ignore the temporal dynamics of the synapses [10][11] and some of them even overlook the reset process of the membrane potential [11]. Consequently, their performance is compromised, particularly for longer input sequences.

In this work we present SOLSA (Spatiotemporal Online Learning for Synaptic Adaptation), a learning algorithm to train SNNs with temporal filters on both synapses and neuron membrane potentials. SOLSA goes beyond learning synaptic weights. It can also update synapse filter kernels automatically. Our unique techniques such as scheduled weight update and early stop, allow SOLSA to converge faster and learn with better accuracy.

The uniqueness and benefit of the SOLSA learning is summarized as the following:

1. It does not record historical neuron activities for backpropagation over time, hence the memory complexity is much lower than that of BPTT.
2. It is capable of learning not only the synapse weights but also the synaptic filter kernels. Compared to the existing non-BPTT based learning algorithms, which only update

This project was partially funded under AFRL's FA8750-21-C-1511 contract and NSF I/UCRC ASIC Center (CNS-1822165).

the synapse weights, model learned by SOLSA can better adapt to the given spatiotemporal sequence.

3. Unlike BPTT, which updates the model only once at the end of each input sequence and blindly computes the gradient for the entire input sequence, the scheduled update and early stop techniques allows SOLSA to update the model more frequently and focus more on the signature patterns. Hence, it outperforms BPTT in terms of learning accuracy.
4. Our experimental results show that SNN models trained using SOLSA have an average of 14.2% higher accuracy in spatiotemporal pattern classification compared to models trained with existing non-BPTT algorithms. It also outperforms BPTT by improving the classification accuracy by 5% while requiring 72% less memory.

The rest of the paper is organized as following. In Section II, we will introduce related works. Section III will present details of our proposed learning algorithm. In Section IV, we will show experimental results of SOLSA on different spatial temporal datasets and compare it with other learning algorithms. Section V gives the conclusions.

II. Background and Related Works

The LIF neuron uses a recurrent structure to maintain its membrane potential. A network formed by connected LIF neurons is a recurrent neural network and can be trained using backpropagation through time (BPTT) [18][19]. However, BPTT is not suitable for online learning [20] on edge devices. Before initiating backpropagation (i.e., learning), the entire input sequence, which can be quite long, must be received and complete the forward propagation process. This not only increases the latency but also causes an unbalanced temporal distribution of workload. Additionally, the delayed learning requires all neurons and synapses maintain a cycle-accurate record of their activities during the processing of input data, as this information is needed for backpropagation at the end. Consequently, a large memory is needed.

To address the limitations of the original BPTT and enable edge applications, Truncated BPTT was proposed [15]. It truncates the original sequence to a bounded history considering only a fixed number of time steps during backpropagation. However, as a variation of the BPTT algorithm, truncated BPTT still requires maintaining a record of history for certain time steps. Therefore, it does not fundamentally change the overall memory and computation complexity. Furthermore, by ignoring long term history, Truncated BPTT introduces bias. Consequently, its performance deteriorates when dealing with very long sequences.

To enable online learning and edge implementation, recent works have employed three-factor Hebbian Learning [16][17]. This type of learning typically incorporates three factors in the weight update process: a term that represents the presynaptic activities, a term that represents the postsynaptic activities, and a term that represents the neuromodulated global error signal. Additionally, the first two factors can be combined to form an eligibility trace [16], which serves as a transient memory or a flag. In a biological system, the eligibility trace is activated when there is a coincidence of spikes between a presynaptic neuron and a postsynaptic neuron. When the eligibility trace is active, signals, which may indicate novelty or surprise, can impact the connection weights between neurons.

