Efficient Online Learning for Networks of Two-Compartment Spiking Neurons

Yujia Yin
Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
yujia.yin@connect.polyu.hk

Xinyi Chen
Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
xinyi-97.chen@connect.polyu.hk

Chenxiang Ma
Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
chenxiang.ma@connect.polyu.hk

Jibin Wu*
Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
jibin.wu@polyu.edu.hk

Kay Chen Tan
Department of Computing
The Hong Kong Polytechnic University
Hong Kong SAR, China
kctan@polyu.edu.hk

***Abstract*—The brain-inspired Spiking Neural Networks (SNNs) have garnered considerable research interest due to their superior performance and energy efficiency in processing temporal signals. Recently, a novel multi-compartment spiking neuron model, namely the Two-Compartment LIF (TC-LIF) model, has been proposed and exhibited a remarkable capacity for sequential modelling. However, training the TC-LIF model presents challenges stemming from the large memory consumption and the issue of gradient vanishing associated with the Backpropagation Through Time (BPTT) algorithm. To address these challenges, online learning methodologies emerge as a promising solution. Yet, to date, the application of online learning methods in SNNs has been predominantly confined to simplified Leaky Integrate-and-Fire (LIF) neuron models. In this paper, we present a novel online learning method specifically tailored for networks of TC-LIF neurons. Additionally, we propose a refined TC-LIF neuron model called Adaptive TC-LIF, which is carefully designed to enhance temporal information integration in online learning scenarios. Extensive experiments, conducted on various sequential benchmarks, demonstrate that our approach successfully preserves the superior sequential modeling capabilities of the TC-LIF neuron while incorporating the training efficiency and hardware friendliness of online learning. As a result, it offers a multitude of opportunities to leverage neuromorphic solutions for processing temporal signals.**

I. Introduction

The human brain, with a power consumption of only 20 Watts, is capable of processing complex sensory signals efficiently [1]. Mimicking the discrete, spike-based information processing observed in the brain, spiking neural networks (SNNs) have been introduced for artificial intelligence (AI) applications [2], [3], [4]. Remarkably, owing to the introduction of the backpropagation through time (BPTT) learning algorithm with surrogate gradients [5], [6], SNNs have achieved promising performance across a wide range of AI tasks [7], [8], [9]. Moreover, these models demonstrate superior energy efficiency when deployed on ultra-low-power neuromorphic hardware [10], [11], [12].

However, current SNNs still face challenges in effectively processing sensory signals compared to their biological counterparts [13], [14]. One contributing factor is the use of overly simplified neuron models, such as the widely adopted Leaky Integrate-and-Fire (LIF) model [15], [16], which has limited capacity to store information over long time windows. To overcome this limitation, recent research has introduced slow-decaying variables into spiking neuron models to enrich their neuronal dynamics and improve their ability to process complex temporal signals [17], [18], [19]. Yet, these models struggle to establish long-term temporal dependencies that are crucial for effective temporal signal processing. Another line of research has explored the use of self-attention mechanisms to enable more flexible integration of temporal information within SNNs [20], [21], [22]. However, these models require significant memory resources that are currently unavailable on neuromorphic chips. Recently, a two-compartment LIF (TC-LIF) neuron model has been proposed, which captures the intricate neuronal structure and rich temporal dynamics observed in biological neurons [23]. By incorporating the interaction between the somatic and dendritic compartments, the TC-LIF neuron demonstrates superior sequential modeling capabilities in many challenging temporal processing tasks.

Despite the impressive performance of SNNs utilizing TC-LIF neurons in sequential modeling, they face significant challenges during BPTT training. As depicted in Fig. 1, the BPTT algorithm necessitates storing all internal states over the whole time window for gradient updates, leading to a linear increase in memory requirements as time window increases. Additionally, the BPTT algorithm is susceptible to issues like vanishing or exploding gradients [24], [25], which further complicates the task of stable training due to the necessity of carefully initializing the parameters. Furthermore, the BPTT algorithm requires the entire sequence to be available before

*Corresponding Author
This manuscript is under review by 2024 International Joint Conference on Neural Networks (IJCNN 2024).

