Advancing Training Efficiency of Deep Spiking Neural Networks through Rate-based Backpropagation

Chengting Yu[1,2], **Lei Liu**[2], **Gaoang Wang**[2], **Erping Li**[1,2], **Aili Wang**[1,2*]
[1] College of Information Science and Electronic Engineering, Zhejiang University
[2] ZJU-UIUC Institute, Zhejiang University
chengting.21@intl.zju.edu.cn, ailiwang@intl.zju.edu.cn

Abstract

Recent insights have revealed that rate-coding is a primary form of information representation captured by surrogate-gradient-based Backpropagation Through Time (BPTT) in training deep Spiking Neural Networks (SNNs). Motivated by these findings, we propose rate-based backpropagation, a training strategy specifically designed to exploit rate-based representations to reduce the complexity of BPTT. Our method minimizes reliance on detailed temporal derivatives by focusing on averaged dynamics, streamlining the computational graph to reduce memory and computational demands of SNNs training. We substantiate the rationality of the gradient approximation between BPTT and the proposed method through both theoretical analysis and empirical observations. Comprehensive experiments on CIFAR-10, CIFAR-100, ImageNet, and CIFAR10-DVS validate that our method achieves comparable performance to BPTT counterparts, and surpasses state-of-the-art efficient training techniques. By leveraging the inherent benefits of rate-coding, this work sets the stage for more scalable and efficient SNNs training within resource-constrained environments. Our code is available at https://github.com/Tab-ct/rate-based-backpropagation.

1 Introduction

Spiking Neural Networks (SNNs) are conceptualized as biologically inspired neural systems, incorporating spiking neurons that closely mimic biological neural dynamics [46, 56]. Unlike Artificial Neural Networks (ANNs) based on continuous data representations, SNNs adopt spike-coding strategies to facilitate data transmission through discrete binary spike trains [52]. The intrinsic binary mechanism eliminates the need for the extensive multiply-accumulate operations typically required for synaptic connectivity [56], thereby enhancing energy efficiency and inference speed when deployed on neuromorphic hardware systems [1, 10, 54].

The mainstream training methods for SNNs primarily utilize Backpropagation Through Time (BPTT) with surrogate gradients to overcome non-differentiable spike events, allowing SNNs to achieve comparable results with ANNs counterparts [51, 62, 74]. However, the direct training method necessitates the storage of all temporal activations for backward propagation across the network's depth and duration, leading to high training costs in terms of both computational time and memory demands [43, 86, 35, 77, 76, 47, 13]. To alleviate memory burdens, online training techniques have been developed that partially decouple the time dependencies of backward computations in BPTT [2, 4, 76, 48, 89]. However, online methods still require iterative computations based on the time dimension, increasing training time complexity as the number of timesteps grows.

*Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Observed across most biological sensory systems, rate coding is a phenomenon where information is encoded through the rate of neuronal spikes, regardless of precise spike timing [52, 64, 23]. Recent explorations into spike representation have demonstrated the significant role of rate coding in enhancing the robustness of SNNs, further confirming its dominant position as the encoding representation in networks [38, 60, 18]. A significant observation has shown that BPTT-trained SNNs on static benchmark exhibit spike representation primarily following the rate-coding manner by highlighting strong similarities in representation between SNNs and their ANN counterparts [44]. A similar conclusion resonated with findings in fields of adversarial attacks, where recent methods significantly benefit from rate-based representations to enhance attack effectiveness [6, 29, 50].

Motivated by rate coding's status as the most effective and predominant form of representation in SNNs, we posit that targeted training based on rate-based information could offer a high cost-effectiveness ratio. We propose to decouple BPTT based on rate-coding approximation and simplify rate-based derivative computations to a single spatial backpropagation. We further provide theoretical analysis and empirical evidence to reveal the rationality of the gradient approximation between BPTT and the proposed method. Experimental results demonstrate that the proposed method achieves performance comparable to BPTT counterparts while significantly reducing memory and computational demands. Comparison results also indicate that the proposed method outperforms state-of-the-art efficient training methods on benchmarks. We expect our work to facilitate more efficient and scalable training for SNNs in resource-constrained environments. Our main contributions are as follows:

- We propose rate-based backpropagation that leverages rate-coded information for efficient training of deep SNNs. This method simplifies the computational graph by decoupling and compressing temporal dependencies, reducing training time and memory requirements.
- Alongside the proposed method, we conduct theoretical analysis and empirical validation to demonstrate its effectiveness in approximating the gradient computations performed by BPTT-based SNNs training.
- We conduct experiments on CIFAR-10, CIFAR-100, CIFAR10-DVS, and ImageNet, demonstrating that our proposed method matches the comparable performance of the BPTT counterpart and achieves state-of-the-art results among efficient SNN training methods.

2 Related Work

Training Methods for Deep SNNs. Deep SNNs are trained primarily through two principal strategies: (1) conversion methods that establish links between SNNs and ANNs through equivalent closed-form mappings, and (2) direct training from scratch utilizing Backpropagation Through Time (BPTT). Conversion methods develop closed-form formulations for spike representations [39, 67, 72, 88, 73, 47], enabling seamless transitions of pre-trained ANNs into SNNs and facilitating comparable performance on large-scale datasets [8, 15, 27, 59, 57, 12, 42, 16]. However, the precision of these mappings under ultra-low latency conditions is not consistently reliable, often necessitating extensive time steps to accumulate spikes, which may compromise performance [7, 40, 31, 28, 34]. Direct training methods permit SNNs' performance with extremely low time steps by employing BPTT along with surrogate gradients to compute derivatives of discrete spiking events [51, 62, 74, 22, 81, 87, 85, 43, 66, 69, 13]. The strategy fosters innovation in SNN-specific modules, including optimized neurons, synapses, and network architectures, thereby enhancing performance [25, 21, 20, 17, 79, 83, 24, 80, 61]. Despite the advantages of low latency, direct training imposes substantial memory and time burdens to maintain the backward computational graph [43, 86, 35, 77, 76, 47, 13]. To mitigate training costs associated with direct methods, light training strategies have attracted considerable attention [49, 35, 86, 55, 70]. Several studies have explored the concept of decoupling the forward and backward passes in SNNs, which generally assumes that neuronal dynamics follow deterministic processes and aims to establish closed-form fixed-point equivalences between spike representations and corresponding rate-based activations [72, 73, 77, 47, 68]. Drawing on online training techniques from recurrent neural networks, several studies have adapted the principles of Real-time Recurrent Learning (RTRL) [71] to streamline the online training process for SNNs, aiming to decrease memory demands while preserving biologically plausible online properties of the networks [84, 2, 4, 82, 55, 76, 47, 89]. The online methodologies have proven effective in large-scale tasks [76, 47, 89]. Nevertheless, the significant time costs associated with training methods continue to challenge SNNs' broader application.

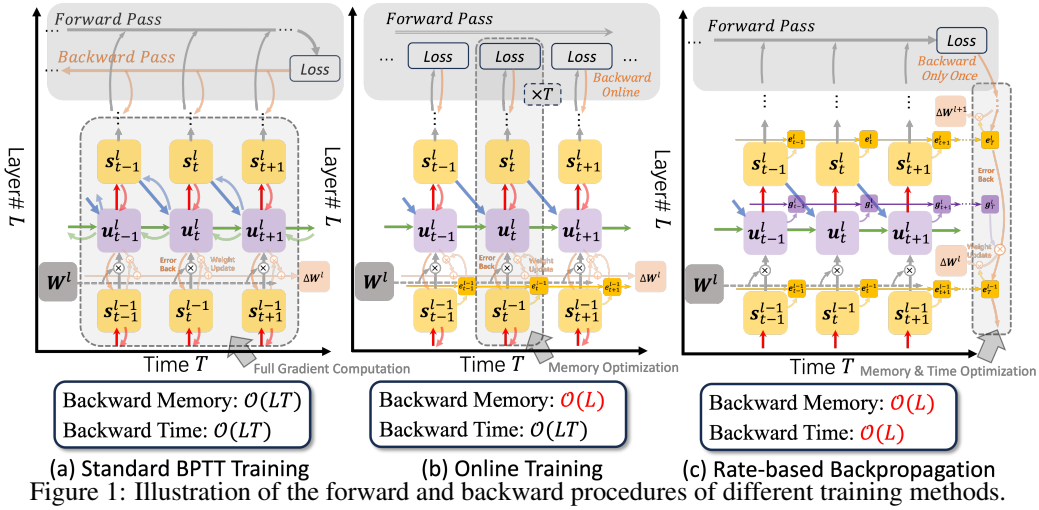


Figure 1: Illustration of the forward and backward procedures of different training methods.