Several three-factor Hebbian learning algorithms have been proposed for online learning in SNNs [10][11][12]. One of them is Online Training Through Time (OTTT) [11], which trains SNNs with LIF neurons. OTTT avoided propagating gradients backward through time by disregarding the dependency of the neuron membrane potential on the output activities in the previous time step. In other words, it ignores the membrane potential reset process. While this approximation simplifies the backpropagation process, it also leads to a degradation in learning performance. Another notable work [10], known as E-prop learning, calculates weight change as an accumulation of the product of two variables, the eligibility trace and the learning signal. The former is incrementally computed in a forward manner, while the latter approximates the influence of the spike output of a neuron on output error. Reference [12] proposed a method call DECOLLE which attaches random readout matrices to each layer and defines the global loss as sum of each layer's loss. DECOLLE set all non-local gradients as zero to enforce locality. The weight update rule can then be defined as three parts, errors, pre-synaptic activity, and post-synaptic activity. The weights will be updated at each time step based on three parts in update rule. The algorithms based on three-factor Hebbian learning calculate weight changes incrementally without unrolling the network, making them suitable for online learning and edge implementation. However, existing algorithms of this kind tend to overlook the temporal dynamics on the synapses, which results in a degradation of learning performance as the sequence lengthens.

The proposed SOLSA learning can also be classified as a Three-Factor Hebbian Learning approach. It tackles the limitations of BPTT by incrementally updating the gradient through forward time-axis updates and interleaving forward and backward propagation. In comparison to the BPTT algorithm, SOLSA offers benefits such as reduced memory usage, shorter latency, and a more balanced distribution of workload. As a result, it is better suited for online training in edge applications. Furthermore, SOLSA distinguishes itself from other three-factor Hebbian learning algorithms by considering synapse temporal dynamics and incorporating various enhancement techniques. These include adaptive filter kernel, scheduled weight updates, and early stop mechanisms, all of which significantly contribute to the improved training performance of SOLSA.

III. Proposed Method

A. Neuron and Synapse Models

In this work, we follow [4] to consider a layered network structure with temporal dynamics in both neurons and synapses. A summary of the notations used in this paper is given in TABLE I.

The membrane potential $V_i^l[t]$of the ith neuron at layer l at time t is updated based on the following equation:

$$V_i^l[t] = \lambda V_i^l[t-1] + \sum_j^{N_{l-1}} w_{i,j}^l F_{i,j}^l[t] - V_{th} O_i^l[t-1] , \quad (2)$$

where $O_i^l[t-1]$ denotes the output of this neuron in previous time step. The output is calculated using the Heaviside

activation function given in Equation (1), $O_i^l[t] = U\left(V_i^l[t] - V_{th}\right)$, where V_{th} is potential threshold, $w_{i,j}^l$ is the weight coefficient and $F_j^l[t]$ is the post synaptic potential of the jth input. To model the filter effect of synapses, $F_j^l[t]$ is the output of a first order IIR filter updated as the follows:

$$F_j^l[t] = \alpha_{ij}^l F_j^l[t-1] + \beta_{ij}^l O_j^{l-1}[t-1] \tag{3}$$

Equation (3) shows that the output of synapse filter in current time step is determined by its previous value and the input (i.e., $O_j^{l-1}[t-1]$) in previous time stamps. α_{ij}^l and β_{ij}^l are coefficients of the filter that control the balance between the forward and backward taps of the filter. Fig. 1 depicts a diagram of the neuron model.

Fig. 2 illustrates the data flow graph of the SNN after being unrolled over time. Each column represents a single time step. The duration of the time step is determined by the intervals between consecutive data in the input sequence. During inference, the signal propagates from bottom to top and left to right. During training, when applying BPTT, the gradient propagates reversely from top right corner to the bottom left corner. The vertical propagation is referred to as the *spatial propagation* and the horizontal propagation is referred to as the *temporal propagation*. All parameters in the SNN, including w_{ij}^l, α_{ij}^l, and β_{ij}^l, can be trained using BPTT.

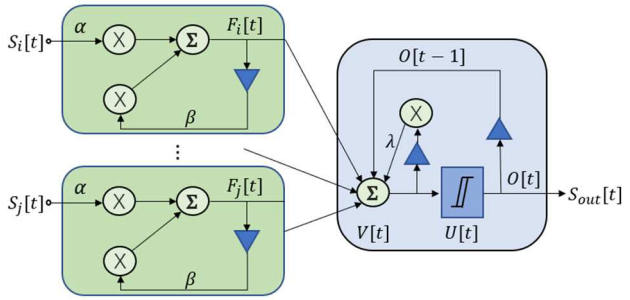

Fig. 1. The architecture of the neuron model.