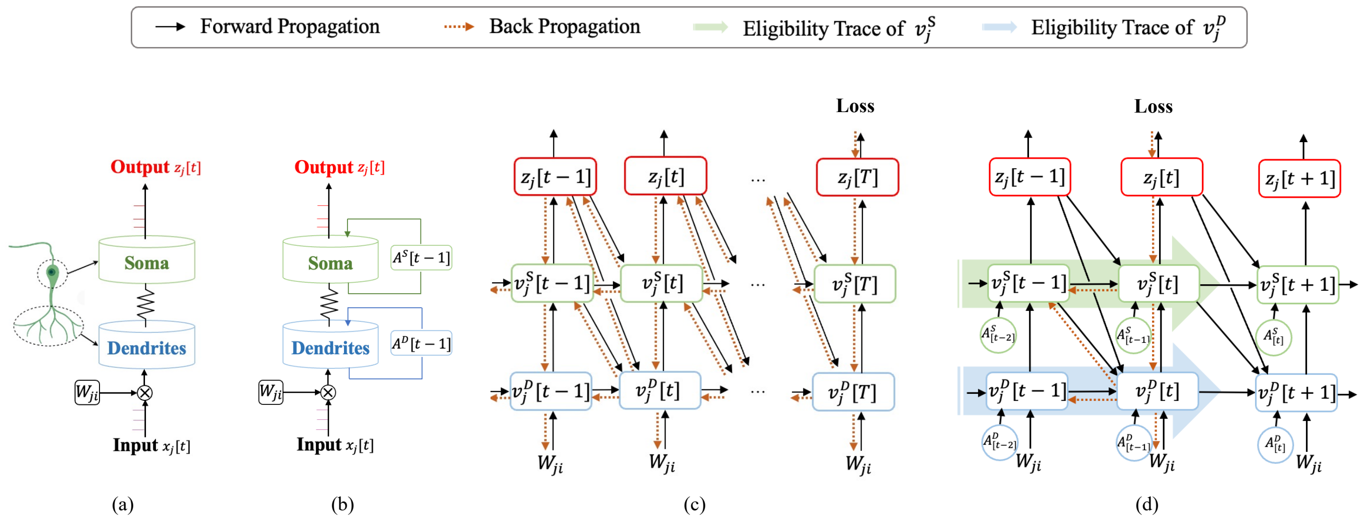


Fig. 1: Comparison of the proposed neuron models and learning algorithms. **(a)** The vanilla TC-LIF neuron captures the interaction between dendritic and somatic compartments in biological neurons to facilitate temporal signal processing. **(b)** The proposed Adaptive TC-LIF neuron further introduces time-varying membrane potentials decaying constants to facilitate temporal information integration during online learning. **(c)** The vanilla TC-LIF model is trained using BPTT algorithm, where the gradients are propagated from the last time step to all preceding time steps for parameter update. **(d)** Our proposed Adaptive TC-LIF neuron facilitates efficient online learning by computing parameter updates at each time step based on the local loss and eligibility trace derived at that specific time step. Importantly, the computation of eligibility traces for both neuronal compartments occurs in a forward manner, eliminating the need to store intermediate network states as required in BPTT training.

training, making it incompatible with online learning scenarios [26], [27], [28].

To tackle these challenges, this study explores the biologically plausible and efficient online training approach known as e-prop [27]. The e-prop algorithm is based on the concept of eligibility trace, which enables the local and feedforward computation of the error gradient at each time step. Notably, e-prop has demonstrated comparable performance to BPTT on a number of AI benchmarks [29], [30]. However, it is worth noting that this method has primarily been investigated using simplified LIF neurons, and its applicability to more complex multi-compartment neuron models has yet to be explored [17]. Notably, the TC-LIF neuron was initially designed to work with the BPTT training algorithm, which necessitates the adaptation of its parameter space when transitioning to online learning settings.

In this study, we present an extension of the e-prop formulation to accommodate the unique neuronal dynamics of the TC-LIF neuron model. Additionally, we propose an Adaptive TC-LIF model specifically designed for online learning scenarios. The traditional BPTT algorithm suffers from the issue of gradient vanishing, which restricts the leakiness of the membrane potential in the original TC-LIF model. However, the introduction of the e-prop algorithm addresses this limitation, allowing for more flexible selection of the membrane potential decaying constants. To ensure a balanced influence of inputs from different time steps on parameter updates, we further propose to set the membrane potential decaying constants to be time-varying. Experimental results demonstrate that our proposed approach successfully preserves the superior sequential modeling ability of TC-LIF neurons while incorporating the efficiency of online learning. The main contributions of this work can be summarized as follows:

- We expand upon the e-prop online learning approach, which was initially designed for LIF neurons, and adapt it to accommodate multi-compartment TC-LIF neurons. We provide comprehensive and rigorous mathematical derivations for the proposed online learning method.
- We propose a novel Adaptive TC-LIF model that incorporates time-varying membrane potential decaying constants. This enhances temporal information integration in online learning scenarios.
- We perform comprehensive experiments on a range of sequential modeling tasks to evaluate the performance of our proposed Adaptive TC-LIF model. The results substantiate that our model exhibits exceptional sequential modeling capacity, high training efficiency, and neuromorphic hardware friendliness.

II. Preliminaries

A. Spiking Neuron Models

Motivated by the intricate structure and dynamic firing behavior of biological neurons, numerous spiking neuron models have been proposed in the literature [31], [32], [33]. These models offer SNNs superior capabilities in processing temporal signals. Typically, the neuronal dynamics of these spiking neurons encompass three fundamental components: charging, firing, and resetting. For instance, the discrete-time equations below can be used to model recurrently connected [17], [34] LIF neurons:

$$v_j[t+1] = \alpha v_j[t] - v_{th} z_j[t] + I_j[t+1], \tag{1}$$

$$I_j[t+1] = \sum_{i \neq j} W_{ji}^{rec} z_i[t] + \sum_i W_{ji}^{in} x_i[t+1] + b, \tag{2}$$

$$z_j[t+1] = H(v_j[t+1] - v_{th}), \tag{3}$$

where W_{ji}^{in} and W_{ji}^{rec} denote the feedforward and recurrent connections from the presynaptic neuron i to the postsynaptic neuron j, respectively. The parameter α represents the membrane decaying coefficient, which ranges between 0 and 1. The variable $I_j[t+1]$ represents the input current at time $t+1$. During the charging process, the membrane potential $v_j[t]$ is updated until it reaches the firing threshold v_{th}, at which point an output spike $z_j[t]$ is generated. Subsequently, after firing, the membrane potential is reset.

While the LIF neuron demonstrates promising results in tasks with limited temporal context, it encounters difficulties in retaining information over longer time periods (e.g., a few hundred time steps), resulting in poor performance for tasks that require long-term memory. This limitation arises from the exponential decay of inputs to LIF neurons during their membrane potential update. To enhance the sequential modeling capacity of SNNs, a recent study has introduced a novel spiking neuron model called TC-LIF [23]. The TC-LIF neuron is specifically designed to replicate the interaction between dendritic and somatic compartments of biological neurons. The neuronal dynamics of the TC-LIF neuron can be formulated as follows:

$$v_j^D[t+1] = v_j^D[t] + \beta_1 v_j^S[t] - \gamma z_j[t] + I_j[t+1], \tag{4}$$

$$v_j^S[t+1] = v_j^S[t] + \beta_2 v_j^D[t+1] - v_{th} z_j[t], \tag{5}$$

$$I_j[t+1] = \sum_{i \neq j} W_{ji}^{rec} z_i[t] + \sum_i W_{ji}^{in} x_i[t+1] + b, \tag{6}$$