Spike Coding in SNNs. SNNs transmit information through spike trains [52], with encoding mechanisms classified into temporal and rate coding. Temporal coding is defined on firing times, employed by several direct trainings [49, 75, 88] and ANN-to-SNN conversions [26, 65], is noted for its low energy consumption due to sparse spiking. However, temporal coding schemes often require specialized neuron configurations and are generally effective only on simpler datasets [26, 65, 88]. Conversely, rate coding is widely adopted across both conversion [12, 15, 16, 27, 36, 57, 59, 78] and direct training approaches [73, 77, 47], consistently achieving superior performance and facilitating low-latency operations [77, 47]. Moreover, rate coding has demonstrated significant potential in enhancing the robustness of SNNs against adversarial attacks [38, 60, 18], with attack methods specifically designed to exploit rate-based representations showing promise in surpassing benchmarks for SNNs defense against attacks [6, 30, 50]. By employing representation similarity analysis to compare BPTT-trained SNNs with their ANN counterparts, Li et al. [44] has indicated that rate coding serves as the primary mode of information representation [44]. Inspired by previous findings, we consider that rate-coded information represents the most effective and predominant form of signal expression in SNNs, and the targeted training based on rate-based spike representations may offer a high cost-effectiveness ratio. Therefore, we propose to decouple BPTT towards rate-based backpropagation with the purpose of enhancing the efficiency of SNNs training.

3 Preliminaries

3.1 Spiking Neural Networks

Inspired by the brain's ability to transmit information through discrete spikes, the Leaky Integrate-and-Fire (LIF) model serves as the basic building block of SNNs due to its simplicity. For practical implementation of SNNs based on connected spiking neurons, the dynamics of the LIF model are typically rendered in a discrete iterative format:

$$\boldsymbol{u}_t^l = \lambda(\boldsymbol{u}_{t-1}^l - V_{\text{th}}\boldsymbol{s}_{t-1}^l) + \boldsymbol{W}^l\boldsymbol{s}_t^{l-1}, \quad \boldsymbol{s}_t^l = H(\boldsymbol{u}_t^l - V_{\text{th}}) \tag{1}$$

where $\boldsymbol{u}_t^l$ and $\boldsymbol{s}_t^l$ represent the membrane potential and output spike of neurons in layer l at time t, respectively. $\boldsymbol{W}^l$ denotes the linear synaptic connections between layers $l-1$ and l, and λ acts as the decay term for the membrane potential. The Heaviside step function, $H(\cdot)$, determines spike generation, ensuring $\boldsymbol{s}_t^l$ in binary forms. Noting that $H(\cdot)$ is not differentiable, SNNs' direct training employs surrogate gradients to achieve error propagation by creating various pseudo-derivatives [51, 74, 19], following the basic idea of Straight-Through Estimator (STE) [3].

3.2 Training SNNs with BPTT

The network outputs at each timestep t are given by $\boldsymbol{o}_t = \boldsymbol{W}^L\boldsymbol{s}_t^L$, where $\boldsymbol{W}^L$ denotes the classifier's weights. Classification is based on the average of these outputs across all timesteps, computed as $\boldsymbol{y}_{\text{pred}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{o}_t$. The loss function $\mathcal{L}$ is defined over averaged outputs and is typically formulated as $\mathcal{L} = \ell\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{o}_t, \boldsymbol{y}\right)$, where $\boldsymbol{y}$ represents the true labels and ℓ could be the cross-entropy function, as noted in various studies [87, 48, 19, 69]. BPTT unfolds the iterations described in Eq. (1), and propagates gradients back along the computational graphs across both temporal and spatial dimensions, as illustrated in Fig. 1a. The gradients of the membrane potential $\boldsymbol{u}$ incorporate elements

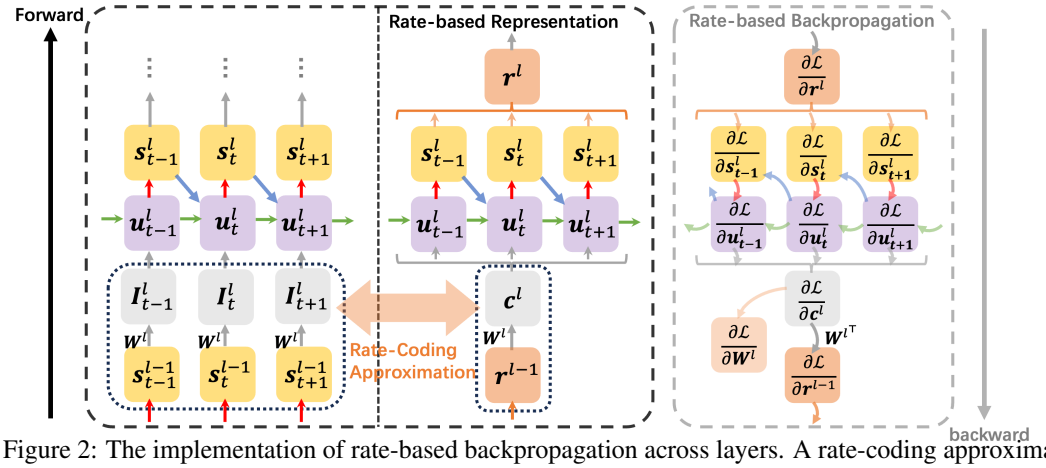


Figure 2: The implementation of rate-based backpropagation across layers. A rate-coding approximation is utilized for the forward procedure to connect average inputs with rate outputs, enabling fast rate-based error backpropagation throughout the training process.

from both (spatial) spike generation and (temporal) potential accumulation, expressed as:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_t^l} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_{t+1}^l}\left(\frac{\partial \boldsymbol{u}_{t+1}^l}{\partial \boldsymbol{u}_t^l} + \frac{\partial \boldsymbol{u}_{t+1}^l}{\partial \boldsymbol{s}_t^l}\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l}\right) \\ &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} + \sum_{\tau>t}\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_\tau^l}\frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{u}_\tau^l}\prod_{i=\tau-1}^{t}\left(\frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{u}_i^l} + \frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{s}_i^l}\frac{\partial \boldsymbol{s}_i^l}{\partial \boldsymbol{u}_i^l}\right)\end{aligned} \tag{2}$$

Subsequently, the weight update for layer l is determined among all timesteps T, i.e. $\nabla_{\boldsymbol{W}^l}\mathcal{L} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_t^l}\frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{W}^l} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_t^l}{\boldsymbol{s}_t^{l-1}}^\top$, and the gradient is further propagated to previous layers through the linear part by $\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^{l-1}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_t^l}{\boldsymbol{W}^l}^\top$.

4 Rate-based Backpropagation for SNNs Training

4.1 Derivation of Rate-based Backpropagation

Incorporating rate-based representation. Under the rate coding assumption, essential information is effectively encapsulated within the spike frequency averages. We start by defining the rate-based representation as an approximation for the forward procedure in SNNs, as shown in Figure 2. The average firing rate at each layer l, denoted as $\boldsymbol{r}^l$, is calculated as the expected value of the spike outputs $\boldsymbol{s}_t^l$ over the temporal dimension $\boldsymbol{r}^l = \mathbb{E}[\boldsymbol{s}_t^l] = \frac{1}{T}\sum_{t\leq T}\boldsymbol{s}_t^l$.