TABLE I. NOTATIONS USED IN THE PAPER

Symbol	Description
$V_i^l[t]$	Membrane potential of the ith neuron in layer l at time t
$O_i^l[t]$	Output of the ith neuron in layer l at time t
$F_{ij}^l[t]$	Synapse potential of the ijth connection in layer l (i.e., connection from the jth neuron in layer $l-1$ to the ith neuron in layer l)
$\alpha_{ij}^l, \beta_{ij}^l$	Coefficients of synaptic filter of ijth connection in layer l
$w_{i,j}^l$	Synaptic weight of ijth connection in layer l
$U(\cdot)$	Heaviside activation function of spiking neurons
$\epsilon_i^l[t]$	Surrogate gradient of the Heaviside activation function
$\mu_i^l[t]$	Gradient of $V_i^l[t]$ backpropagated through the spatial path
E	Output error (difference of the actual and expected output)
$E[t]$	Output error at time t

B. SOLSA Learning

The Heaviside activation function is non-differentiable [21]. However, under a Gaussian noise $z \sim N(0, \sigma)$, the probability that a neuron with membrane potential V and threshold V_{th} will fire an output spike, which can be calculated using Equation (4),

$$P(V + z > V_{th}) = \frac{1}{2}\text{erfc}\left(\frac{V_{th}-V}{\sqrt{2}\sigma}\right), \tag{4}$$

is differentiable. Following many previous works [4][13], we adopt the derivative of the spiking probability as the surrogate gradient for the Heaviside activation. Let $\epsilon_i^l[t]$ denote the derivative of spike activity of neuron i in layer l, $\epsilon_i^l[t] \approx \frac{dP\left(V_i^l[t]+z>V_{th}\right)}{dV_i^l[t]}$.

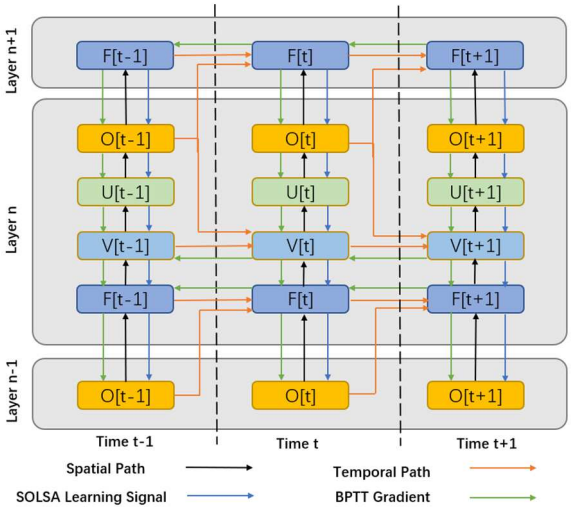

Fig. 2. Unrolled ST SNN.

We denote E as the error, which is the difference between the actual output and the expected output. The term "ijth connection in layer l" refers to the connection from the jth neuron in layer $l-1$ to the ith neuron in layer l. With the given error, the gradient of the weight w_{ij}^l of the ijth connection in layer l can be expressed as Equation (5):

$$\frac{dE}{dw_{ij}^l} = \sum_t \frac{dE}{dV_i^l[t]} \frac{\partial V_i^l[t]}{\partial w_{ij}^l}, \tag{5}$$

where the term $\frac{dE}{dv_i^l[t]}$ calculates the impact of the membrane potential $V_i^l[t]$ on the error E.

The $V_i^l[t]$ affects the error in two ways. Firstly, through the spatial path, it determines the neuron's output $O_i^l[t]$, which subsequently propagates to neurons in layer l+1 and beyond, ultimately reaching the output layer to determine the output error in the current time step. Secondly, through the temporal path, $V_j^l[t]$ determines the neuron's membrane potential in the next time step (i.e., $V_j^l[t+1]$) via the leaky integrate process, which will further impact the future output error. Therefore, we can rewrite Equation (5) as the following expression:

$$\frac{dE}{dV_i^l[t]} = \underbrace{\frac{dE}{dO_i^l[t]}\frac{\partial O_i^l[t]}{\partial V_i^l[t]}}_{\text{Spatial Path}} + \underbrace{\frac{dE}{dV_i^l[t+1]}\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}}_{\text{Temporal Path}}, \tag{6}$$