$$z_j[t+1] = H(v_j^S[t+1] - v_{th}). \tag{7}$$

The TC-LIF neuron exhibits two distinct neuronal states, denoted as $h_j[t] = [v_j^D[t], v_j^S[t]]$. Notably, $v_j^D[t]$ corresponds to the dendritic compartment, which is responsible for storing long-term memory. On the other hand, $v_j^S[t]$ represents the somatic compartment, which is used to store short-term memory. The interaction between these two compartments is regulated by the coefficients β_1 and β_2. Specifically, β_1 is defined as $-\sigma(c_1)$, and β_2 is defined as $\sigma(c_2)$, where $\sigma(\cdot)$ denotes a sigmoid function. Both c_1 and c_2 are learnable parameters that are jointly trained with network weights. Following the somatic firing, both compartments undergo a reset in their membrane potential. It is worth noting that the reset of the dendritic compartment involves the initiation of a backpropagating spike, the effect of which is governed by the hyperparameter γ.

B. Challenges of Training SNNs with BPTT

Training SNNs presents a significant challenge due to the intricate spatial-temporal dependency between distant spikes and the discrete nature of spike generation. However, recent advancements have addressed these issues by applying BPTT algorithm [35] coupled with the surrogate gradient method. To illustrate how BPTT algorithm can be applied across LIF and TC-LIF spiking neurons, we denote the observable state (i.e., spike) of neuron j at time t as $z_j[t]$, the hidden state as $h_j[t]$ (i.e., membrane potential), and the input as $x_j[t]$. The update of the hidden state of a neuron can be expressed as $h_j[t] = f(h_j[t-1], z_j[t-1], x_j[t], W_{ji})$. The weight gradients can be expressed as follows:

$$\frac{dE}{dW_{ji}} = \sum_{t=1}^{T} \frac{dE}{dz_j[T]} \frac{\partial z_j[T]}{\partial W_{ji}}, \tag{8}$$

where E represents the error term. It is worth noting that by expanding the above gradient expression using the chain rule, as shown in (9), it becomes apparent that the BPTT algorithm optimizes SNN parameters by unfolding the network state over time and then applying the backpropagation algorithm. However, this process requires the storage of intermediate states for all neurons, which imposes a significant computational burden.

$$\frac{dE}{dW_{ji}} = \sum_{t=1}^{T} \frac{dE}{dz_j[T]} \frac{\partial z_j[T]}{\partial h_j[T]} \frac{\partial h_j[T]}{\partial h_j[t]} \frac{\partial h_j[t]}{\partial W_{ji}}. \tag{9}$$

C. Online learning with e-prop algorithm

The e-prop algorithm [27] has emerged as an alternative to the BPTT training algorithm. This biologically inspired online learning algorithm consists of two essential components: learning signals and eligibility traces. The learning signals encompass various top-down signals in the brain, such as neuromodulators and error-related neural activities. These signals convey information to neuronal populations about behavioral outcomes. On the other hand, eligibility traces refer to molecular-level traces exist within each neuron, such as calcium ions or activated CaMKII enzyme. These traces represent fading memories of past events. Supported by rigorous mathematical derivation [27], the effective integration of eligibility trace and learning signals can lead to high-performance online learning.

As given in (10), the impact of future time steps from $t+1$ to T on the present state t is disregarded in e-prop, allowing the weight gradient $\frac{dE}{dW_{ji}}$ to be expressed as a summation of products across time steps 1 to t. The second term of this equation represents a local gradient that remains independent of error E:

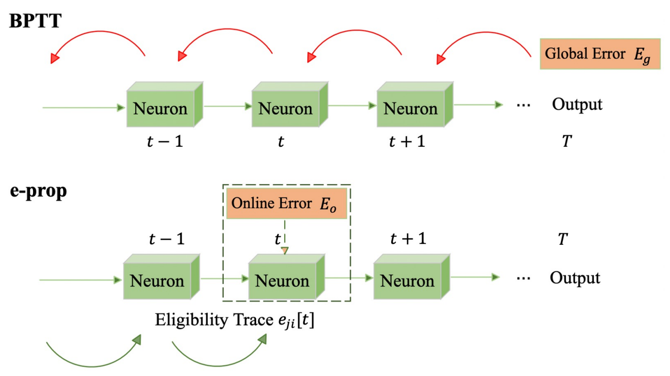


Fig. 2: The comparison of gradient update principle of BPTT and e-prop algorithms at time t. BPTT algorithm updates the network parameters based on the global loss, which can only be obtained at the last time step T. In contrast, the e-prop algorithm accumulates eligibility traces that propagate forward in time and multiplies them with the online error term. This allows for the online update of network parameters.

$$\frac{dE}{dW_{ji}} = \sum_t \frac{dE}{dz_j[t]} \left(\frac{dz_j[t]}{dW_{ji}} \right)_{local}. \tag{10}$$

In e-prop, the component $\frac{dE}{dz_j[t]}$ is further approximated with an online learning signal $L_j[t] \triangleq \frac{\partial E}{\partial z_j[t]}$, where E is the cross entropy loss. Furthermore, the eligibility traces $e_{ji}[t] \triangleq \left(\frac{dz_j[t]}{dW_{ji}}\right)_{local}$ aim to capture the maximum amount of information about the network gradient, which can be updated during the forward computation. Consequently, the computation of the gradient in e-prop involves a combination of the learning signal and the eligibility trace as:

$$\frac{dE}{dW_{ji}} = \sum_t L_j[t] \cdot e_{ji}[t]. \tag{11}$$

In the case where the learning signal assumes an ideal value, we can provide rigorous proof demonstrating that the eligibility trace effectively preserves the maximum amount of gradient information [27]. In the e-prop algorithm, the derivation of the eligibility trace accounts for the evolution process of the hidden state within the neuron, eliminating the need to store all hidden states by disregarding the influence of future time steps. The specific expression of the eligibility trace is:

$$e_{ji}[t] = \frac{\partial z_j[t]}{\partial h_j[t]} \underbrace{\sum_{t' \le t} \frac{\partial h_j[t]}{\partial h_j[t-1]} \cdots \frac{\partial h_j[t'+1]}{\partial h_j[t']} \frac{\partial h_j[t']}{\partial W_{ji}}}_{=\varepsilon_{ji}[t]}. \tag{12}$$

