ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240804

Reasonable Gradients for Online Training Algorithms in Spiking Neural Networks

Lang Xue[a], HanWen Liu[a], Jing Wang[a] and Hong Qu[a,*]

[a]School of Computer Science and Engineering, University of Electronic Science and Technology of China

Abstract. Spiking neural networks (SNNs) have the potential to simulate sparse and spatio-temporal dynamics observed in biological neurons, making them promising for achieving energy-efficient artificial general intelligence. While backpropagation through time (BPTT) ensures reliable precision for training SNNs, it is hampered by high computation and storage complexity and does not conform to the instantaneous learning mechanism in brains. On the contrary, online training algorithms, which are biologically interpretable, offer low latency and memory efficiency, and are well-suited for on-chip learning applications. However, recent research exhibit a deficiency in the scientific comprehension of online gradients, which leads to certain limitations. To address this issue, we conduct an in-depth analysis of the calculation deviation in chain derivations induced by weight update and find two pivotal factors that affect the accuracy of online gradients: completeness and timeliness. To further enhance the performance of online training leveraging these findings, we propose spatio-temporal online learning (STOL), which substantially ameliorates the accuracy of the online gradients and demonstrates superior computation and memory efficiency. Our experiments on CIFAR-10, CIFAR-100, ImageNet, CIFAR10-DVS, and DVS128-Gesture datasets demonstrate that our method achieves state-of-the-art performance across most of these tasks. Besides, it shows a great improvement compared with existing online training algorithms.

1 Introduction

Spiking neural networks (SNNs), hailed as third-generation neural networks [25], are designed with the objective to reduce power usage and to execute highly brain-like neuromorphic computation [24, 30]. Recent works on surrogate gradient (SG) which assigns approximate derivatives to the non-differentiable spiking step function [23, 27], have significantly improved the performance of training deep SNNs across a variety of tasks using BPTT [10]. However, BPTT performs learning only once within the whole time interval T [34, 35], which is inconsistent with the inherent online learning mechanism in biological brains [39]. This learning strategy also incurs substantial memory occupation and training latency, rendering it unsuitable for neuromorphic hardware applications [13, 36]. Furthermore, due to the imprecision of SG in SNNs, increasing the depth in spatial and temporal dimensions is more likely to exacerbate issues of vanishing and exploding gradients [39], further limiting the algorithm's performance.

Online learning presents a more advanced learning paradigm that facilitates continuous real-time learning from temporally continuous data streams [19], aligning with our conventional comprehension of biological learning mechanisms. Simultaneously, it can address those problems associated with BPTT mentioned above. However, related work on online training algorithms in SNNs either engender intricate computation due to the approximation of the BPTT gradient [32], or overlook crucial gradient information, leading to a decline in training performance [36]. Consequently, they fail to fully exploit the benefits of online learning. As identified through our in-depth analysis, we attribute this to a lack of comprehensive understanding of online gradients, which should involve the following two crucial factors: (For clarity of our description, we refer to the backpropagation of error in time as history or historical gradient, the number of propagated time steps as the length of history, and categorize it into two components: temporal historical gradient (THG) and spatial historical gradient (SHG), which are visually represented by red and blue arrows in Figure 1, respectively.)

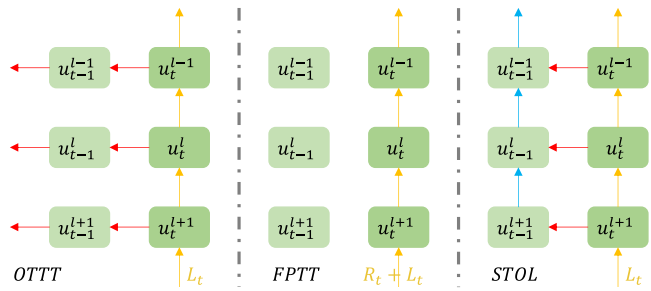

Figure 1: Comparison of the backpropagation paths of gradient for three online learning algorithms. Red, blue and yellow arrows represent THG, SHG and the current gradient, respectively.

- **Completeness:** Incorporating both THG and SHG into online training is crucial to ensure the most accurate gradient descent direction. SNNs possess spatio-temporal information processing capabilities, which require corresponding learning rules to establish spatial and temporal relationships within the network. Simplified processing, as seen in studies such as [39], may not offer such assurance.
- **Timeliness:** Incorporating long-term history into online gradient calculation is unreasonable as it will interfere with the online training process. This is mainly because it amplifies computational imprecision significantly with instant weight updating. Additionally, it contradicts the inherent temporal locality of the learning rules observed in biological neural systems [6].

Drawing upon these two principles, we propose spatio-temporal online learning (STOL), an online training algorithm that executes a

* Corresponding Author. Email: hongqu@uestc.edu.cn

complete backpropagation within the scope of the last two time steps at each moment. To provide a more intuitive comparison between our approach and some existing classical online training algorithms, we illustrate their respective error backpropagation paths in Figure 1. In accordance with our theory, OTTT [36] contravenes both requirements of completeness and timeliness, while FPTT [39] infringes upon the requirement of completeness, which impacts their training effectiveness. On the contrary, STOL satisfies both completeness to ensure an accurate gradient and timeliness to avoid the interference of outdated history, therefore improving the performance of online training. Besides, it also exhibits superior computation and memory resource utilization and enhanced structural versatility.

In conclusion, our contributions to this work can be summarized as follows:

1. We conduct a rigorous analysis of the gradient in online training algorithms, identify the pivotal issues therein, and substantiate our findings through both theoretical derivation and experimental validation.
2. We propose a novel online training algorithm for spiking neural networks, named spatio-temporal online learning (STOL), which enables online training with enhanced precision, liberates the severe memory burden, and improves the computational efficiency of the devices.
3. We conduct experiments on CIFAR-10, CIFAR-100, ImageNet, CIFAR10-DVS and DVS128-Gesture datasets, and the results show that our method outperforms the existing online learning approaches and achieves the state-of-the-art performance on most of these tasks.