First, we analyze the spatial path. Using the surrogate gradient function, we know that $\frac{\partial O_i^l[t]}{\partial V_i^l[t]} = \epsilon_i^l[t]$. Using the chain rule, the spatial gradient can be further expanded as follows:

$$\begin{aligned}\frac{dE}{dO_i^l[t]}\frac{\partial O_i^l[t]}{\partial V_i^l[t]} &\approx \sum_k \frac{\partial E[t]}{\partial O_k^{l+1}[t]}\frac{\partial O_k^{l+1}[t]}{\partial V_k^{l+1}[t]}\frac{\partial V_k^{l+1}[t]}{\partial F_{ik}^{l+1}[t]}\frac{\partial F_{ik}^{l+1}[t]}{\partial O_i^l[t]}\frac{\partial O_i^l[t]}{\partial V_i^l[t]} \\ &= \sum_k \frac{\partial E[t]}{\partial O_k^{l+1}[t]}\epsilon_k^{l+1}[t] w_{ki}^{l+1}\beta_{ki}^{l+1}\epsilon_i^l[t]. \end{aligned} \tag{7}$$

As we can see, Equation (7) calculates the vertical backpropagation from the output layer to the lth layer. For the

simplicity, we denote the spatial path gradient in (7) as $\mu_i^l[t]$ and refer to it as the *learning signal*.

Next, we analyze the temporal path in Equation (6). Note that $\frac{dE}{dV_i^l[t+1]}$ is just the one step shifted version of $\frac{dE}{dV_i^l[t]}$, we can rewrite (6) as the following:

$$\frac{dE}{dV_i^l[t]} = \mu_i^l[t] + \left(\mu_i^l[t+1] + \frac{dE}{dV_i^l[t+2]}\frac{\partial V_i^l[t+2]}{\partial V_i^l[t+1]}\right)\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]} \quad (8)$$

$$= \sum_{t\le t'\le T} \mu_i^l[t'] \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]}\frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]}\cdots\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}, \quad (9)$$

where T is the length of the entire input sequence. Plugging (9) into (5) we get (10).

$$\frac{dE}{dw_{ij}^l} = \sum_t \sum_{t\le t'\le T} \mu_i^l[t'] \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]}\frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]}\cdots\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}\frac{\partial V_i^l[t]}{\partial w_{ij}^l}$$
$$= \sum_{t'\le T} \mu_i^l[t'] \cdot \underbrace{\sum_{t\le t'} \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]}\frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]}\cdots\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}\frac{\partial V_i^l[t]}{\partial w_{ij}^l}}_{\varepsilon_{i,j}^l[t']} \quad (10)$$

We denote $\sum_{t\le t'} \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]}\frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]}\cdots\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}\frac{\partial V_i^l[t]}{\partial w_{ij}^l}$ using $\varepsilon_{i,j}^l[t']$ and refer to it as the *Eligibility Trace*. The Eligibility Trace can be updated incrementally using (11):

$$\varepsilon_{i,j}^l[t'] = \varepsilon_{i,j}^l[t'-1]\frac{\partial V_j^l[t']}{\partial V_j^l[t'-1]} + \frac{\partial V_i^l[t']}{\partial w_{ij}^l}$$
$$= (\lambda - v_{th}\epsilon_i^l[t'])\varepsilon_{i,j}^l[t'-1] + F_{i,j}^l[t'] \quad (11)$$

Using the learning signal ($\mu_j^{l+1}[t']$) and the eligibility trace $\varepsilon_{i,j}^l[t']$, we now have the SOLSA learning rule.

$$\frac{dE}{dw_{ij}^l} = \sum_{t'} \mu_j^{l+1}[t'] \cdot \varepsilon_{i,j}^l[t'] \quad (12)$$

Both the learning signal and eligibility trace rely only on signals of current and previous time steps. The entire algorithm can be implemented using a streaming process. It always moves forward in time.

C. Adaptive Synapse Filter Kernel

Previous works on BPTT training have also shown that optimizing the filter kernels, achieved by adapting α_{ij}^l and β_{ij}^l, leads to improved learning results [4]. We introduce an online technique that approximates the gradient with respect to the filter kernels.