In (12), the first term represents the surrogate gradient $\psi_j[t]$ [5], and the following terms are defined as eligibility vector $\varepsilon_{ji}[t]$. Furthermore, we can derive an iterative expression for the eligibility vector that pertains to the eligibility trace $e_{ji}[t] = \psi_j[t] \cdot \varepsilon_{ji}[t]$ as:

$$\begin{aligned}
\varepsilon_{ji}[t] &\triangleq \sum_{t' \le t} \frac{\partial h_j[t]}{\partial h_j[t-1]} \cdots \frac{\partial h_j[t'+1]}{\partial h_j[t']} \frac{\partial h_j[t']}{\partial W_{ji}} \\
&= \frac{\partial h_j[t]}{\partial W_{ji}} + \frac{\partial h_j[t]}{\partial h_j[t-1]} \frac{\partial h_j[t-1]}{\partial W_{ji}} + \cdots \\
&+ \frac{\partial h_j[t]}{\partial h_j[t-1]} \cdots \frac{\partial h_j[2]}{\partial h_j[1]} \frac{\partial h_j[1]}{\partial W_{ji}} \\
&= \frac{\partial h_j[t]}{\partial W_{ji}} + \frac{\partial h_j[t]}{\partial h_j[t-1]} \cdot \\
&\left(\frac{\partial h_j[t-1]}{\partial W_{ji}} + \frac{\partial h_j[t-1]}{\partial h_j[t-2]} \frac{\partial h_j[t-2]}{\partial W_{ji}} + \cdots \right) \\
&= \frac{\partial h_j[t]}{\partial h_j[t-1]} \varepsilon_{ji}[t-1] + \frac{\partial h_j[t]}{\partial W_{ji}}.
\end{aligned} \tag{13}$$

It is evident that the key aspect of the e-prop algorithm revolves around the derivation of the neuron's eligibility vector, which holds a substantial amount of gradient information and is propagated forward in time.

III. Methodology

A. Eligibility Traces for TC-LIF

Here, we present an elaborate derivation for the eligibility vector of the TC-LIF neuron. By combining the neuronal function of the TC-LIF neuron and the eligibility trace defined in the earlier section, we can obtain the eligibility trace of the dendritic compartment as:

$$\begin{aligned}
\frac{\partial v_j^D[t]}{\partial W_{ji}^{in}} &= \frac{\partial v_j^D[t]}{\partial W_{ji}^{in}} + \frac{\partial v_j^D[t]}{\partial v_j^D[t-1]} \frac{\partial v_j^D[t-1]}{\partial W_{ji}^{in}} \\
&+ \frac{\partial v_j^D[t]}{\partial v_j^S[t-1]} \frac{\partial v_j^S[t-1]}{\partial W_{ji}^{in}} \\
&= x_i[t] + \frac{\partial v_j^D[t-1]}{\partial W_{ji}^{in}} + \beta_1 \frac{\partial v_j^S[t-1]}{\partial W_{ji}^{in}}.
\end{aligned} \tag{14}$$

Similarly, for the somatic compartment, the eligibility trace is formulated as follows:

$$\begin{aligned}
\frac{\partial v_j^S[t]}{\partial W_{ji}^{in}} &= \frac{\partial v_j^S[t]}{\partial v_j^S[t-1]} \frac{\partial v_j^S[t-1]}{\partial W_{ji}^{in}} + \frac{\partial v_j^S[t]}{\partial v_j^D[t]} \frac{\partial v_j^D[t]}{\partial W_{ji}^{in}} \\
&= \frac{\partial v_j^S[t-1]}{\partial W_{ji}^{in}} + \beta_2 \frac{\partial v_j^D[t]}{\partial W_{ji}^{in}}.
\end{aligned} \tag{15}$$

The Eqs. (14) and (15) can be further abstracted into a two-dimensional eligibility vector $\varepsilon_{ji}[t] = [\varepsilon_{ji}^D[t], \varepsilon_{ji}^S[t]] = [\frac{\partial v_j^D[t]}{\partial W_{ji}^{in}}, \frac{\partial v_j^S[t]}{\partial W_{ji}^{in}}]$ and expressed iteratively as follows:

$$\varepsilon_{ji}^D[t] = \varepsilon_{ji}^D[t-1] + \beta_1 \varepsilon_{ji}^S[t-1] + x_i[t], \tag{16}$$
$$\varepsilon_{ji}^S[t] = \varepsilon_{ji}^S[t-1] + \beta_2 \varepsilon_{ji}^D[t]. \tag{17}$$

By plugging (16) into (17), we could rewrite the eligibility vector as:

$$\varepsilon_{ji}^D[t] = \varepsilon_{ji}^D[t-1] + \beta_1 \varepsilon_{ji}^S[t-1] + x_i[t], \tag{18}$$
$$\varepsilon_{ji}^S[t] = (1 + \beta_1\beta_2) \varepsilon_{ji}^S[t-1] + \beta_2 \varepsilon_{ji}^D[t-1] + \beta_2 x_i[t]. \tag{19}$$

As the eligibility trace defined in (12), we are left with the task of deriving $\frac{\partial z_j[t]}{\partial h_j[t]}$, which can be obtained as follows:

$$\begin{aligned}\frac{\partial z_j[t]}{\partial h_j[t]} &\triangleq [\frac{\partial z_j[t]}{\partial v_j^D[t]}, \frac{\partial z_j[t]}{\partial v_j^S[t]}] = \left[\frac{\partial z_j[t]}{\partial v_j^S[t]}\frac{\partial v_j^S[t]}{\partial v_j^D[t]}, \frac{\partial z_j[t]}{\partial v_j^S[t]}\right] \\ &= [\beta_2\psi_j[t], \psi_j[t]],\end{aligned} \tag{20}$$

with

$$\psi_j[t] = \frac{\partial z_j[t]}{\partial v_j^S[t]} = \frac{1}{\gamma^2} max(0, \gamma - |v_j^S[t] - v_{th}|), \tag{21}$$

which is the surrogate gradient [36] function used to overcome the issue of discontinuity arising from spike generation and reset.