2 Related Work

Although gradient descent and error backpropagation have enabled deep neural networks to thrive [15, 22], training SNNs directly in this way is difficult due to the discontinuous nature of spikes. Numerous research efforts have successfully approximated the derivatives of spikes and trained SNNs as recurrent neural networks (RNNs) [38]. To further improve training performance, tdBN [40] normalizes the input in both time and batch dimensions; SEW-ResNet [10] and MS-ResNet [16] investigate suitable deep residual learning strategies for it, effectively alleviating the problems of vanishing and exploding gradients; Diet-SNN [29] employs direct encoding as opposed to rate based encoding, thus minimizing information loss; TET [9] refines the approach to assess the objective function, leading to a smoother decrease in error.

Based on these research, the development of online training algorithms has been further advanced. Deep continuous local learning (DECOLLE) employs local error functions and truncated backpropagation through time (TBPTT) [31, 33] for online learning, which has no memory overhead during gradient computation [19]. While, the constraint of spatial locality hampers its efficacy in complex tasks. Local tandem learning (LTL) follows the teacher-student learning approach by mimicking the intermediate feature representations of a pre-trained ANN [37] and benefits from spatio-temporal locality. However, it also encounters challenges with accuracy degradation when applied to complex tasks. Real-time recurring learning (RTRL) [32] facilitates online training and eliminates the need to cache the previous activations. Nevertheless, this approach incurs significant computational overhead, rendering it unsuitable for application to large-scale networks. E-prop [2] integrates synaptic plasticity and monitors the trace from local activity to facilitate online training of the SNNs without requiring temporal backpropagation of the error signal. Althrough e-prop's demonstrated efficacy across numerous benchmark problems, a discernible gap persists between e-prop and BPTT [39]. Online spatio-temporal learning (OSTL) [3] decouples the computation of gradients in spatial and temporal to facilitate the derivation of weight updates online. This approach allows networks to process and learn new input data simultaneously, thereby addressing the issues of weight symmetry and update lock [28]. OSTL has shown results comparable to BPTT on tasks like language modeling and speech recognition, however, it is notably computation and memory intensive for general RNNs [28].

Another alternative idea for online training involves the selection of effective components of the gradient. Online training through time (OTTT) [36] backpropagates error in time independently in each layer, enabling iterative updates of the historical states. However, this approach limits the application of batch normalization (BN) [17], leading to the adoption of the normalization-free [4, 5] method as an alternative to BN, resulting in a compromise on flexibility. Besides, the structure-independent temporal backpropagation also limits the learning ability of this algorithm. Forward propagation through time (FPTT) [18] substitute novel dynamic regularization part for error backpropagation in time, [39] integrates FPTT and liquid spiking neurons (LSN), exhibits a performance that surpasses BPTT on some specific sequence tasks. However, the performance of this approach in broader application scenarios remains to be further validated, and the incorporation of LSN also introduces certain complexities. For online training algorithms based on this concept, the crux lies in accurately identifying the significant components of the history. The aforementioned two algorithms do not provide scientific solutions to this issue. Instead, our method gives a more reasonable explanation for the historical gradient. Furthermore, our approach can be directly applied to general SNNs which is not limited by the network and neuron structure.

3 Method

In this section, we establish the formulas for forward computation and backpropagation used in this research. Based on this, we will prove our theory about timeliness and then introduce the STOL algorithm. As the concept of completeness is fairly straightforward, we only give its validation within the experimental phase.

3.1 Forward and Backward Propagate Rules

Spiking neurons, serving as a fundamental element in spiking neural networks, have diverse state equations for various neural dynamics. In this research, we adopt the leaky integrate and fire (LIF) model, the differential equation it employs to delineate membrane potential u is presented below:

$$\tau_m \frac{du}{dt} = -(u - V_{reset}) + R \cdot I(t), \tag{1}$$

where V_{reset} is the resting potential, $I(t)$ is the input current, τ_m and R are hyperparameters that control the behavior of the neuron. To accommodate the utilization of prevailing deep learning frameworks, Equation (1) is typically articulated in a discrete variant, culminating in the subsequent form:

$$u_t^l = u_{t-1}^l \cdot (1 - s_{t-1}^l)/\tau_m + f(w^l, s_t^{l-1}), \tag{2}$$

$$s_t^l = H(u_t^l - V_{th}), \tag{3}$$

where the superscript l denotes the number of layers, the subscript t denotes the time step; s, w and V_{th} denote the output spikes, the network weights and the firing threshold, respectively; f represents the synaptic operation (e.g. convolution and fully connected layers, etc), and $H(x)$ is the Heaviside step function.

In backpropagation, to enable the computation of the parameters' gradient in each layer using the chain rule, it is necessary to employ surrogate function for spike generation, which usually takes the following form:

$$H^{'}(x) = \begin{cases} 0, & |x| > \frac{1}{\alpha}, \\ -\alpha^2 x + \alpha, & |x| \leq \frac{1}{\alpha}, \end{cases} \tag{4}$$

where α serves as a control parameter dictating the smoothness of the function and is usually set to 1.0.

As delineated by Equation (2), the membrane potential u imparts both spatial and temporal effects due to the involvement in potential iteration and the propagation to the subsequent layer. Therefore, error backpropagation can be manifested as the backpropagation along the spatio-temporal axis. In this way, the gradient of u_t^l can be calculated like:

$$\frac{\partial \mathcal{L}}{\partial u_t^l} = \frac{\partial \mathcal{L}}{\partial u_{t+1}^l}\frac{\partial u_{t+1}^l}{\partial u_t^l} + \frac{\partial \mathcal{L}}{\partial u_t^{l+1}}\frac{\partial u_t^{l+1}}{\partial s_t^l}\frac{\partial s_t^l}{\partial u_t^l}, \tag{5}$$

where $\mathcal{L}$ donates the loss.