Considering the forward propagation through linear operators with weights $\boldsymbol{W}^l$ that compute the inputs as $\boldsymbol{I}_t^l = \boldsymbol{W}^l\boldsymbol{s}_t^{l-1}$, instead of transmitting distinct spikes over multiple timesteps, we transform the average rates into average inputs $\boldsymbol{c}^l$ in the approximate representation:

$$\boldsymbol{c}^l = \mathbb{E}[\boldsymbol{I}_t^l] = \mathbb{E}[\boldsymbol{W}^l\boldsymbol{s}_t^{l-1}] = \boldsymbol{W}^l\mathbb{E}[\boldsymbol{s}_t^{l-1}] = \boldsymbol{W}^l\boldsymbol{r}^{l-1}.$$

Supposing input representations are well captured within $\boldsymbol{c}^l$, we approximate the exact inputs with the average inputs for all timesteps, $\boldsymbol{I}_t^l \approx \boldsymbol{c}^l$, and follow the neuronal dynamics in Eq. (1) to derive the output rates $\boldsymbol{r}^l = \mathbb{E}[\boldsymbol{s}_t^l]$. With the rate-coding approximation in place, we can derive the gradients with respect to the weights in the linear part based on the error propagated through the average inputs:

$$\left(\nabla_{\boldsymbol{W}^l}\mathcal{L}\right)_{\text{rate}} \equiv \frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\frac{\partial \boldsymbol{c}^l}{\partial \boldsymbol{W}^l} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}{\boldsymbol{r}^{l-1}}^\top \tag{3}$$

Handling temporal dependency during backward. For back-propagating the error, the linear parts operate smoothly as $\frac{\partial \boldsymbol{c}^l}{\partial \boldsymbol{r}^{l-1}} = {\boldsymbol{W}^l}^\top$. The next step is to define the correlation between the averages of inputs and output spike rates, $\frac{\partial \boldsymbol{r}^l}{\partial \boldsymbol{c}^l}$, within the neurons of layer l. Since there is no deterministic relationship between r^l and c^l, we first look into the influence of separated inputs following the exact

gradients in Eq. (2):

$$\frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{I}_t^l} = \frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{u}_t^l} = \begin{cases} \frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{u}_\tau^l} \prod_{i=\tau-1}^{t} \left(\frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{u}_i^l} + \frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{s}_i^l} \frac{\partial \boldsymbol{s}_i^l}{\partial \boldsymbol{u}_i^l} \right) & \text{if } \tau \geq t \\ \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} & \text{if } \tau = t \\ 0 & \text{if } \tau < t \end{cases} \tag{4}$$

By accumulating the intricate dynamics over time, we can derive the gradients of the overall spikes with respect to the inputs at time t:

$$\boldsymbol{\varkappa}_t^l = \sum_\tau \frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{I}_t^l} = \left(\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} + \sum_{\tau > t} \frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{u}_\tau^l} \prod_{i=\tau-1}^{t} \left(\frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{u}_i^l} + \frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{s}_i^l} \frac{\partial \boldsymbol{s}_i^l}{\partial \boldsymbol{u}_i^l} \right) \right) \tag{5}$$

Here, with rate-coding approximating $\boldsymbol{I}_t^l \approx \boldsymbol{c}^l$, we follow the idea of Straight-Through Estimator [3] and define the backward gradients as $\frac{\partial \boldsymbol{I}_t^l}{\partial \boldsymbol{c}^l} = Id$, with Id representing the identity matrix. Then, we can derive the surrogate gradients of neural dynamics through the mean estimator:

$$\left(\frac{\partial \boldsymbol{r}^l}{\partial \boldsymbol{c}^l}\right)_{\text{rate}} \equiv \sum_\tau \left(\frac{\partial (\mathbb{E}\left[\boldsymbol{s}_t^l\right])}{\partial \boldsymbol{I}_\tau^l} \frac{\partial \boldsymbol{I}_\tau^l}{\partial \boldsymbol{c}^l} \right) = \frac{1}{T} \sum_t \sum_\tau \left(\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_\tau^l} \right) = \mathbb{E}\left[\boldsymbol{\varkappa}_t^l\right] \tag{6}$$

With the compressed gradients of neuron parts, the error backpropagation of the rate-based representation is then determined, dependent only on the spatial domain:

$$\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\right)_{\text{rate}} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^L} \prod_{i=L-1}^{l} \left(\frac{\partial \boldsymbol{c}^{i+1}}{\partial \boldsymbol{r}^i} \left(\frac{\partial \boldsymbol{r}^i}{\partial \boldsymbol{c}^i}\right)_{\text{rate}} \right) \right) = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^L} \prod_{i=L-1}^{l} \left(\boldsymbol{W}^{i\top} \mathbb{E}\left[\boldsymbol{\varkappa}_t^l\right] \right) \right) \tag{7}$$

where we define the objective $\mathcal{L} = \frac{1}{T}\ell(\mathbb{E}[\boldsymbol{o}_t], y) = \frac{1}{T}\ell(\boldsymbol{c}^L, \boldsymbol{y})$. Note that the rate-based representation, while instrumental in constructing the backward computational graph for learning, does not necessitate actual implementation during the forward pass.

4.2 Rate-based Gradient Computation for Memory and Time Efficiency

As previously discussed, rate-based backpropagation can be executed on spatial-dimension computation by decoupling BPTT. We now show how rate-based backpropagation can be efficiently implemented within the overall learning framework. As depicted in Figure 2b, online schemes apply eligibility traces $\boldsymbol{e}_t^l$ locally within neurons to store historical information, effectively blocking backward access to past gradients. The gradient computation is optimized by compressing all past temporal dependencies into $\boldsymbol{e}_t^l$. Similarly, we utilize iterative variables $\{\boldsymbol{g}_t^l\}_{l \leq L}$ and $\{\boldsymbol{e}_t^l\}_{l \leq L}$ as the accumulated post- and pre-synaptic dependencies, synchronously recorded during the neural dynamics computations. The iteration of $\{\boldsymbol{e}_t^l\}_{l \leq L}$ dynamically records the firing rates, where $\boldsymbol{e}_t^l = \frac{1}{t}((t-1)\boldsymbol{e}_{t-1}^l + \boldsymbol{s}_t^l)$, and it is straightforward to derive $\boldsymbol{r}^l = \boldsymbol{e}_T^l$. Considering the surrogate gradients of neural dynamics, $\frac{\partial \boldsymbol{r}^l}{\partial \boldsymbol{c}^l}$, to estimate future-dependent terms outlined in Eq. 5, we first construct equivalent eligibility trace forms, $\{\boldsymbol{\rho}_t^l\}_{t \leq T}$, with iterative expressions starting at $\boldsymbol{\rho}_1^l = 1$:

$$\boldsymbol{\rho}_t^l = 1 + \boldsymbol{\rho}_{t-1}^l \left(\frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{u}_{t-1}^l} + \frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{s}_{t-1}^l} \frac{\partial \boldsymbol{s}_{t-1}^l}{\partial \boldsymbol{u}_{t-1}^l} \right) = 1 + \sum_{\tau < t} \prod_{i=t-1}^{\tau} \left(\frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{u}_i^l} + \frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{s}_i^l} \frac{\partial \boldsymbol{s}_i^l}{\partial \boldsymbol{u}_i^l} \right) \tag{8}$$

with the equivalence that:

$$\begin{aligned} \sum_t \boldsymbol{\varkappa}_t^l &= \sum_t \left(\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} + \sum_{\tau > t} \left(\frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{u}_\tau^l} \prod_{i=\tau-1}^{t} \left(\frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{u}_i^l} + \frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{s}_i^l} \frac{\partial \boldsymbol{s}_i^l}{\partial \boldsymbol{u}_i^l} \right) \right) \right) \\ &= \sum_t \left(\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} \left(1 + \sum_{\tau < t} \prod_{i=t-1}^{\tau} \left(\frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{u}_i^l} + \frac{\partial \boldsymbol{u}_{i+1}^l}{\partial \boldsymbol{s}_i^l} \frac{\partial \boldsymbol{s}_i^l}{\partial \boldsymbol{u}_i^l} \right) \right) \right) = \sum_t \left(\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l} \boldsymbol{\rho}_t^l \right) \end{aligned} \tag{9}$$

By iteratively accumulating $\boldsymbol{g}_t^l = \frac{1}{t}((t-1)\boldsymbol{g}_{t-1}^l + \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l}\boldsymbol{\rho}_t)$, we obtain $\boldsymbol{g}_T^l = \mathbb{E}[\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l}\boldsymbol{\rho}_t^l] = \mathbb{E}[\boldsymbol{\varkappa}_t^l]$. Now, we have collapsed the required computation graph through the iterative calculation to complexity $\mathcal{O}(L)$. The rate-based propagation is then conducted in one go, relying only on the intermediate variables $\boldsymbol{e}_T^l$, $\boldsymbol{g}_T^l$, and $\boldsymbol{W}^l$, within one-time spatial-dimension backpropagation.