As a result, the eligibility trace of the TC-LIF neuron can be calculated as:

$$\begin{aligned}e_{ji}[t] &= \frac{\partial z_j[t]}{\partial h_j[t]} \cdot \varepsilon_{ji}[t] \\ &= \psi_j[t](\beta_2\varepsilon_{ji}^D[t] + \varepsilon_{ji}^S[t]).\end{aligned} \tag{22}$$

B. Redesign TC-LIF for Effective Online Learning

1) Redesigned Parameter Space: The original TC-LIF neuron model has been meticulously designed to work with the BPTT algorithm, wherein the membrane decaying constants of the two neuronal compartments, namely α_1 and α_2, are set to 1 to ensure the preservation of gradient information across a long time window. Such a design is critical for circumventing the vanishing gradient problem commonly encountered in BPTT training. However, within the context of e-prop online learning, the gradient preservation strategy is unnecessary. Unlike BPTT, which relies on the chain rule for gradient updates, e-prop employs an eligibility trace that updates iteratively for online parameter update. As a result, e-prop algorithm is not susceptible to the catastrophic vanishing gradient problem. Hence, we can relax the parameter space of the vanilla TC-LIF neuron to allow the values of α_1 and α_2 to become learnable. This modification can significantly enhance the representation power of TC-LIF neurons. As a result, the modified TC-LIF neuron can be expressed as:

$$v_j^D[t+1] = \alpha_1 v_j^D[t] + \beta_1 v_j^S[t] - \gamma z_j[t] + I_j[t+1], \tag{23}$$
$$v_j^S[t+1] = \alpha_2 v_j^S[t] + \beta_2 v_j^D[t+1] - v_{th} z_j[t], \tag{24}$$
$$I_j[t+1] = \sum_{i\neq j} W_{ji}^{rec} z_i[t] + \sum_i W_{ji}^{in} x_i[t+1] + b, \tag{25}$$
$$z_j[t+1] = H(v_j^S[t+1] - v_{th}), \tag{26}$$

where the value of β_1 and β_2 are fixed.

Furthermore, this redesigned parameter space of α_1 and α_2 can also facilitate the memory integration process by balancing the gradient contributions from different time steps. This point is clearer if we look at the eligibility trace of the somatic compartment of the original TC-LIF neuron as expressed in (19). It is obvious that equal importance β_2 has been assigned to both the input term $x_i[t]$ and the memory term $\varepsilon_{ji}^D[t-1]$. However, we argue that this is not a reasonable strategy. Specifically, the memory term $\varepsilon_{ji}^D[t-1]$ defined in (18) consistently integrates the inputs $x_i[i]$ from $i = 0$ to $t-1$ without any decay, and its impact on weight parameters in (11) will also accumulate over time. As a result, the inputs from earlier time steps' will have a much greater impact than the later time steps, and the model will gradually lose sensitivity to new input data as time progresses. Hence, it is imperative to re-weight the impact of $\varepsilon_{ji}^D[t-1], \varepsilon_{ji}^S[t-1]$, and $x_i[t]$ in (19) to ensure the inputs at different time step can have a more balanced contribution. This issue can be effectively alleviated in our redesigned parameter space described as follows:

$$\varepsilon_{ji}^D[t] = \alpha_1\varepsilon_{ji}^D[t-1] + \beta_1\varepsilon_{ji}^S[t-1] + x_i[t], \tag{27}$$
$$\varepsilon_{ji}^S[t] = (\alpha_2 + \beta_1\beta_2)\varepsilon_{ji}^S[t-1] + \alpha_1\beta_2\varepsilon_{ji}^D[t-1] + \beta_2 x_i[t], \tag{28}$$

where the importance of each item can be independently adjusted by adapting the value of the membrane potential decaying constants α_1, α_2.

2) Time-varying Decaying Constant: As previously discussed, it is essential to balance the influence of inputs from different time steps during the gradient calculation. To further improve the sensitivity on new inputs, we should gradually reduce the contribution of $\varepsilon_{ji}^D[t-1], \varepsilon_{ji}^S[t-1]$ in (28) as the training progresses. In order to achieve this, we introduce modifications to the decaying constants α_1 and α_2 of the membrane potentials, making them time-varying. Specifically, we use two time-varying terms $A^D[t], A^S[t]$ to describe the memory trace decaying rates for the dendritic and somatic compartments, respectively, and we call the resulted neuron model as **Adaptive TC-LIF model**. The neuronal dynamics of the Adaptive TC-LIF model can be sumamrised as:

$$v_j^D[t+1] = A^D[t] \cdot v_j^D[t] + \beta_1 v_j^S[t] - \gamma z_j[t] + I_j[t+1], \tag{29}$$
$$v_j^S[t+1] = A^S[t] \cdot v_j^S[t] + \beta_2 v_j^D[t+1] - v_{th} z_j[t], \tag{30}$$
$$A^D[t] = Clamp\ (\Gamma(t+1, 1/(t+1)),\ a^d,\ 1), \tag{31}$$
$$A^S[t] = Clamp\ (\Gamma(t+1, 1/(t+1)),\ a^s,\ 1), \tag{32}$$