3.2 *Deviation in Gradient Calculation*

Our core idea *timeliness* emphasizes that the computation of long term historical gradients will interfere with online training, which is contrary to the conventional understanding to some extent. So, in this subsection, we will give the proof of this claim.

To streamline our notation, we utilize δ_t^l to represent the actual value of $\frac{\partial \mathcal{L}_t}{\partial u_t^l}$ while $\hat{\delta}_t^l$ signifies its calculated counterpart. The deviation between these two quantities is denoted by ϵ_t^l, which can be zero or any other value. Then the problem of timeliness comes from:

Theorem 1. *The instantaneous update in weight w leads to a deviation between the calculated gradient and its actual value, and the deviation between time j and i at layer l is:*

$$\epsilon_j^l = \sum_{\tau_l=j}^{i} \big(c_{\tau_l}^l + \big(\dots \sum_{\tau_L=\tau_{L-1}}^{i} \frac{\partial u_{\tau_L}^L}{\partial u_{\tau_{L-1}}^L} c_{\tau_L}^L\big) w_i^{l+1}\theta_{\tau_l}^l\big)\frac{\partial u_{\tau_l}^l}{\partial u_j^l}, \tag{6}$$

where $c_j^l = \hat{\delta}_j^{l+1}\sum_{\tau=j}^{i-1}\Delta w_\tau^{l+1}\theta_j^l$, and $\theta_j^l = H^{'}(u_t^l - V_{th})$, L is the depth of the neural network. This value will increase exponentially as the time interval between j and i increases, eventually causing the wrong gradient $\hat{\delta_j^l}$.

Proof. Suppose the current time step is i, and our objective is to calculate the gradient at time j. As per Equation (5), we can derive:

$$\delta_j^l = \delta_j^{l+1} w_j^{l+1}\theta_j^l + \delta_{j+1}^l \frac{\partial u_{j+1}^l}{\partial u_j^l}, \tag{7}$$

which is determined by weight at time j. But in calculation we only have weight w_i^l at time i, which equals $w_j^l + \sum_{\tau=j}^{i-1}\Delta w_\tau^l$, so the calculated gradient is:

$$\hat{\delta}_j^l = \hat{\delta}_j^{l+1}\big(w_j^{l+1} + \sum_{\tau=j}^{i-1}\Delta w_\tau^{l+1}\big)\theta_j^l + \hat{\delta}_{j+1}^l\frac{\partial u_{j+1}^l}{\partial u_j^l}. \tag{8}$$

The difference between Equations (7) and (8) gives the gradient bias as follows:

$$\epsilon_j^l = \big(c_j^l + \epsilon_j^{l+1} w_i^{l+1}\theta_j^l\big)\frac{\partial u_j^l}{\partial u_j^l} + \epsilon_{j+1}^l\frac{\partial u_{j+1}^l}{\partial u_j^l}, \tag{9}$$

which demonstrates exponential increase due to the combination of ϵ_j^{l+1} and ϵ_{j+1}^l. Firstly, do the recursion in temporal dimension by expanding ϵ_{j+1}^l using Equation (9), which yields $\epsilon_j^l = \big(c_j^l + \epsilon_j^{l+1}w_i^{l+1}\theta_j^l\big)\frac{\partial u_j^l}{\partial u_j^l} + \big(c_{j+1}^l + \epsilon_{j+1}^{l+1}w_i^{l+1}\theta_{j+1}^l\big)\frac{\partial u_{j+1}^l}{\partial u_j^l} + \epsilon_{j+2}^l\frac{\partial u_{j+2}^l}{\partial u_j^l}$. As the current time is i, $\epsilon_i^l = \big(c_i^l + \epsilon_i^{l+1}w_i^{l+1}\theta_i^l\big)\frac{\partial u_i^l}{\partial u_i^l}$, so finally we can derive:

$$\epsilon_j^l = \sum_{\tau=j}^{i}\big(c_\tau^l + \epsilon_\tau^{l+1}w_i^{l+1}\theta_\tau^l\big)\frac{\partial u_\tau^l}{\partial u_j^l}. \tag{10}$$

Based on Equation (10), spatial recursion can be completed. Substituting ϵ_τ^{l+1} yields $\epsilon_j^l = \sum_{\tau_l=j}^{i}\big(c_{\tau_l}^l + \big(\sum_{\tau_{l+1}=\tau_l}^{i}\big(c_{\tau_{l+1}}^{l+1} + \epsilon_{\tau_{l+1}}^{l+1}w_i^{l+2}\theta_{\tau_{l+1}}^{l+1}\big)\frac{\partial u_{\tau_{l+1}}^{l+1}}{\partial u_j^{l+1}}\big)w_i^{l+1}\theta_{\tau_l}^l\big)\frac{\partial u_{\tau_l}^l}{\partial u_j^l}$. Unrolling to layer L in this way gives the result shown in Equation (6). □

For ease of description, we consider f as linear transformation, therefore $\frac{\partial u_t^{l+1}}{\partial s_t^l} = f_w^{'}(w^{l+1}, s_t^l) = w^{l+1}$. The above proof shows that the change in weights, Δw^{l+1}, gives rise to a fundamental bias c_j^l. This bias, inherent to each state, progressively accumulates across all states from time j to i, spanning from layer l to L, finally leading to a discrepancy ϵ_j^l. Notably, when the interval between j and i is substantial, c_j^l becomes large due to the significant Δw_{l+1}, causing the failure of gradient $\hat{\delta}_j^l$ and interfering with the training.