If we only consider time step t, the gradient of error $E[t]$ with respect to α_{ij}^l and β_{ij}^l can be calculated using the spatial backpropagation path as follows:

$$\nabla_{\alpha_{ij}^l}[t] = \frac{\partial E[t]}{\partial \alpha_{ij}^l} = \mu_i^l[t]\frac{\partial V_i^l[t]}{\partial F_{ij}^l[t]}\frac{\partial F_i^l[t]}{\partial \alpha_{ij}^l} = \mu_i^l[t] \cdot w_{ij}^l \cdot F_i^l[t-1],$$

$$\nabla_{\beta_{ij}^l}[t] = \frac{\partial E[t]}{\partial \beta_{ij}^l} = \mu_i^l[t]\frac{\partial V_i^l[t]}{\partial F_{ij}^l[t]}\frac{\partial F_i^l[t]}{\partial \beta_{ij}^l} = \mu_i^l[t] \cdot w_{ij}^l \cdot O_j^{l-1}[t-1].$$

However, the error $E[t]$ is not solely influenced by neuron activities in time step t, but also by activities during past time steps. We assume that the impact on the future error decays exponentially with time. To capture this historical impact, we calculate the overall gradient of $E[t]$ as follows:

$$\frac{dE[t]}{d\beta_{ij}^l} \approx \nabla_{\beta_{ij}^l}[t] \cdot \sum_{n=0}^t \gamma^n,$$

$$\frac{dE[t]}{d\beta_{ij}^l} \approx \nabla_{\beta_{ij}^l}[t] \cdot \sum_{n=0}^t \gamma^n,$$

where $0 < \gamma < 1$ is a decay factor. Finally, we consider all $E[t]$ from time 0 to T, and calculate the gradient of α_{ij}^l and β_{ij}^l using (13) and (14):

$$\frac{dE}{d\alpha_{ij}^l} = \sum_t \nabla_{\alpha_{ij}^l}[t] \cdot \frac{1-\gamma^{t+1}}{1-\gamma}, \quad (13)$$

$$\frac{dE}{d\beta_{ij}^l} = \sum_t \nabla_{\beta_{ij}^l}[t] \cdot \frac{1-\gamma^{t+1}}{1-\gamma}. \quad (14)$$

D. Scheduled weight update

Unlike BPTT, where gradients and weight updates are calculated at the end of the input sequence, SOLSA learning accumulates Δw overtime. This enables us to update the network to reflect the partial results of learning even before the entire input sequence has been received. This facilitates faster convergence by allowing the SNN to escape inferior settings earlier. However, updating weights frequently at each time step can introduce noise due to local variations in the input sequence. Therefore, careful selection of when to update is crucial. We propose an approach called *scheduled weight update*, which selects specific time steps as update points and only update weights at these selected points. The number of update points serves as a hyper-parameter and is configured before training. Generally, a ratio of 1 to 50 between the update points and the sequence length yields desirable results.

Initially, the update schedule consists of only one update point, which is the end of the input sequence. At each time step t, we calculate and record g_t, the total absolute partial gradient of all neuron connections:

$$g_t = \sum_{ijl} \left|\frac{dE[t]}{dw_{ij}^l}\right| = \sum_{ijl} \left|\sum_{t'=0}^t \mu_j^l[t'] \cdot \varepsilon_{i,j}^l[t']\right|.$$

The time step with the highest g_t are selected as an update point and added to the update schedule. We will repeat this process until the schedule has the required number of update points.

E. Eearly-Stop Training

For certain input sequences, the distinctive pattern that signifies the class information emerges right at the beginning. This enables the SNN model to make accurate predictions before reaching the end of the sequence. However, the data that follows the signature pattern may consist of random noise that does not contribute to the classification. Training the model using the data beyond the signature pattern will deteriorate the learned features. To address this issue, we propose an early-stop technique that focuses on the segment of the sequences containing useful information.

For each training sample, the early-stop detector monitors the prediction accuracy, denoted as $A(t)$, at every scheduled update point t. If the accuracy surpasses a predefined threshold indicating a correct prediction, a counter C is incremented. We stop processing this training sample when C reaches $N/2$, where N is the total number of update points.