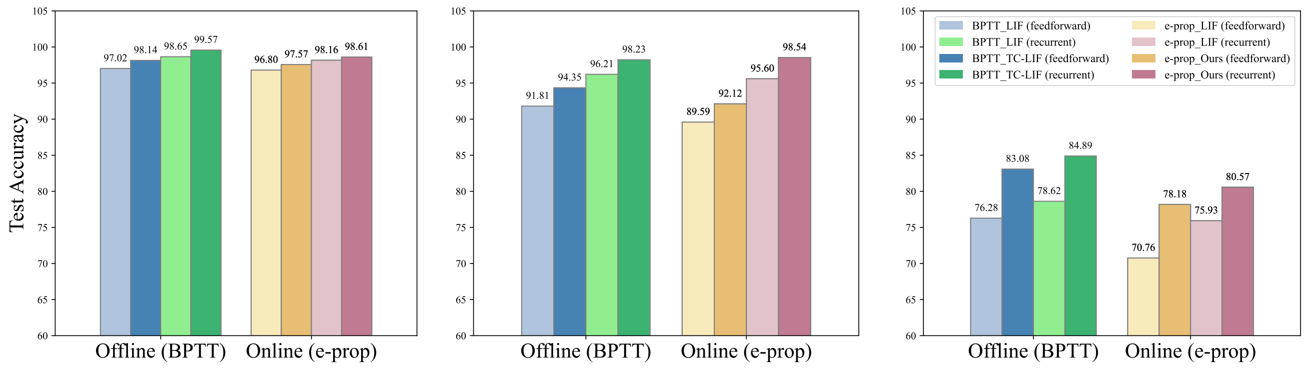
where $\Gamma(\cdot)$ is the gamma distribution function, which follows the long-tail distribution of memory trace discovered in cerebral cortical neurons [37]. a^d and a^s are two learnable coefficients, describing the minimum value of $A^D[t], A^S[t]$, respectively.

C. Extending e-prop to Multi-layer SNNs

The original e-prop algorithm is proposed for single-layer networks. In this work, we extend this algorithm to multi-layer networks to solve more complex sequential tasks [38]. With a single-layer network architecture, the original weight update rule of e-prop can be formulated as:

$$\Delta W_{ji}^o[t] = -\eta L_j[t] \frac{\partial z_j^o[t]}{\partial h_j^o[t]} \frac{\partial h_j^o[t]}{\partial W_{ji}^o}. \tag{33}$$

where η denotes the learning rate, the superscript "o" indicates the output layer, and $\frac{\partial h_j^o[t]}{\partial W_{ji}^o}$ represents the eligibility vector of



(a) S-MNIST (b) PS-MNIST (c) SHD

Fig. 3: Comparison of model performance on three benchmark tasks.

the output layer. We can straightforwardly extend (33) to the hidden layers of multi-layer networks as:

$$\Delta W^l_{ji}[t] = -\eta L_j[t] \frac{\partial z^o_j[t]}{\partial h^o_j[t]} \frac{\partial h^o_j[t]}{\partial z^l_j[t]} \frac{\partial z^l_j[t]}{\partial h^l_j[t]} \frac{\partial h^l_j[t]}{\partial W^l_{ji}}, \qquad (34)$$

where the superscript "l" denotes the l^{th} hidden layer.

IV. Experimental Evaluation

In this section, we first evaluate the online learning performance of the Adaptive TC-LIF neuron model across various sequential modeling benchmarks, including Sequential MNIST (S-MNIST) [39], Permutated Sequential MNIST (PS-MNIST) [39], and Spiking Heidelberg Digits (SHD) [40]. In the case of the S-MNIST and PS-MNIST datasets, the pixels of images are fed into networks row by row. Subsequently, we conduct comprehensive ablation studies to verify the effectiveness of the proposed modifications to TC-LIF neurons. Finally, the memory efficiency of our proposed online training approach has been compared against the BPTT-based training methodologies.

A. Experimental Setups

1) Hyper-parameter and Network Architecture: We provide the detailed settings of network structure and hyperparameters of the Adaptive TC-LIF model in Table I.

2) Training Configuration: We employed the SGD optimizer [41] to train the S-MNIST dataset for 300 epochs, with an initial learning rate of 0.08. The learning rate was adjusted using a cosine schedule. For the PS-MNIST and SHD datasets, we utilized the Adam optimizer [42] and trained the networks for 200 epochs. In the case of the PS-MNIST dataset, both the feedforward and recurrent networks had an initial learning rate of 0.0005, which decayed to 0.8 times the previous value after every 15 epochs. Similarly, for the SHD dataset, both networks used an initial learning rate of $5e^{-5}$, which also decayed to 0.8 times the previous value after every 15 epochs. The training process was conducted on an Nvidia GeForce GTX 3090Ti GPU card with 24GB of memory.

TABLE I: Summary of hyperparameters and network architectures.

Dataset	v_{th}	γ	(a^d, a^s)	Network	Architecture	Parameters(K)
S-MNIST	1.0	0.5	(0.7,0.8)	feedforward	64-256-256-10	85.1
	1.0	0.5	(0.8,0.9)	recurrent	64-256-256-10	155.1
PS-MNIST	1.0	0.5	(0.7,0.8)	feedforward	64-256-256-10	85.1
	1.8	1.0	(0.75,0.85)	recurrent	64-256-256-10	155.1
SHD	1.6	0.5	(0.65,0.75)	feedforward	700-128-128-20	108.8
	1.6	0.5	(0.75,0.85)	recurrent	700-128-128-20	141.8

B. Superior Sequential Modeling Capability

We conducted a comparative analysis of the proposed Adaptive TC-LIF model against both LIF and TC-LIF models using three widely-used sequential modeling datasets. The comparison encompassed results obtained through both BPTT (offline) and e-prop (online) training methodologies. As the results presented in Fig. 3, our proposed Adaptive TC-LIF model consistently outperforms the LIF neuron model in terms of classification accuracy, under the online learning setting. With the same number of parameters and employing the recurrent architecture, the Adaptive TC-LIF model achieved a remarkable classification accuracy of 98.61%, 98.54%, and 80.57% for the S-MNIST, PS-MNIST, and SHD datasets, respectively. These findings highlight the superior sequential modeling capacity of our proposed model.