3.3 *Spatio-Temporal Online Learning*

In the preceding subsection, we highlighted substantial inaccuracies in the computation of long-term historical gradients. Given these findings, it is reasonable to mitigate these inaccuracies and simplify the computational process by disregarding long-term historical gradients. Motivated by this insight, we propose STOL, a method that facilitates high-precision online training and further optimizes the performance of computing devices with enhanced efficiency.

The central idea of our approach is to propagate the instantaneous loss, which is computed from the predicted output at each time step and the ground truth, back to the neuron states of the two most recent time steps. Then the gradients of these two time steps are accumulated to update the network parameters. The specific calculation rules rely on the chain derivative and the error backpropagation described in Subsection 3.1. We employ the same prediction strategy as in TET, wherein the output layer is a linear transformation without neuron activation, and the loss function is the combination of cross-entropy and mean square error, which is defined as:

$$\mathcal{L}_t = (1-\lambda)CE(\hat{y}_t, y) + \lambda MSE(\hat{y}_t, y). \tag{11}$$

where $\hat{y}_t$ is the prediction at time t, y is the ground truth, and the hyper-parameter λ is set to 0.05.

Algorithm 1 gives the complete procedure of our approach, which defines the details of the forward and backward computation, as well as the data interaction in the overall flow. The derivative calculation in the backward process depends on three key formulas for chain rule, which are $\frac{\partial u_t^l}{\partial s_t^{l-1}} = f_s^{'}(w^l, s_t^{l-1})$, $\frac{\partial u_t^l}{\partial w^l} = f_w^{'}(w^l, s_t^{l-1})$, and

Algorithm 1 Spatio-temporal online learning

Require: Input data x_t at time t, label y of the input.

Forward:
for $l = 1$ to N **do**
  if $l == N$ **then**
    Calculate predicted output $y_t = f(w^l, s_t^{l-1})$.
  else
    Calculate u_t^l and s_t^l using Equations (2), (3).
    Save the states u_t^l, s_t^l, s_{t-1}^l, u_{t-1}^l.
  end if
end for
Backward:
Calculate loss $\mathcal{L}_t$ depend on y_t, y and Equation (11).
for $l = N$ downto 1 **do**
  if $l == N$ **then** ▷ Output layer, no activation.
    $\frac{\partial \mathcal{L}_t}{\partial w^l} \leftarrow \frac{\partial \mathcal{L}_t}{\partial y_t} f_w^{'}(w^l, s_t^{l-1})$, $\frac{\partial \mathcal{L}_t}{\partial s_t^{l-1}} \leftarrow \frac{\partial \mathcal{L}_t}{\partial y_t} f_s^{'}(w^l, s_t^{l-1})$
    $\frac{\partial \mathcal{L}_t}{\partial s_{t-1}^{l-1}} \leftarrow 0$ ▷ Initialize to zero for iterating.
  else
    $\frac{\partial \mathcal{L}_t}{\partial u_t^l} \leftarrow \frac{\partial \mathcal{L}_t}{\partial s_t^l}\frac{\partial s_t^l}{\partial u_t^l}$, $\frac{\partial \mathcal{L}_t}{\partial u_{t-1}^l} \leftarrow \frac{\partial \mathcal{L}_t}{\partial s_{t-1}^l}\frac{\partial s_{t-1}^l}{\partial u_{t-1}^l} + \frac{\partial \mathcal{L}_t}{\partial u_t^l}\frac{\partial u_t^l}{\partial u_{t-1}^l}$
    $\frac{\partial \mathcal{L}_t}{\partial w^l} \leftarrow \frac{\partial \mathcal{L}_t}{\partial u_t^l}\frac{\partial u_t^l}{\partial w^l} + \frac{\partial \mathcal{L}_t}{\partial u_{t-1}^l}\frac{\partial u_{t-1}^l}{\partial w^l}$
    $\frac{\partial \mathcal{L}_t}{\partial s_t^{l-1}} \leftarrow \frac{\partial \mathcal{L}_t}{\partial u_t^l}\frac{\partial u_t^l}{\partial s_t^{l-1}}$, $\frac{\partial \mathcal{L}_t}{\partial s_{t-1}^{l-1}} \leftarrow \frac{\partial \mathcal{L}_t}{\partial u_{t-1}^l}\frac{\partial u_{t-1}^l}{\partial s_{t-1}^{l-1}}$
  end if
  Update parameters w^l using gradient descent algorithm.
end for

$\frac{\partial u_{t+1}^l}{\partial u_t^l} = \big(1 - s_t^l - u_t^l \cdot H^{'}\big(u_t^l - V_{th}\big)\big)/\tau_m$. This procedure is performed ad each time step, thus suitable for online time stream learning. And for offline batch training, due to STOL requires state preservation for error backpropagation, it is imperative to reset u_{t-1}^l and s_{t-1}^l of each layer to 0 at the start of each batch to update the inconsistent neuron states, which will further ensure the accuracy.

STOL also guarantees the condition of *completeness* because it performs complete spatial temporal backpropagation in recent two moments. Therefore, our approach provides a more reasonable online gradient computing paradigm to obtain a better performance. And for offline tasks which usually simulate a time interval with length of T, STOL updates the parameters T times, whereas BPTT updates only once. Consequently, our approach demonstrates swifter convergence and proficiently minimizes the loss function towards a smaller local minima.

3.4 Deviation in STOL

To enhance our comprehension of STOL, we proceed to derive its gradient deviation in detail. The results exhibit that the gradient deviation in STOL is almost negligible, primarily because we ensure minimal bias accumulation and weight inconsistency.