4.3 Connecting Error Backward of Rate-based Backpropagation to BPTT

Having derived the fundamental form of rate-based backpropagation through the rate-encoding approximation, we now explore potential divergences with BPTT during error propagation. Although rate-based backpropagation is derived from the approximated forward pass, it still provides valid gradients for the original network parameters.

The primary divergence between rate-back and BPTT in backward computation primarily arises from the assumptions regarding the approximation of rate-based representation through mean estimators, as outlined in Eq.(3) and Eq.(6). The rate-coding motivations establish equivalence with BPTT by assuming temporal components are independent, which is formalized in Theorem 1.

Theorem 1. *Given* $\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}$ *that refers to gradients computed following the chain rule of BPTT in Eq. (2), and* $\boldsymbol{\kappa}_t^l = \sum_\tau \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_\tau^l}$ *(where* $\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right] = \mathbb{E}\left[\boldsymbol{\varkappa}_t^l\right]$ *in Eq.(6-7)) , if* $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]$ *holds for* $\forall l$*, we have* $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right] = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^l}\right)_{rate}$*. Furthermore, given* $\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}$*, if* $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\right]\mathbb{E}[\boldsymbol{s}_t^{l-1}]$ *for* $\forall l$*, we then obtain* $(\nabla_{\boldsymbol{W}^l}\mathcal{L})_{rate} = \frac{1}{T}(\nabla_{\boldsymbol{W}^l}\mathcal{L})$*. Here,* $\mathbb{E}[\boldsymbol{x}_t] = \frac{1}{T}\sum_t \boldsymbol{x}_t$ *refers the mean value of tensor* $\boldsymbol{x}_t$ *over temporal dimension* T*.*

To confirm our hypotheses, we carried out empirical experiments, the results of which are detailed in the experimental section. Our empirical findings support the core assumptions outlined in Theorem 1, demonstrating the relative independence between $\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}$ and $\boldsymbol{\kappa}_t^l$ (Figure 3a,b), as well as between $\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}$ and $\boldsymbol{s}_t^l$ (Figure 3c). For minor discrepancies that may arise, we introduced Theorem 2, which tolerates small deviations and confirms that approximation errors in rate-based backpropagation can be effectively bounded, ensuring the robustness of training under practical conditions.

Theorem 2. *For gradients* $\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}$ *and* $\boldsymbol{\kappa}_t^l = \sum_\tau \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_\tau^l}$*, given the approximation error bound* $\epsilon > 0$ *s.t.* $\left\| \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right] - \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right] \right\| \leq \epsilon(1 + \left\| \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right] \right\|)$ *for* $\forall l$*. Denote the stacked tensor* $\boldsymbol{I}^l = [\boldsymbol{I}_1^l, ..., \boldsymbol{I}_T^l]$ *and* $\boldsymbol{s}^l = [\boldsymbol{s}_1^l, ..., \boldsymbol{s}_T^l]$*. Assuming the backward procedure follows non-expansivity s.t.* $\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l} = \boldsymbol{W}^{l+1^\top}\frac{\partial \boldsymbol{s}^l}{\partial \boldsymbol{I}^l}$ *is 1-lipschitz continuous without loss of generality and the biases are bounded uniformly by B, i.e.* $\left\| \boldsymbol{x}\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l} - \hat{\boldsymbol{x}}\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l} \right\| \leq \left\| \boldsymbol{x} - \hat{\boldsymbol{x}} \right\|$ *for* $\forall \boldsymbol{x}, \hat{\boldsymbol{x}}$*. Define* $\boldsymbol{\delta}_{rate}^l = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\right)_{rate}$ *as the error propagated through Eq. (7), and* $\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}$ *as the error propagated through BPTT, with* $\boldsymbol{\delta}_{rate}^L = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^L)}]$*. We have the gradient difference bounded by* $\left\| \boldsymbol{\delta}_{rate}^{L-k} - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{L-k})}] \right\| = \mathcal{O}(k^2\epsilon)$*.*

Theorem 2 elucidates the stability of rate-based backpropagation relative to BPTT, showing that the proposed method can provide a bound on the overall objective solution. The bounded error could further be interpreted as a form of randomness suitable for stochastic optimization. The similarity measurement of the descent directions between the two methods provides empirical evidence for the effectiveness of the proposed method (Figure 3d). Detailed proof is provided in Appendix A.

4.4 Implementation Details

For implementations of direct training, two distinct training modes are recognized: (multi-step) activation-based and (single-step) time-based [19], differing fundamentally in handling the timesteps loop. We implement our rate-based propagation in both modes: $\textbf{rate}_M$ denotes the multi-step training mode where T loops are embedded within layers, and $\textbf{rate}_S$ refers to the single-step training mode with T loops outside the layers. A detailed discussion of training modes is included in Appendix B.

Another aspect of our implementation concerns handling batch normalization (BN), especially given its critical role in BPTT, which adjusts mean and variance statistics during the forward pass. The application of BN varies depending on the training mode. In the multi-step mode, BN benefits from access to information across all timesteps and can normalize based on statistics aggregated over temporal dimensions. We employed tdBN [87] in $\textbf{rate}_M$ since it has been widely adopted in direct training on various benchmarks. In contrast, the single-step mode limits BN to current timestep inputs, necessitating normalization across spatial dimensions only. In line with online schemes, SLTT [48] demonstrates the feasibility of implementing spatial BN iteratively across timesteps, an approach we adopt for $\textbf{rate}_S$. Further details on the BN implementation are provided in Appendix B.

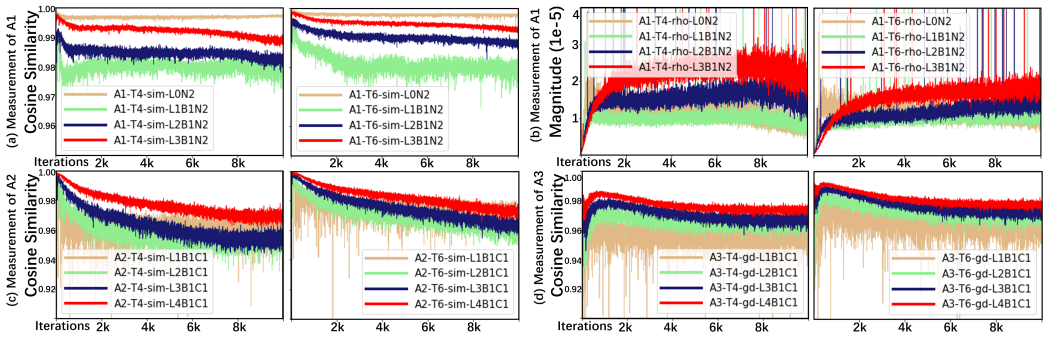


Figure 3: Empirical measurements conducted on the training procedure of BPTT. The experiments are carried out on the CIFAR-100 dataset using ResNet-18. Each subplot is labeled according to the naming convention "A{test#}-T{timesteps#}-{target}-L{layer#}B{block#}N{LIF#}/C{conv#}."