Furthermore, the experimental results also demonstrate that our online training approach achieves competitive accuracies compared to the offline BPTT algorithm. Specifically, for the S-MNIST dataset, our model when trained online exhibits a minor accuracy gap of only -0.57% and -0.96% for the feedforward and feedback architectures, respectively. Similarly, for the PS-MNIST dataset, our online trained feedforward model shows an accuracy gap of -2.23% compared to BPTT, while the recurrent model even surpasses the accuracy of the vanilla TC-LIF model. The SHD dataset, comprising 250 time steps, poses a significant challenge for online learning algorithms. Notably, the online training results of our Adaptive TC-LIF model remain within 5% of the BPTT results. These outcomes indicate that our proposed Adaptive TC-LIF neuron model when coupled with the online learning method, strikes a desirable balance between classification accuracy and training efficiency.

C. *Ablation Studies*

1) Effectiveness of Learnable Decaying Constant: To emphasize the importance of adapting the membrane decaying constants α_1 and α_2 to suit the online learning scenario, we begin by conducting an ablation study on the vanilla TC-LIF neuron. Specifically, we initialize the hyperparameters β_1 and β_2 of TC-LIF neurons to -0.5 and 1, respectively. Furthermore, we conduct a grid search for hyperparameters α_1, α_2. The results depicted in Fig. 4 demonstrate that vanilla TC-LIF neurons, with $\alpha_1 = \alpha_2 = 1$, struggle to adapt effectively to the online learning setting. Conversely, the findings reveal that employing moderate decay constants leads to improved convergence. This enhancement can be attributed to a more balanced contribution between past and new inputs.

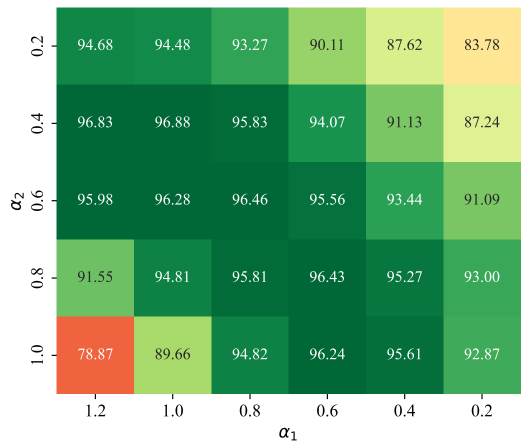


Fig. 4: The impact of hyperparameter setting on the test accuracy of the S-MNIST dataset. In this figure, the dark green color indicates high accuracy, while light red indicates low accuracy. The location where both α_1 and α_2 are equals to 1.0 corresponds to the vanilla TC-LIF model.

2) Effectiveness of Time-varying Decaying Constant: To further assess the effectiveness of incorporating time-varying decaying constants into Adaptive TC-LIF neurons, we conducted an ablation study on the SHD dataset. In this experiment, we focused solely on modifying the decaying constant $A[t]$, while keeping other hyperparameters constant. As the experimental results depicted in Fig. 5, the proposed Adaptive TC-LIF neuron with time-varying decaying constants exhibited a substantial increase of nearly 10% in accuracy compared to the best-performing vanilla TC-LIF model. This improvement underscores the crucial role played by time-varying decaying constants in enhancing the learning capabilities of TC-LIF neurons, particularly in the context of long-term sequential modeling tasks.

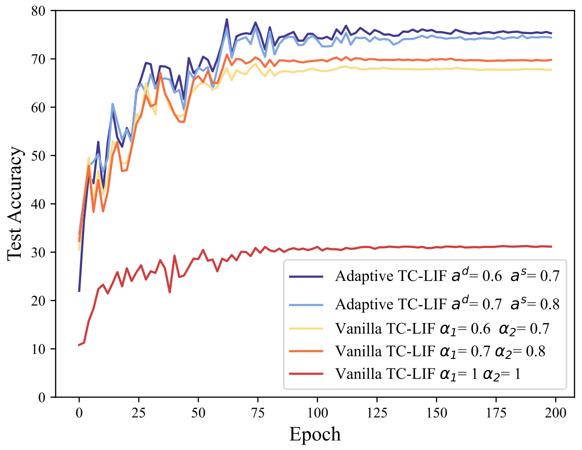


Fig. 5: Comparison of the performance with and without time-varying membrane decaying constants on the SHD dataset.

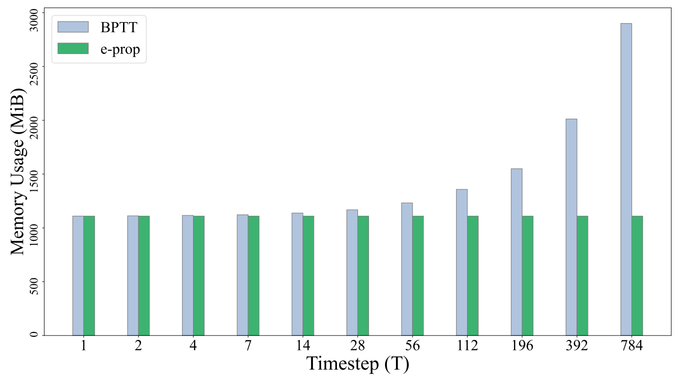


Fig. 6: Actual GPU memory usage with varying time sequence lengths on the S-MNIST dataset.