Firstly, as STOL disregards the long-term historical gradients, which is $\sum_{\tau=0}^{t-2} \delta_\tau^l$, there might be problem of precision loss. However, this effect is deemed to be negligible due to the pervasive issue of gradient vanishing that occurs due to the spatio-temporal depths in neural networks, especially in SNNs that have inaccurate surrogate gradient.

In Subsection 3.2, we establish the general deviation equation, now consider $i = t$, $j = t - 1$, which is the situation of STOL, so the total deviation should be $\epsilon^l = \epsilon_{t-1}^l + \epsilon_t^l$. As $\Delta w_t^l = 0$, we can get $c_t^l = 0$ and $\epsilon_t^l = 0$. Plugging these conditions into Equation (9) yields $\epsilon_{t-1}^l = c_{t-1}^l + \epsilon_{t-1}^{l+1} w_{t-1}^{l+1} \theta_{t-1}^l$, expanding this recursion to the output layer L eventually yields:

$$\epsilon^l = \epsilon_{t-1}^l = \sum_{i=l}^{L} c_{t-1}^i \prod_{j=l}^{i-1} w_t^{j+1} \theta_{t-1}^j, \tag{12}$$

this result can also obtained by directly expanding Equation (6). Since the magnitude of weight change Δw_{t-1}^{l+1} produced by each single-step gradient descent is minimal, the influence of c_{t-1}^l is considered negligible. However, the error ϵ_{t-1}^l, accumulated spatially due to the deepening of the layers, may exert an impact on initial layers of the network, which is a common problem of online training algorithms.

We can see from above that STOL effectively reduces the gradient deviation from the complex form given by Equation (6) to a negligible effect. This is consistent with our expected results.

4 Experiments

Datasets and Models

In order to enable a more straightforward comparison with other studies, we employ the VGG and ResNet models, which are frequently used in contemporary research, and the latter encompasses MS-ResNet and SEW-ResNet. In the rest of this paper, we use ResNet to refer to MS-ResNet. Our experiments are carried out on a variety of benchmark tasks in static image recognition and neuromorphic visual classification.

CIFAR-10: The CIFAR-10 [21] dataset consists of 60,000 natural images in 10 classes, with 6,000 images per class. The number of training images is 50,000, and that of test images is 10,000. We utilize the ResNet18 model to conduct our experiments on the CIFAR-10 dataset. This choice of model aligns with current standards in the field and allows for a fair comparison of results.

CIFAR-100: The CIFAR-100 dataset [21] is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. We also employ the ResNet18 model to conduct our experiments on this dataset.

ImageNet: The ImageNet [8] has more than 1250k training images and 50k test images. The utilization of this large-scale dataset serves as a robust means to validate the applicability of our method in complex tasks. For this experiment, we employ the SEW-ResNet34 model, a variant of the deep residual learning method in SNNs that has more efficient information representation capabilities.

CIFAR10-DVS: The CIFAR10-DVS [7] dataset is the neuromorphic version of the CIFAR-10 dataset. It is composed of 10,000 examples in 10 classes, with 1000 examples in each class. Considering that the CIFAR10-DVS dataset does not intrinsically provide a division into training and testing sets, we adopt the same data partitioning strategy as delineated in the study by [12]. Subsequently, we employ the VGG11 network architecture for this task.

DVS128-Gesture: The DVS128-Gesture [1] dataset is recorded by a dynamic vision sensor, which contains 11 kinds of hand gestures from 29 subjects under 3 kinds of illumination conditions. For this particular task, we utilize the VGG-11 network architecture.

Considering that the static datasets lack a temporal dimension, it becomes necessary to encode them into a format that is compatible with SNNs. In our experiments, we employ direct encoding for this purpose. The intensity of the pixels was replicated T times as the input of the model, in order to retain as much information from the original data as possible. For the neuromorphic datasets, we adopted

Table 1: Comparison with other SNN training methods, includes some advanced online training methods and offline training methods. In the table, the bold rows highlight the online methods, and the underling in the last column highlight the best results of each task.

	Method	Network	Time Steps	Online	Mean±Std(Best)
CIFAR-10	Dspike [23]	ResNet-18	6	✘	94.25%
	TET [9]	ResNet-19	6	✘	94.50%
	SLTT [26]	ResNet-18	6	✘	94.44±0.21%(94.59%)
	LTL-Online [37]	**ResNet-20**	**16**	✔	**94.82%**
	OTTT [36]	**VGG(sWS)**	**6**	✔	**93.52±0.06%(93.58%)**
	STOL(ours)	**ResNet-18**	**6**	✔	**95.47%**
CIFAR-100	RecDis [14]	ResNet-19	4	✘	74.10±0.13%
	Dspike [23]	ResNet-18	6	✘	74.24%
	TET [9]	ResNet-19	6	✘	74.72%±0.28%
	SLTT [26]	ResNet-18	6	✘	74.38±0.30%(74.67%)
	OTTT [36]	**VGG(sWS)**	**6**	✔	**71.05±0.04%(71.11%)**
	STOL(ours)	**ResNet-18**	**6**	✔	**74.71%**
ImageNet	TET [9]	ResNet-34	6	✘	64.79%
	SEW [10]	Sew ResNet-34	4	✘	67.04%
	SLTT[26]	NF-ResNet-34	6	✘	66.19%
	LTL-Online [37]	**ResNet-20**	**16**	✔	**56.24%**
	OTTT [36]	**NF-ResNet-34**	**6**	✔	**65.15%**
	STOL(ours)	**Sew ResNet-34**	**6**	✔	**64.90%**
DVS-CIFAR10	Dspike [23]	ResNet-18	10	✘	75.40±0.05%
	TET [9]	VGG-11	10	✘	83.17±0.15%
	SLTT [26]	VGG-11	10	✘	82.20±0.95%(83.10%)
	FPTT [39]	**LTC-SRNN**	**20**	✔	**72.3%**
	OTTT [36]	**VGG(sWS)**	**10**	✔	**76.63±0.34%(77.10%)**
	STOL(ours)	**VGG-11**	**10**	✔	**84.30%**
DVS128-Gesture	PLIF [11]	8-layer CNN	20	✘	97.57%
	SEW [10]	Sew ResNet	16	✘	97.92%
	SLTT [26]	VGG-11, VGG-11(WS)	20	✘	97.92%, 98.50±0.21%(98.62%)
	FPTT [39]	**LTC-SRNN**	**20**	✔	**97.22%**
	OTTT [36]	**VGG(sWS)**	**20**	✔	**96.88%**
	STOL(ours)	**VGG-11**	**20**	✔	**98.26±0.31%(98.61%)**

the widely utilized method of event-to-frame integration [11]. This method transforms the event stream into a sequence of frames, enabling the data to be processed using convolutional neural networks (CNNs) [20].