Fig. 3 shows an illustrative example of the early stop training. The model is trained binary classification of input sequences. It has two output neurons, which generate spike trains $O_0[t]$ and $O_1[t]$. The output neuron with the higher spiking activities indicates the prediction. The accuracy function is defined as $A(t) = \frac{\sum_{i=0}^t O_y[i]}{\sum_k \sum_{i=0}^t O_k[i]}$, and the threshold is

set to be 0.5. In other words, the prediction is considered correct if the neuron corresponding to the target class (y) generates more output spikes than the other neuron. In the figure, $N = 4$ and the C value reached $N/2$ at the 3rd update point, which indicates an early stop of the training process for this input sample.

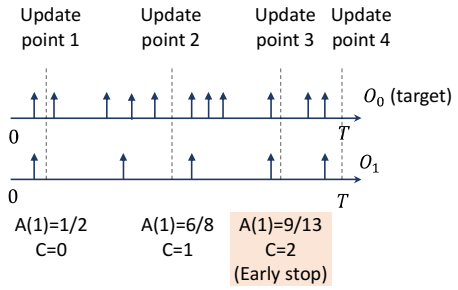

Fig. 3. An example of early-stop training.

IV. Experimental results

We tested SOLSA learning on the classification of multivariate time series. The datasets are sequences of sensor readings from motion sensor, EEG sensor, and visual event sensor, etc. Except the DVS128 [23], all other datasets can be downloaded from The UCR Time Series Classification Archive and UCI Machine Learning Repository. Most of them consist of current readings of the sensors except DVS128, which consists of recoded visual events. To convert the current readings into spikes, current-based LIF neurons [22] are employed in the first layer for temporal population coding. TABLE II. summarizes the key statistics of those datasets including their input and output dimensions and sequence length. The architectures of the SNN models used as the classifiers are also given. All layers are fully connected layers. All layers, including the input layer, are trainable.

TABLE II. Dataset information

	Dataset Name	Input dimension	Sequence length	Output size	SNN architecture
Regular	EMG gesture	8	100	7	8-150-150-7
	Finger mov.	28	50	2	28-100-100-2
	Basic motion	6	100	4	6-100-100-4
	Epilepsy	3	207	4	3-100-100-4
	Jap. Vowel	12	29	9	12-100-100-9
	RacketSports	6	30	4	6-100-100-4
	DVS128	4096	100	11	4096-100-100-11
Long	Self reg. scp	6	896	2	6-100-100-2
	EMG action	8	1000	10	8-200-200-10

Four reference algorithms, namely BPTT [9], Truncated BPTT (T-BPTT) [15], E-prop [10], and OTTT [11], were implemented for comparison. The T-BPTT considers up to 20 steps of historical information during training and disregards past information beyond that point. Both BPTT and T-BPTT require additional storage for the unrolled network, hence they are unsuitable for edge implementation. E-prop learning and OTTT are non-BPTT based, however, they do not have synapse filters or enhancement techniques such as scheduled weight update and early stop. Furthermore, OTTT omits the impact of membrane potential reset in the gradient approximation. For ablation test, three variants of SOLSA were created by disabling adaptive filter kernel or early stop, or a combination of both. TABLE III. summarized the non-BPTT algorithms tested in the experiment.

A. Performance Comparison

TABLE IV. compares different learning algorithms and SOLSA for their accuracy and memory usage. All training uses a batch size of 1 to resemble online learning. The results show that SOLSA achieves the highest accuracy among all algorithms for all datasets. In addition, it requires much lower memory usage compared to BPTT especially when data sequence is long. Truncated BPTT effectively reduces the overhead of temporal backpropagation and lowers the memory cost compared with original BPTT, however, as shown in TABLE IV. , its performance is unstable.

TABLE III. A Summery of Tested Non-BPTT algorithms

Feature	SOLSA	E-prop	OTTT	SOLSA variant 1	SOLSA variant 2	SOLSA variant 3
Synapse filter	✓			✓	✓	✓
Adaptive weight	✓	✓	✓	✓	✓	✓
Adaptive kernel	✓				✓	
Impact of reset	✓	✓		✓	✓	✓
Scheduled updates	✓			✓	✓	✓
Early stop	✓					✓