5 Experiments

In this section, we conduct experiments on CIFAR-10 [37], CIFAR-100 [37], ImageNet [11], and CIFAR10-DVS [41] to evaluate the proposed training method. We implement SNNs training on the Pytorch [53] and SpikingJelly [19] frameworks. We set $V_{th} = 1$, $\lambda = 0.2$, and employ the sigmoid-based surrogate function [19] for LIF neurons. Detailed setups are provided in Appendix C.

5.1 Empirical Validation

Empirical experiments are conducted to support the preconditions of theorems discussed in Section 4.3. These preconditions assert the independence of paired variables across the temporal dimension: $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]$ (A1) and $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\right]\mathbb{E}[\boldsymbol{s}_t^{l-1}]$ (A2). To explore these relationships, we conducted experiments training ResNet-18 on CIFAR-100 using BPTT. Cosine similarity measures were employed to compare the empirical expectation products, $cos\langle\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right], \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]\rangle$ as shown in Figure 3a, where values approaching 1 indicate a high degree of alignment, suggesting that the variables' directions are similar. Additionally, the correlation coefficient, ρ was measured to further assess the independence of these variables $\rho = \frac{\mathcal{COV}(\boldsymbol{\kappa}_t, \boldsymbol{\delta}_t^{(\boldsymbol{s}^l)})}{\sqrt{\text{var}(\boldsymbol{\kappa}_t)\text{var}(\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)})}}$ where $\mathcal{COV}(\boldsymbol{\kappa}_t, \boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}) = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right] - \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]$ It is clear that ρ equals the cosine distance between the variables after centering by their means, $\rho = cos\langle\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)} - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}], \boldsymbol{\kappa}_t^l - \mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]\rangle$. Results, shown in Figure 3b, reveal that the correlation coefficients are constrained within a very small range, typically around the magnitude of $\sim 10^{-5}$, supporting the hypothesis of their relative independence. We also conducted cosine similarity measurements to validate the assumption $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\right]\mathbb{E}[\boldsymbol{s}_t^{l-1}]$, as shown in Figure 3c. Additionally, we implement both BPTT and the proposed method simultaneously within the same training iteration, allowing direct observation of the gradient descent directions. The relation $(\nabla_{\boldsymbol{W}^l}\mathcal{L})_{\text{rate}} = \frac{1}{T}(\nabla_{\boldsymbol{W}^l}\mathcal{L})$ (A3) was visualized in Figure 3d, which revealed that the convergence directions for rate-based backpropagation and BPTT are closely aligned. Remarkably, all tests consistently demonstrate that configurations with T=6 better adhere to the theoretical assumptions than T=4, suggesting that the proposed method can more closely mimic BPTT computations as the timestep increases. This observation also highlights the intrinsic link between our method and rate-coding, suggesting that a larger temporal window may facilitate more stable manifestations of rate-coding.

5.2 Comparison with the State-of-the-Art

We present comparison results in Table 1. In single-step mode, **rate**$_S$ offers fair comparisons with online schemes, while **rate**$_M$ in multi-step mode competes fairly with other methods employing one-step backpropagation. Unlike online methods such as OTTT[76], SLTT[48], and OS[89], which necessitate spatial backpropagation at every timestep, our proposed method conducts this process only once at the final timestep. Although methods of DSR [47] and SSF[68] delay decoupled backpropagation until the final timestep, allowing for parallel processing across all timesteps to enhance computational speed, they still require each timestep's backpropagation to be managed independently within the backward computation graph. In contrast, our method fully compresses the temporal dimension, achieving one-step time-independent spatial backpropagation. As shown in Table 1, our method yields comparable performance with BPTT counterparts on benchmarks, showcasing promising capabilities compared to other efficient training methods. While our theoretical

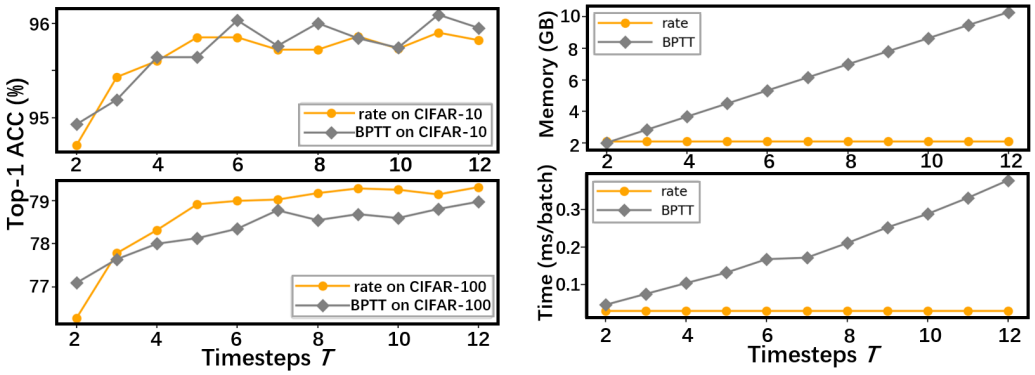
Table 1: Performance on CIFAR-10, CIFAR-100, ImageNet, and CIFAR10-DVS. Results are averaged over three runs of experiments, except for single crop evaluations on ImageNet. Models marked with (*) employ scaled weight standardization, adapting to normalizer-free architectures.

	Training	Method	Model	Timesteps	Top-1 Acc (%)
CIFAR10	QCFS [7]	ANN2SNN	ResNet-18	8	94.82
	DSR [47]	one-step	PreAct-ResNet-18	20	95.10±0.15
	SSF [68]	one-step	PreAct-ResNet-18	20	94.90
	BPTT$_M$	BPTT	ResNet-18	4	95.64
	rate$_M$ (ours)	one-step	ResNet-18	4	95.61±0.02(95.64)
	OTTT [76]	online	VGG-11*	6	93.52±0.06
	SLTT [48]	online	ResNet-18	6	94.44±0.21
	OS [89]	online	VGG-11 ResNet-19	4 4	94.35 95.20
	BPTT$_S$	BPTT	ResNet-18	4	95.53
			VGG-11	4	95.61
	rate$_S$ (ours)	one-step	ResNet-18	4	95.42±0.11(95.56)
			VGG-11	4	95.57±0.08(95.68)
CIFAR100	DSR [47]	one-step	PreAct-ResNet-18	20	78.50±0.12
	SSF [68]	one-step	PreAct-ResNet-18	20	75.48
	BPTT$_M$	BPTT	ResNet-18	4	77.93
	rate$_M$ (ours)	one-step	ResNet-18	4	78.26±0.12(78.38)
	OTTT [76]	online	VGG-11*	6	71.05±0.04
	SLTT [48]	online	ResNet-18	6	74.38±0.30
	OS [89]	online	VGG-11 ResNet-19	4 4	76.48 77.86
	BPTT$_S$	BPTT	ResNet-18	4	77.72
			VGG-11	4	77.82
	rate$_S$ (ours)	one-step	ResNet-18	4	77.73±0.28(77.93)
			VGG-11	4	77.87±0.35(78.13)
ImageNet	OTTT [76]	online	PreAct-ResNet-34*	6	65.15
	SLTT [48]	online	PreAct-ResNet-34*	6	66.19
	OS [89]	online	SEW-ResNet-34 PreAct-ResNet-34	4 4	64.14 67.54
	SEW-ResNet [20]	BPTT	SEW-ResNet-34	4	67.04
	rate$_S$ (ours)	one-step	SEW-ResNet-34 PreAct-ResNet-34	4 4	65.66 69.58
	rate$_M$ (ours)	one-step	SEW-ResNet-34 PreAct-ResNet-34	4 4	65.84 70.01
CIFAR10-DVS	DSR [47] SSF [68]	one-step	VGG-11 VGG-11	20 20	77.27±0.24 78.0
	OTTT [76] SLTT [48]	online	VGG-11* VGG-11	10 10	76.63±0.34 77.17±0.23
	BPTT$_S$ BPTT$_M$	BPTT	VGG-11 VGG-11	10 10	76.73 76.86
	rate$_S$ (ours) **rate$_M$ (ours)**	one-step	VGG-11 VGG-11	10 10	76.48±0.23(76.71) 76.96±0.13(77.13)

analysis and motivation primarily adhere to rate-coding approximations, the performance on static datasets aligns with expectations. The results on the dynamic dataset CIFAR10-DVS also achieve comparable levels, implying a significant presence of rate-based representation within CIFAR10-DVS. More results regarding the performance comparisons between the proposed method and BPTT across various architectures and settings have been detailed in Appendix D.