Table 2: Different training hyper-parameters configuration for each task.

Dataset	Epoch	Batch	LR	L2	T	τ_m
CIFAR-10	200	64	5e-2	5e-5	6	2
CIFAR-100	300	128	2e-2	5e-4	6	2
CIFAR10-DVS	300	64	2.5e-2	5e-4	10	4
DVS128-Gesture	200	32	1e-2	5e-3	20	2
ImageNet	100	200	1e-1	0	6	2

We employ the stochastic gradient descent (SGD) optimizer for the training across all experiments. Nonetheless, we adjust the learning rates, L2 regularization parameters, and other hyper-parameters to suit the specific characteristics inherent to each dataset. Detailed information regarding these adjustments is provided in Table 2. The code for experiments is available[1].

Comparison with SOTA Methods

We compare our results with those obtained from state-of-the-art offline and online training approaches. The comparative results are presented in Table 1, our method outperforms other approaches on these neuromorphic datasets, achieving accuracy of 84.3% on CIFAR10-DVS and 98.26% on DVS128-Gesture. Furthermore, our method surpasses other state-of-the-art methods with a test accuracy of 95.47% on CIFAR-10. Additionally, our method scores 74.71% on the CIFAR-100 and 64.9% on the ImageNet datasets. The aggregate outcomes of these experiments suggest that our methodology

[1] https://github.com/1xueLang/SpatioTemporalOnlineLearning

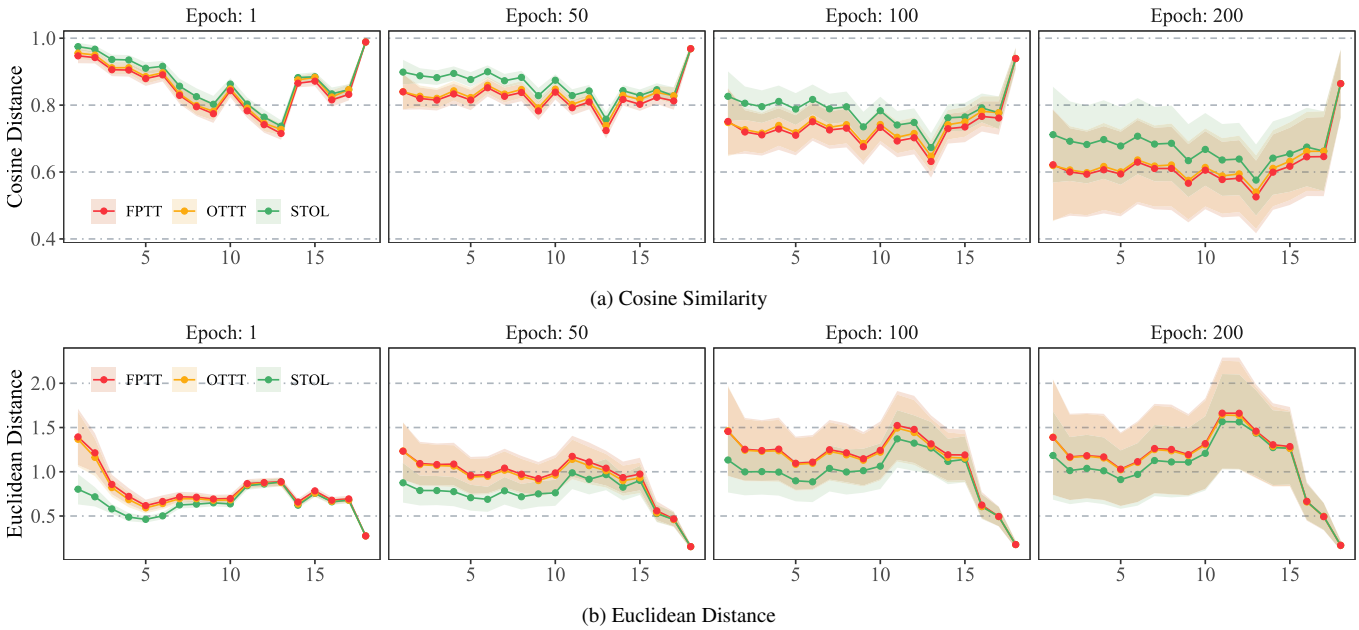

Figure 2: Gradient similarity of weights in ResNet18 between BPTT and three online training algorithms, tested on CIFAR-10. The figure shows the mean (solid line) and standard deviation (light-colored band) of cosine similarity (top) and Euclidean distance (bottom) at four training epochs: 1, 50, 100, and 200. The horizontal axis represents the index of the layer number.

can equal or even exceed the precision of offline BPTT. Importantly, this caliber of performance is maintained across a broad spectrum of models and tasks of varying complexities. Furthermore, our method requires only a minimal time step to obtain a highly accurate result. This demonstrates that our method can fully leverage the information representation capability of SNNs. In conclusion, these results demonstrate a high degree of model fit and generalization capability, further attesting to the effectiveness of our method and highlighting its potential for practical applications in the field.