D. *Efficient Memory Consumption*

In theory, the memory cost of BPTT increases linearly with the time window size T, denoted as $\mathcal{O}(T)$. This linear relationship poses significant challenges when dealing with long sequence data. In contrast, the e-prop method enables online updates of model parameters, resulting in constant memory consumption. To evaluate the actual memory usage of our proposed Adaptive TC-LIF model during e-prop (online) and BPTT (offline) training, we conducted an empirical study using the S-MNIST dataset. We varied the sequence length from $T = 1$ to $T = 784$. The results, illustrated in Fig. 6, validate our theoretical analysis. While BPTT exhibits a linear increase in memory requirements, the e-prop method maintains a relatively stable memory usage regardless of the sequence length. This finding highlights the advantage of our proposed online training approach, especially in scenarios involving extended data sequences.

V. Conclusion

In this study, we investigated the feasibility of utilizing an online learning approach to train networks of multi-compartment spiking neurons. Our investigation began with the mathematical derivation of the e-prop algorithm tailored specifically for the TC-LIF neuron model. Additionally, we introduced the Adaptive TC-LIF neuron model by carefully analyzing the limitations of the original hyperparameter space used in the vanilla TC-LIF neuron for online training scenarios. Our approaches showcased comparable performance to the vanilla TC-LIF model achieved using the offline BPTT method, substantially surpassing the performance of LIF models trained with the e-prop algorithm. Furthermore, our proposed online training approach requires consistent memory usage, regardless of the sequence length. This highlights the superior memory efficiency of our approach. As a result, our study has laid a solid foundation towards enabling the training of high-performance multi-compartmental SNNs on emerging neuromorphic hardware.

References

[1] H. Markram, "The blue brain project," *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–160, 2006.

[2] Q. Yang, Q. Liu, and H. Li, "Deep residual spiking neural network for keyword spotting in low-resource settings," *Interspeech 2022*, pp. 3023–3027, 2022.

[3] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Frontiers in neuroscience*, vol. 12, p. 836, 2018.

[4] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[5] 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*, vol. 36, no. 6, pp. 51–63, 2019.

[6] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, 2018.

[7] J. Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, and K. C. Tan, "Progressive tandem learning for pattern recognition with deep spiking neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[8] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, and Z.-Q. Luo, "Training high-performance low-latency spiking neural networks by differentiation on spike representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 444–12 453.

[9] X. Chen, Q. Yang, J. Wu, H. Li, and K. C. Tan, "A hybrid neural coding approach for pattern recognition with spiking neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–15, 2023.

[10] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He *et al.*, "Towards artificial general intelligence with hybrid tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[11] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[12] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[13] R. Gütig, "Spiking neurons can discover predictive features by aggregate-label learning," *Science*, vol. 351, no. 6277, p. aab4113, 2016.

[14] X. Chen, J. Wu, H. Tang, Q. Ren, and K. C. Tan, "Unleashing the potential of spiking neural networks for sequential modeling with contextual embedding," 2023.

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

[16] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological cybernetics*, vol. 95, pp. 1–19, 2006.

[17] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.

[18] A. Rao, P. Plank, A. Wild, and W. Maass, "A long short-term memory for ai applications in spike-based neuromorphic hardware," *Nature Machine Intelligence*, vol. 4, no. 5, pp. 467–479, 2022.

[19] A. Shaban, S. S. Bezugam, and M. Suri, "An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation," *Nature Communications*, vol. 12, no. 1, p. 4234, 2021.

[20] M. Yao, H. Gao, G. Zhao, D. Wang, Y. Lin, Z. Yang, and G. Li, "Temporal-wise attention spiking neural networks for event streams classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 221–10 230.

[21] ——, "Temporal-wise attention spiking neural networks for event streams classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 221–10 230.

[22] L. Qin, Z. Wang, R. Yan, and H. Tang, "Attention-based deep spiking neural networks for temporal credit assignment problems," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2023.

[23] S. Zhang, Q. Yang, C. Ma, J. Wu, H. Li, and K. C. Tan, "Tc-lif: A two-compartment spiking neuron model for long-term sequential modelling," *arXiv preprint arXiv:2308.13250*, 2023.

[24] 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*, vol. 35, no. 12, 2021, pp. 11 062–11 070.

[25] Z. Wang, R. Jiang, S. Lian, R. Yan, and H. Tang, "Adaptive smoothing gradient learning for spiking neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 35 798–35 816.

[26] 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*, vol. 35, pp. 20 717–20 730, 2022.

[27] 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*, vol. 11, no. 1, p. 3625, 2020.

[28] Q. Yu, C. Ma, S. Song, G. Zhang, J. Dang, and K. C. Tan, "Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1714–1726, 2022.

[29] L. Manneschi and E. Vasilaki, "An alternative to backpropagation through time," *Nature Machine Intelligence*, vol. 2, no. 3, pp. 155–156, 2020.

[30] C. Frenkel and G. Indiveri, "Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 1–3.

[31] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[32] 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*, 2021, pp. 2661–2671.

[33] X. Yao, F. Li, Z. Mo, and J. Cheng, "Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 160–32 171, 2022.

[34] B. Yin, F. Corradi, and S. M. Bohté, "Effective and efficient computation with multiple-timescale spiking recurrent neural networks," in *International Conference on Neuromorphic Systems 2020*, 2020, pp. 1–8.

[35] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[36] 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.

[37] A. Bernacchia, H. Seo, D. Lee, and X.-J. Wang, “A reservoir of time constants for memory traces in cortical neurons,” *Nature neuroscience*, vol. 14, no. 3, pp. 366–372, 2011.

[38] C. Ma, R. Yan, Z. Yu, and Q. Yu, “Deep spike learning with local classifiers,” *IEEE Transactions on Cybernetics*, 2022.

[39] Q. V. Le, N. Jaitly, and G. E. Hinton, “A simple way to initialize recurrent networks of rectified linear units,” *arXiv preprint arXiv:1504.00941*, 2015.

[40] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, “The heidelberg spiking data sets for the systematic evaluation of spiking neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2020.

[41] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[42] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.