TABLE IV. Comparison of different learning algorithms

Dataset	BPTT based		Three factor Hebbian			Memory usage (MB)	
	BPTT	TBPTT	SOLSA	E-prop	OTTT	BPTT	SOLSA
EMG gesture	0.956	0.664	**0.985**	0.675	0.672	13.4	6.6
Finger mov.	0.58	0.56	**0.64**	0.58	0.59	9.0	7.1
Basic motion	1	0.25	**1**	0.925	1	13.4	7.5
Epilepsy	0.941	0.676	**0.971**	0.816	0.904	22.7	9.9
Jap. Vowel	0.926	0.951	**0.981**	0.944	0.969	7.1	6.6
RacketSports	0.809	0.769	**0.907**	0.388	0.796	7.4	6.2
DVS128	0.959	0.6	**0.979**	0.819	0.875[a]	105.8	67.2
Self reg. scp	0.836	0.866	**0.897**	0.876	0.89	79.6	22.2
EMG action	0.973	0.696	**0.979**	0.77	0.161	158.4	44.1

[a] Online training using batch 1.

From TABLE IV. , it can be observed that the SOLSA learning algorithm outperforms both E-prop and OTTT. On average, it improves the classification accuracy of the 9 datasets by 30% and 66.5% compared to E-prop and OTTT respectively. It is worth noting that although the OTTT is a simpler algorithm than E-prop, it updates weights at every time step, while E-prop only updates at the end of the sequence. As a result, OTTT outperforms E-prop for certain datasets.

B. Ablation Study of SOLSA

In this section, we demonstrate the impact of the enhancement techniques by comparing SOLSA with the 3 variants listed in TABLE III. We also implemented and compared it with an unscheduled SOLSA where the model is always updated only once at the end of the input sequence. For SOLSA without adaptive filter kernel (variant 3), fixed kernel coefficient (i.e., $\alpha_{ij}^l, \beta_{ij}^l$) are used for all synapses. We performed a hyper-parameter search and found the best combination of those two parameters that works for most of the datasets. We need to point out that such hyperparameter tuning is usually not practical for online learnings.

TABLE V. presents the comparison results. We can see that the original SOLSA outperforms the variants for almost all cases. Comparing SOLSA with variant 3 shows that using adaptive filter kernel helps to improve the accuracy by more

than 5%. Comparing SOLSA with variant 2 shows that early stop in training leads to an average improvement of 17% in classification accuracy. Finally, by comparing SOLSA with its unscheduled version, we can see that using scheduled learning provides almost 20% accuracy improvements.

TABLE V. ABLATION STUDY: EARLY STOP AND ADAPTIVE KERNEL

Dataset	Accuracy				
	Unscheduled	variant 1	variant 2	variant 3	SOLSA
EMG gesture	0.912	0.942	0.671	0.957	**0.985**
Finger mov.	0.56	**0.65**	0.59	0.58	0.64
Basic motion	0.95	1	1	1	**1**
Epilepsy	0.794	0.934	0.713	0.958	**0.971**
Jap. Vowel	0.869	0.975	0.619	0.96	**0.981**
RacketSports	0.598	0.855	0.901	0.835	**0.907**
DVS128	0.895	0.93	0.959	0.93	**0.979**
Self reg. scp	0.88	0.894	0.897	0.893	**0.897**
EMG action	0.134	0.93	0.946	0.848	**0.979**

We also observed that, without adapting the filter kernels, the learning performance is very sensitive to the values of filter coefficients α and β. To show the importance of filter adaptation, we tested different random combinations of fixed α and β values on each dataset and recorded the best, worst and medium accuracy of trained models as shown in Fig. 4. As mentioned earlier, variant 3 was obtained after extensive hyperparameter tunning to make sure that the same set of filter coefficients works relatively good for all datasets. Its accuracy is also shown in the plot. Finally, we list the accuracy of SOSA in the same plot as a reference. The results clearly show the importance of adapting the filter kernel coefficients instead of using the fixed value.

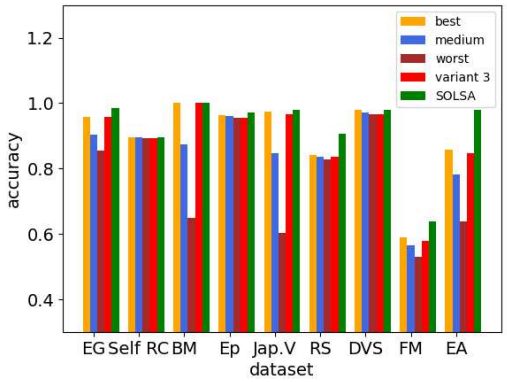

Fig. 4. Accuracies of different α and β settings.