Although these datasets cover the most commonly used datasets in SNNs research, but they all belong to image recognition tasks, so there may be considerations regarding the generality of STOL in different types of machine learning tasks. It is reasonable to think that STOL should have the same application range as BPTT, and in long sequence tasks, such as speech recognition, even more advantages. This is because of the efficient resource utilization and significantly more training opportunities, in addition to the errors accumulated in temporal depth because of the inherent inaccurate surrogate gradient. And compared with existing online training methods, our method has better generality on datasets and models, such as the requirement of NF technique for OTTT and the reliance on LSN for FPTT as we mentioned in related work.

Ablation 1: Completeness

We have posited in previous sections that one of the crucial factors that influence the performance of online training is the completeness of the gradient. This necessitates the provision of spatio-temporal backpropagation that is pertinent to the model structure, which emphasis the important role of the SHG. Here, we design and conduct experiments to empirically validate the correctness of this perspective. Figure 1 presents three distinct online training algorithms, each characterized by a different backpropagation path. By integrating the BPTT algorithm and comparing the training performance across these four methods, we can validate the correctness of our perspective on completeness.

Firstly, we train ResNet18 with these four algorithms to perform the recognition task of CIFAR-10 dataset, and compare the final test error of the respective trained models while keeping the experimental conditions consistent. In order to speed up the training process, we adopt mixed precision training here, and the results are shown in Table 3. It is clear that when training with the STOL and BPTT, which ensure gradient completeness, the test accuracy is significantly superior compared to the other two algorithms. This underscores the pivotal role of gradient completeness. Furthermore, the performance of our proposed STOL is comparable to the results achieved by the BPTT, which demonstrates that our method attains an acceptable level of completeness.

Table 3: Ablation result of completeness. Tested on CIFAR-10, ResNet18.

Method	Accuracy/%	SGH	TGH
BPTT	94.56	w/	w/
FPTT	92.99	w/o	w/o
OTTT	94.15	w/o	w/
STOL	94.61	w/	w/

To more intuitively illustrate the deviation of the gradients, we calculate the similarity between the gradients determined by three online training algorithms and those determined by BPTT. The results of this comparison are presented in Figure 2. Initially, we train a model using BPTT, subsequently saving the model parameters at four distinct training stages. Following this, we compute the gradient similarity for each respective stage. For the similarity calculation, the same parameters are loaded into the models and the gradients at each time step are calculated by four algorithms and accumulated, while keeping the models' weight unchanged. Then the gradient similarity

is calculated. From Figure 2a we can see that STOL always provides more accurate gradient direction than other two methods. This observation aligns with our theoretical understanding.

Ablation 2: Timeliness

The second theoretical premise of our research is the timeliness of long-term historical gradients; in Section 3 we give a theoretical analysis of this assumption, and here we will present an experimental verification.

In this experiment, we propagate the instantaneous error among the nearest k time steps at each moment, and then update the weights online according to the computed gradient; the values of k range from 1 to T. We conduct experiments on CIFAR-10 using ResNet18, with $T = 6$.

To minimize any potential interference, we remove BN layers from the model. Figure 3 presents the results of the experiment. It is worth

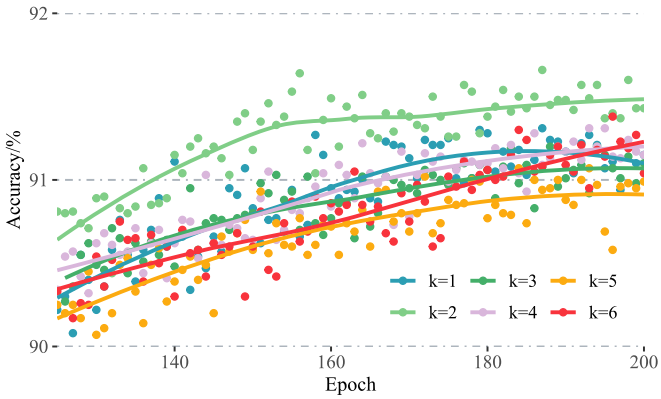

Figure 3: Test accuracy when historical gradients of different lengths participate in training (Ablation result of timeliness).

noting that when k is set to 2, it corresponds to the STOL method, and when k is set to 1, it aligns with the FPTT method. Our experiments reveal that when k is set to 1 or exceeds 2, the final prediction error is approximately 0.5% to 1% higher compared to when k is set to 2. It is indicated that propagating the error over an extended time range does not necessarily enhance model convergence. In fact, it can lead to significant interference; when k is set to 2, which ensures both completeness and timeliness of the gradients, it produces the most effective training result.

Efficient Resource Utilization

Our approach not only matches the training precision of BPTT, but also demonstrates a much superior resource utilization efficiency. This allows for a better exploitation of the hardware's computational capacity, thereby enhancing the acceleration effect. This effect becomes increasingly pronounced as the scale of the network and input grows. Given that our approach necessitates the propagation of the error gradient only in the last two moments, the memory utilization of the algorithm is significantly improved when the time step length exceeds 2, and that is far greater than 2 in SNNs at present. Thus, our approach is capable of training deep learning tasks with much larger scale on devices with equivalent memory capacity. Besides, efficient memory utilization will further alleviate the reduction in computational capacity caused by memory scheduling, thereby reducing the training duration required for the same task to varying extents.

We validate the efficient resource utilization of our approach on multiple tasks. The computational device employed for these tasks is

Table 4: Comparison of computation and memory efficiency between BPTT and STOL. The column **Time** represents the time consumed for each epoch.