5.3 Impact of Time Expansion

we assess the impact of extending timesteps on both accuracy and training efficiency. Figure 4a validates that our method capably manages increased timesteps, thereby confirming the scalability of the proposed method for larger T values. Figure 4b displays the computational and memory expenses incurred during the backward phase, which, as anticipated, do not escalate with increasing T.



(a) Comparison of Classification Performance (b) Comparison of Training Costs

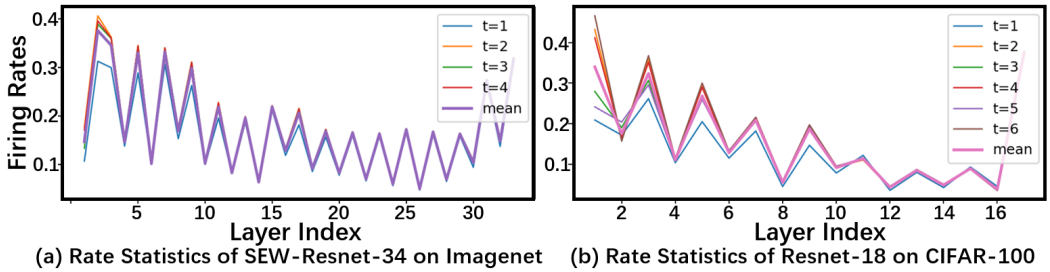
Figure 4: Results of BPTT and **rate**$_M$ across various timesteps.

Table 2: Performance w/o and w/ temporal shuffle for models trained by **rate**$_M$

Dataset	Model	Timesteps	Accuracy	Shuffled
CIFAR-10	ResNet-18	2	94.77	94.63±0.04
		4	95.51	95.50±0.04
		6	95.97	95.95±0.09
	VGG-11	2	95.13	95.10±0.05
		4	95.37	95.37±0.03
		6	95.77	95.79±0.05
CIFAR-100	ResNet-18	2	76.27	75.59±0.11
		4	78.32	77.72±0.15
		6	79.10	79.10±0.14
	VGG-11	2	77.46	77.21±0.12
		4	77.88	77.78±0.16
		6	77.97	78.02±0.09
ImageNet	SEW-ResNet-34	4	65.84	65.11±0.11
	PreAct-ResNet-34	4	70.01	69.78±0.10
CIFAR10-DVS	VGG-11	10	76.50	74.69±0.17

5.4 Analysis of Rate Statistics

Our method, derived from the principles of rate-based representation, necessitates examining the impact of rate coding on model behavior. Following an insightful approach from [6], we assess the robustness of our models by shuffling the temporal order of spike sequences while maintaining their rate consistency. This experiment, designed to disrupt temporal information without changing the firing rate, was applied to models trained using rate-based backpropagation. During inference on the test dataset, we introduced perturbations by randomly shuffling the temporal dimensions of input tensors across all neurons, as reported in Table 2. Notably, models mostly resisted these changes to some degree, which suggests that they follow the basic rules of rate coding, where the reordering of timesteps does not significantly impact overall accuracy. Furthermore, we tracked the average firing rates across each layer over time, presented in Figure 5. As layers increase, the average spike rates per layer are closely aligned with the temporal mean, validating the idea of rate-coding approximation. Those two experiments support the notion that rate-based backpropagation proficiently captures rate-based representations during training.



(a) Rate Statistics of SEW-Resnet-34 on Imagenet (b) Rate Statistics of Resnet-18 on CIFAR-100

Figure 5: Firing rates statistics for models trained by **rate**$_M$.

6 Conclusion

In this work, we propose rate-based backpropagation, utilizing rate-coding approximation to streamline the gradient computational graph, significantly reducing both memory usage and training time. Through theoretical analyses and empirical validation, we show the method's feasibility in approximating the optimization direction of BPTT. Experimental results across benchmarks reveal that our

method achieves comparable performance with BPTT and surpasses other state-of-the-art efficient training methods. We expect our work to pave the way for more scalable and resource-efficient training of SNNs.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 62304203), the Natural Science Foundation of Zhejiang Province, China (Grant No. LQ22F010011), and the ZJU-UIUC Center for Heterogeneously Integrated Brain-Inspired Computing.

References

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[2] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Thomas Bohnstingl, Stanisław Woźniak, Angeliki Pantazi, and Evangelos Eleftheriou. Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

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

[6] Tong Bu, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. Rate gradient approximation attack threats deep spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7896–7906, 2023.

[7] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*, 2023.

[8] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2015.

[9] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

[10] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

[13] Shikuang Deng, Hao Lin, Yuhang Li, and Shi Gu. Surrogate module learning: Reduce the gradient error accumulation in training spiking neural networks. In *International Conference on Machine Learning*, pages 7645–7657. PMLR, 2023.

[14] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[15] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–8. ieee, 2015.

[16] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv preprint arXiv:2105.11654*, 2021.

[17] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. *Advances in Neural Information Processing Systems*, 35:34377–34390, 2022.

[18] Rida El-Allami, Alberto Marchisio, Muhammad Shafique, and Ihsen Alouani. Securing deep spiking neural networks against adversarial attacks through inherent structural parameters. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 774–779. IEEE, 2021.

[19] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[20] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.

[21] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong 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.

[22] Pengjie Gu, Rong Xiao, Gang Pan, and Huajin Tang. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *IJCAI*, volume 15, pages 1366–1372, 2019.

[23] Wenzhe Guo, Mohammed E Fouda, Ahmed M Eltawil, and Khaled Nabil Salama. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience*, 15:638474, 2021.

[24] Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. Im-loss: information maximization loss for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:156–166, 2022.

[25] Yufei Guo, Xuhui Huang, and Zhe Ma. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023.

[26] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pages 388–404. Springer, 2020.

[27] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13558–13567, 2020.

[28] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11–21, 2023.

[29] Zecheng Hao, Tong Bu, Xinyu Shi, Zihan Huang, Zhaofei Yu, and Tiejun Huang. Threaten spiking neural networks through combining rate and temporal information. In *The Twelfth International Conference on Learning Representations*.

[30] Zecheng Hao, Tong Bu, Xinyu Shi, Zihan Huang, Zhaofei Yu, and Tiejun Huang. Threaten spiking neural networks through combining rate and temporal information. In *The Twelfth International Conference on Learning Representations*.

[31] Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[34] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: Towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pages 14945–14974. PMLR, 2023.

[35] Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation-and timing-based learning rules for spiking neural networks. *Advances in neural information processing systems*, 33:19534–19544, 2020.

[36] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11270–11277, 2020.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[38] Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5209–5218, 2021.

[39] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:228000, 2016.

[40] Chen Li, Lei Ma, and Steve Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022.

[41] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:244131, 2017.

[42] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pages 6316–6325. PMLR, 2021.

[43] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021.

[44] Yuhang Li, Youngeun Kim, Hyoungseob Park, and Priyadarshini Panda. Uncovering the representation of spiking neural networks trained with surrogate gradient. *arXiv preprint arXiv:2304.13098*, 2023.

[45] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

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

[47] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan 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*, pages 12444–12453, 2022.

[48] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan 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.

[49] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

[50] Bhaskar Mukhoty, Hilal AlQuabeh, Giulia De Masi, Huan Xiong, and Bin Gu. Certified adversarial robustness for rate encoded spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2023.

[51] Emre O Neftci, Hesham Mostafa, and Friedemann 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.

[52] Stefano Panzeri and Simon R Schultz. A unified approach to the study of temporal, correlational, and rate coding. *Neural Computation*, 13(6):1311–1349, 2001.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[54] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[55] Nitin Rathi and Kaushik 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.