V. CONCLUSION

In this paper, we present SOLSA, an online learning algorithm for SNNs with both synapse and neuron dynamics. Enhancement techniques such as adaptive filter kernel, scheduled weight update and early stop are also presented. Compared with BPTT training, SOLSA requires much less memory storage and has more balanced temporal workload distribution, hence more suitable for edge implementation. Compared with other online learning algorithms, SOLSA leads to more accurate models with more robust performance. The algorithm is evaluated using data series collected from sensor readings, and the results indicate SOLSA provides outstanding performance in learning temporal sequences compared to existing algorithms.

REFERENCES

[1] Tavanaei, A. et al., 2019. Deep learning in spiking neural networks. Neural Networks, Volume 111, pp. 47-63.

[2] Ghosh-Dastidar, Samanwoy, and Hojjat Adeli. "Spiking neural networks." International journal of neural systems 19.04 (2009): 295-308.

[3] La Camera, G. et al., 2004. Minimal models of adapted neuronal response to in vivo-like input currents. Neural Computation, 16(10), p. 2101–2124.

[4] Fang, H., Shrestha, A., Zhao, Z. & Qiu, Q., 2020. Exploiting neuron and synapse filter dynamics in spatial temporal learning of deep spiking neural network. Yokohama, International Joint Conferences on Artificial Intelligence, pp. 2799-2806.

[5] Koch, C. & Segev, I., 2000. The role of single neurons in information processing. Nature Neuroscience, 3(Suppl 11), p. 1171–1177.

[6] Fang, H. et al., 2020. Encoding, Model, and Architecture: Systematic Optimization for Spiking Neural Network in FPGAs. San Diego, IEEE/ACM International Conference On Computer Aided Design (ICCAD)

[7] Markram, H., Lübke, J., Frotscher, M. & Sakmann, B., 1997. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science, 275(5297), pp. 213-215.

[8] Hebb, D., 1949. The organization of behavior; a neuropsychological theory. s.l.: Wiley

[9] Werbos, P. J., 1988. Generalization of backpropagation with application to a recurrent gas market model. Neural Networks, 1(4), pp. 339-356.

[10] Bellec, G. et al., 2020. A solution to the learning dilemma for recurrent networks of spiking neurons. Nature Communications, 11(3625).

[11] Xiao, M. et al., 2022. Online Training Through Time for Spiking Neural Networks. s.l., Conference on Neural Information Processing System.

[12] Kaiser, Jacques, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE).Frontiers in Neuroscience 14 (2020): 424.

[13] Zenke, F. & Vogels, T. P., 2021. The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks. Neural Computation, 33(4), p. 899–925.

[14] Robinson, A. J., and Frank Fallside. The utility driven dynamic error propagation network. Vol. 1. Cambridge: University of Cambridge Department of Engineering, 1987.

[15] Williams, R. J. & Peng, J., 1990. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. Neural Computation, 2(4), p. 490–501.

[16] Gerstner, W. et al., 2018. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. Frontiers in Neural Circuits.

[17] Frémaux, N. & Gerstner, W., 2016. Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules. Frontiers in Neural Circuits.

[18] Yin, Shihui, et al. "Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations." 2017 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2017.

[19] Shen, Guobin, Dongcheng Zhao, and Yi Zeng. "Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks." Patterns 3.6 (2022): 100522.

[20] Lobo, Jesus L., et al. "Spiking neural networks and online learning: An overview and perspectives." Neural Networks 121 (2020): 88-100.

[21] Wang, Jun. "Analysis and design of a k-winners-take-all model with a single state variable and the heaviside step activation function." IEEE Transactions on Neural Networks 21.9 (2010): 1496-1506.

[22] Cavallari, Stefano, Stefano Panzeri, and Alberto Mazzoni. "Comparison of the dynamics of neural interactions between current-based and conductance-based integrate-and-fire recurrent networks." Frontiers in neural circuits 8 (2014): 12.

[23] Amir, Arnon, et al. "A low power, fully event-based gesture recognition system." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.