	T	Arch	Method	Space/G	Time/s
CIFAR-10	2	ResNet18	STOL	2.199	33
			BPTT	2.853	29
	8	ResNet18	STOL	2.221	132
			BPTT	5.903	168
	32	ResNet18	STOL	2.221	541
			BPTT	18.259	1601
CIFAR10-DVS	10	VGG-11	STOL	2.489	38
			BPTT	5.459	38
DVS128-Geature	20	VGG-11	STOL	3.959	35
			BPTT	21.575	64

an NVIDIA RTX3090 graphics card, equipped with 24GB of video memory. The results are shown in Table 4. Initially, we verify the computational efficiency of both BPTT and STOL on the CIFAR-10 dataset, spanning tasks of small, medium, and large scales. With T set to 2, 8, and 32, respectively, corresponding to the three scales, the results indicate that STOL maintains higher memory efficiency on smaller tasks, and enhances it by a factor of 2 to 10 on tasks of medium and large scales; while significantly improves computational efficiency as well. Besides, it is evident that the computing time of STOL increases steadily with increasing T. In contrast, there is a substantial surge in the increase of the computing time of BPTT. This substantiates that the extensive memory footprint will become the bottleneck in the computation, which will further considerably restrict the computational capacity of the device. Subsequently, we verify the computational efficiency of both BPTT and STOL on tasks such as CIFAR10-DVS and DVS128-Gesture, which mirrors the execution of the algorithms on real-world datasets. As evidenced by the results, the memory efficiency of STOL witnesses an enhancement by a factor ranging from 2 to 5. In a similar vein, the computational efficiency of BPTT is reduced by half when there is a high memory occupancy.

5 Conclusion

In this paper, we conduct a comprehensive analysis of the efficacy of gradient computation for online learning in SNNs, which has been overlooked in recent research. We propose two pivotal factors, completeness and timeliness, which ensure the accuracy of the error backpropagation, thereby reducing the calculation bias and improving the performance of online training. Leveraging these findings, we propose spatio-temporal online learning (STOL), which substantially advances the accuracy of the online gradients, therefore outperforms existing online training algorithms. Combining with the advantage of fast convergence of online learning, the training performance of STOL can reach or even surpass that of BPTT.

In addition, we conduct experiments to verify the computation and memory complexity of our approach, and the results show that STOL can maintain a fixed low memory overhead when training tasks with different simulated T, therefore further exploiting the computational capacity of the devices, such advantage makes STOL more suitable for application on neuromorphic devices.

Acknowledgements

This work was supported by the National Science Foundation of China under Grant 62236007 and the Young Scientists Fund of the Natural Science Foundation of Sichuan Province under Grant 2024NSFSC1469.

References

[1] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

[2] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

[3] T. Bohnstingl, S. Woźniak, A. Pantazi, and E. Eleftheriou. Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[4] A. Brock, S. De, and S. L. Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. *arXiv preprint arXiv:2101.08692*, 2021.

[5] A. Brock, S. De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071. PMLR, 2021.

[6] N. Caporale and Y. Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[7] W. Cheng, H. Luo, W. Yang, L. Yu, and W. Li. Structure-aware network for lane marker extraction with dynamic vision sensor. *arXiv preprint arXiv:2008.06204*, 2020.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] S. Deng, Y. Li, S. Zhang, and S. Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

[10] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.

[11] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021.

[12] W. Fang, Z. Yu, Z. Zhou, D. Chen, Y. Chen, Z. Ma, T. Masquelier, and Y. Tian. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama. Efficient training of spiking neural networks with temporally-truncated local backpropagation through time. *Frontiers in Neuroscience*, 17:1047008, 2023.

[14] Y. Guo, X. Tong, Y. Chen, L. Zhang, X. Liu, Z. Ma, and X. Huang. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2022.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Y. Hu, L. Deng, Y. Wu, M. Yao, and G. Li. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[18] A. Kag and V. Saligrama. Training recurrent neural networks via forward propagation through time. In *International Conference on Machine Learning*, pages 5189–5200. PMLR, 2021.

[19] J. Kaiser, H. Mostafa, and E. Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14: 515306, 2020.

[20] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz. Review on convolutional neural networks (cnn) in vegetation remote sensing. *ISPRS journal of photogrammetry and remote sensing*, 173:24–49, 2021.

[21] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[23] Y. Li, Y. Guo, S. Zhang, S. Deng, Y. Hai, and S. Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021.

[24] H. Liu, Y. Chen, Z. Zeng, M. Zhang, and H. Qu. A low power and low latency fpga-based spiking neural network accelerator. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2023.

[25] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[26] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, and Z.-Q. Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6166–6176, 2023.

[27] E. O. Neftci, H. Mostafa, and F. Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[28] T. Ortner, L. Pes, J. Gentinetta, C. Frenkel, and A. Pantazi. Online spatio-temporal learning with target projection. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–5. IEEE, 2023.

[29] N. Rathi and K. Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34 (6):3174–3182, 2021.

[30] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[31] I. Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

[32] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[33] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Backpropagation*, pages 433–486. Psychology Press, 2013.

[34] Y. Wu, L. Deng, G. Li, and L. Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:323875, 2018.

[35] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1311–1318, 2019.

[36] M. Xiao, Q. Meng, Z. Zhang, D. He, and Z. Lin. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35:20717–20730, 2022.

[37] Q. Yang, J. Wu, M. Zhang, Y. Chua, X. Wang, and H. Li. Training spiking neural networks with local tandem learning. *Advances in Neural Information Processing Systems*, 35:12662–12676, 2022.

[38] M. Yao, G. Zhao, H. Zhang, Y. Hu, L. Deng, Y. Tian, B. Xu, and G. Li. Attention spiking neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[39] B. Yin, F. Corradi, and S. M. Bohté. Accurate online training of dynamical spiking neural networks through forward propagation through time. *Nature Machine Intelligence*, 5(5):518–527, 2023.

[40] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11062–11070, 2021.