[56] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[57] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:294078, 2017.

[58] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[59] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[60] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 399–414. Springer, 2020.

[61] Guobin Shen, Dongcheng Zhao, and Yi Zeng. Exploiting high performance spiking neural networks with efficient spiking patterns. *arXiv preprint arXiv:2301.12356*, 2023.

[62] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

[63] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[64] Kyle H Srivastava, Caroline M Holmes, Michiel Vellema, Andrea R Pack, Coen PH Elemans, Ilya Nemenman, and Samuel J Sober. Motor control by precisely timed spike patterns. *Proceedings of the National Academy of Sciences*, 114(5):1171–1176, 2017.

[65] Christoph Stöckl and Wolfgang Maass. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 3(3):230–238, 2021.

[66] Kazuma Suetake, Shin-ichi Ikegawa, Ryuji Saiin, and Yoshihide Sawada. S3nn: Time step reduction of spiking surrogate gradients for training energy efficient single-step spiking neural networks. *Neural Networks*, 159:208–219, 2023.

[67] Johannes Christian Thiele, Olivier Bichler, and Antoine Dupret. Spikegrad: An ann-equivalent computation model for implementing backpropagation with spikes. *arXiv preprint arXiv:1906.00851*, 2019.

[68] Jingtao Wang, Zengjie Song, Yuxi Wang, Jun Xiao, Yuran Yang, Shuqi Mei, and Zhaoxiang Zhang. Ssf: Accelerating training of spiking neural networks with stabilized spiking flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991, 2023.

[69] Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In *International Conference on Machine Learning*, pages 35798–35816. PMLR, 2023.

[70] Ziming Wang, Shuang Lian, Yuhao Zhang, Xiaoxin Cui, Rui Yan, and Huajin Tang. Towards lossless ann-snn conversion under ultra-low latency with dual-phase optimization. *arXiv preprint arXiv:2205.07473*, 2022.

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

[72] Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10320–10328, 2021.

[73] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):446–460, 2021.

[74] Yujie Wu, Lei Deng, Guoqi Li, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:323875, 2018.

[75] Timo C Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):12829, 2021.

[76] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35:20717–20730, 2022.

[77] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in neural information processing systems*, 34:14516–14528, 2021.

[78] Zhanglu Yan, Jun Zhou, and Weng-Fai Wong. Near lossless transfer learning for spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10577–10584, 2021.

[79] Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10221–10230, 2021.

[80] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:32160–32171, 2022.

[81] Bojian Yin, Federico Corradi, and Sander M Bohté. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In *International Conference on Neuromorphic Systems 2020*, pages 1–8, 2020.

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

[83] Chengting Yu, Zheming Gu, Da Li, Gaoang Wang, Aili Wang, and Erping Li. Stsc-snn: Spatio-temporal synaptic connection with temporal convolution and attention for spiking neural networks. *Frontiers in Neuroscience*, 16:1079357, 2022.

[84] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[85] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

[86] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in neural information processing systems*, 33:12022–12033, 2020.

[87] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi 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.

[88] Shibo Zhou, Xiaohua Li, Ying Chen, Sanjeev T Chandrasekaran, and Arindam Sanyal. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11143–11151, 2021.

[89] Yaoyu Zhu, Jianhao Ding, Tiejun Huang, Xiaodong Xie, and Zhaofei Yu. Online stabilization of spiking neural networks. In *The Twelfth International Conference on Learning Representations*.

A Proof of Theorems

Theorem 1. *Given $\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}$ that refers to gradients computed following the chain rule of BPTT in Eq. (2), and $\boldsymbol{\kappa}_t^l = \sum_\tau \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_\tau^l}$ (where $\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right] = \mathbb{E}\left[\boldsymbol{\varkappa}_t^l\right]$ in Eq.(6-7)) , if $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]$ holds for $\forall l$, we have $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right] = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^l}\right)_{rate}$. Furthermore, given $\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}$, if $\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}\right] = \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\right]\mathbb{E}[\boldsymbol{s}_t^{l-1}]$ for $\forall l$, we then obtain $(\nabla_{\boldsymbol{W}^l}\mathcal{L})_{rate} = \frac{1}{T}(\nabla_{\boldsymbol{W}^l}\mathcal{L})$. Here, $\mathbb{E}\left[\boldsymbol{x}_t\right] = \frac{1}{T}\sum_t \boldsymbol{x}_t$ refers the mean value of tensor $\boldsymbol{x}_t$ over temporal dimension T.*

Proof. Given $\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}$ and $\boldsymbol{\kappa}_t^l = \sum_\tau \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_\tau^l}$, we establish the mean gradients through neural dynamics based on the chain rule:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}\right] = \mathbb{E}\left[\sum_\tau \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_\tau^l}\frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{I}_t^l}\right] = \frac{1}{T}\sum_t\sum_\tau \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_\tau^l}\frac{\partial \boldsymbol{s}_\tau^l}{\partial \boldsymbol{I}_t^l} = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l], \tag{10}$$

Considering the output layer $l = L$, the objective for BPTT can be expressed as $\mathcal{L} = \ell(\mathbb{E}[\boldsymbol{o}_t], \boldsymbol{y}) = \ell(\mathbb{E}[\boldsymbol{W}^L\boldsymbol{s}_t^L], \boldsymbol{y}) = \ell(\boldsymbol{W}^L\mathbb{E}[\boldsymbol{s}_t^L], \boldsymbol{y}) = \ell(\boldsymbol{W}^L\boldsymbol{r}^L, \boldsymbol{y})$. Under the rate-based objective $\mathcal{L} = \frac{1}{T}\ell(\boldsymbol{c}^L, \boldsymbol{y}) = \frac{1}{T}\ell(\boldsymbol{W}^L\boldsymbol{r}^{L-1}, \boldsymbol{y})$, it is clear that $\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{L-1})}] = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^{L-1}}\right)_{\text{rate}}$. Applying the precondition $\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{L-1})}\boldsymbol{\kappa}_t^L] = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{L-1})}]\mathbb{E}[\boldsymbol{\kappa}_t^{L-1}]$, we obtain:

$$\begin{aligned}
\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^{L-1}}\frac{\partial \boldsymbol{r}^{L-1}}{\partial \boldsymbol{c}^{L-2}}\frac{\partial \boldsymbol{c}^{L-2}}{\partial \boldsymbol{r}^{L-2}}\right)_{\text{rate}} &= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^{L-1}}\right)_{\text{rate}}\mathbb{E}[\boldsymbol{\kappa}_t^{L-1}]\boldsymbol{W}^{(L-1)^\top} \\
&= \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{L-1})}]\mathbb{E}[\boldsymbol{\kappa}_t^{L-1}]\boldsymbol{W}^{(L-1)^\top} = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{L-1})}\boldsymbol{\kappa}_t^{L-1}]\boldsymbol{W}^{(L-1)^\top} \\
&= \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^{L-1}}\right]\boldsymbol{W}^{(L-1)^\top} = \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^{L-1}}\boldsymbol{W}^{(L-1)^\top}\right] \\
&= \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^{L-2}}\right] = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{L-2})}],
\end{aligned} \tag{11}$$

Continuing this induction process, we can derive that $\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}] = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^l}\right)_{\text{rate}}$ for all layers l. Further, given $\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}$, the gradient for the weight matrix under BPTT: $\nabla_{\boldsymbol{W}^l}\mathcal{L} = \sum_t\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}\frac{\partial \boldsymbol{I}_t^l}{\partial \boldsymbol{W}^l}\right) = \sum_t \boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}$. The gradients passing through the linear parts maintain the equivalence:

$$\begin{aligned}
\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\right)_{\text{rate}} &= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^{l+1}}\frac{\partial \boldsymbol{r}^{l+1}}{\partial \boldsymbol{c}^l}\right)_{\text{rate}} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^{l+1}}\boldsymbol{W}^{l+1^\top}\right)_{\text{rate}} \\
&= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^{l+1}}\right)_{\text{rate}}\boldsymbol{W}^{l+1^\top} = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^{l+1})}\boldsymbol{W}^{l+1^\top}] = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}].
\end{aligned} \tag{12}$$

With the precondition that $\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}] = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}]\mathbb{E}[\boldsymbol{s}_t^{l-1}]$ holds for $\forall l$, we obtain:

$$\begin{aligned}
(\nabla_{\boldsymbol{W}^l}\mathcal{L})_{\text{rate}} &= \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\frac{\partial \boldsymbol{c}^l}{\partial \boldsymbol{W}^l}\right)_{\text{rate}} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\right)_{\text{rate}}\boldsymbol{r}^{l-1} \\
&= \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}]\mathbb{E}[\boldsymbol{s}_t^{l-1}] = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}\boldsymbol{s}_t^{l-1}] = \frac{1}{T}\nabla_{\boldsymbol{W}^l}\mathcal{L}.
\end{aligned} \tag{13}$$

Theorem 2. *For gradients $\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^l}$ and $\boldsymbol{\kappa}_t^l = \sum_\tau \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_\tau^l}$, given the approximation error bound $\epsilon > 0$ s.t. $\left\|\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l\right] - \mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\mathbb{E}\left[\boldsymbol{\kappa}_t^l\right]\right\| \leq \epsilon(1 + \left\|\mathbb{E}\left[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\right]\right\|)$ for $\forall l$. Denote the stacked tensor $\boldsymbol{I}^l = [\boldsymbol{I}_1^l, ..., \boldsymbol{I}_T^l]$ and $\boldsymbol{s}^l = [\boldsymbol{s}_1^l, ..., \boldsymbol{s}_T^l]$. Assuming the backward procedure follows non-expansivity s.t. $\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l} = \boldsymbol{W}^{l+1^\top}\frac{\partial \boldsymbol{s}^l}{\partial \boldsymbol{I}^l}$ is 1-lipschitz continuous without loss of generality and the biases are bounded uniformly by B, i.e. $\left\|\boldsymbol{x}\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l} - \hat{\boldsymbol{x}}\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l}\right\| \leq \left\|\boldsymbol{x} - \hat{\boldsymbol{x}}\right\|$ for $\forall \boldsymbol{x}, \hat{\boldsymbol{x}}$. Define $\boldsymbol{\delta}_{rate}^l = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l}\right)_{rate}$ as the*

error propagated through Eq. (7), and $\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{I}_t^l}$ *as the error propagated through BPTT, with* $\boldsymbol{\delta}_{rate}^L = \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^L)}]$. *We have the gradient difference bounded by* $\left\|\boldsymbol{\delta}_{rate}^{L-k} - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{L-k})}]\right\| = \mathcal{O}(k^2\epsilon)$.

Proof. Given that the error backpropagation $\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l}$ follows a 1-Lipschitz condition with biases bounded by B for all l, we can derive $\left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}]\right\| = \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})} \frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}^l}]\right\| \leq \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}]\right\| + B$ by non-expansivity. Then, by induction, we obtain the gradient bound between the intermediate layers and the final layer:

$$\left\|(\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}])\right\| \leq (L-l)B + \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^L)}]\right\|.$$

Since $\frac{\partial \boldsymbol{I}^{l+1}}{\partial \boldsymbol{I}_t^l} = \boldsymbol{W}^{l+1^\top} \boldsymbol{\kappa}_t^l$ is also 1-Lipschitz continuous without loss of generality, given the approximated approximated error $\epsilon > 0$ s.t.

$$\begin{aligned}\left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}\boldsymbol{W}^{l+1^\top}\mathbb{E}[\boldsymbol{\kappa}_t^l] - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}\boldsymbol{W}^{l+1^\top}\boldsymbol{\kappa}_t^l]\right\| &= \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}]\mathbb{E}[\boldsymbol{\kappa}_t^l] - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l]\right\| \\ &\leq \epsilon(1 + \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{s}^l)}\boldsymbol{\kappa}_t^l]\right\|) = \epsilon(1 + \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}\boldsymbol{W}^{l+1^\top}\boldsymbol{\kappa}_t^l]\right\|)\end{aligned} \tag{14}$$

we have

$$\begin{aligned}\left\|\boldsymbol{\delta}_{\text{rate}}^l - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}]\right\| =& \left\|\boldsymbol{\delta}_{\text{rate}}^{l+1}\boldsymbol{W}^{l+1^\top}\mathbb{E}[\boldsymbol{\kappa}_t^l] - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}]\boldsymbol{W}^{l+1^\top}\boldsymbol{\kappa}_t^l]\right\| \\ =& \Big\| \Big(\boldsymbol{\delta}_{\text{rate}}^{l+1}\boldsymbol{W}^{l+1^\top}\mathbb{E}[\boldsymbol{\kappa}_t^l] - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}]\boldsymbol{W}^{l+1^\top}\mathbb{E}[\boldsymbol{\kappa}_t^l]\Big) \\ &+ \Big(\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}\boldsymbol{W}^{l+1^\top}\mathbb{E}[\boldsymbol{\kappa}_t^l] - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}\boldsymbol{W}^{l+1^\top}\boldsymbol{\kappa}_t^l]\Big)\Big\| \\ \leq& \left\|\boldsymbol{\delta}_{\text{rate}}^{l+1} - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}]\right\| + \epsilon(1 + \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}\boldsymbol{W}^{l+1^\top}\boldsymbol{\kappa}_t^l]\right\|) \\ \leq& \left\|\boldsymbol{\delta}_{\text{rate}}^{l+1} - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^{l+1})}]\right\| + \epsilon(1 + (L-l)B + \left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^L)}]\right\|)\end{aligned} \tag{15}$$

By induction, we obtain

$$\left\|\boldsymbol{\delta}_{\text{rate}}^l - \mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^l)}]\right\| \leq \epsilon\Big((L-l) + \frac{(L-l+1)(L-l)}{2}B + (L-l)\left\|\mathbb{E}[\boldsymbol{\delta}_t^{(\boldsymbol{I}^L)}]\right\|\Big) = \mathcal{O}\Big((L-l)^2\epsilon\Big) \tag{16}$$

B Implementation Details

B.1 Pseudocode of the Rate-based Backpropagation

The pseudocode for rate-based backpropagation, illustrating the implementations for both $\textbf{rate}_M$ and $\textbf{rate}_S$, is provided in Algorithm 1.

B.2 About Training Modes in Rate-based Backpropagation

In direct training, two distinct implementation modes are recognized, activation-based and time-based [19], differing fundamentally in their handling of the simulation timestep T. The activation-based, also known as multi-step mode, processes the T loop separately within each layer, transmitting inter-layer tensors within dimensions $[T, B, S]$, where B and S refer to batch and spatial dimensions, respectively. The configuration enables the multi-step mode to enhance computational efficiency by reformatting the tensor dimensions as $[T \times B, S]$ to optimize parallelism in linear parts. However, the coupled processing with temporal calculations embedded within the layers increases memory retention on GPUs, potentially obscuring the benefits of memory cost optimization in both online training and our proposed methods. In contrast, the time-based mode externalizes the T loop, facilitating single-step forward computations at each timestep. This single-step mode aligns well with the dynamic modeling of temporal dimensions and facilitates memory optimization strategies more effectively. However, its restriction on parallel computation in linear components compared to multi-step mode necessitates increased forward time on GPUs, albeit with enhanced support for memory optimization. Our proposed method has been adapted to operate effectively within both frameworks to ensure comprehensiveness, as shown in Algorithm 1.

Algorithm 1: Single Training Iteration of the Rate-based Backpropagation

Input: Timesteps T; Network depth L; Trainable parameters $\{\boldsymbol{W}^l\}_{l\leq L}$; Training Mini-batch $\{(\boldsymbol{x}_t^0, \boldsymbol{y})\}$; Training Mode ***rate***$_S$ or ***rate***$_M$.

Output: Updated parameters $\{\boldsymbol{W}^l\}_{l\leq L}$

Initialize input spikes $\boldsymbol{s}_t^0 = \boldsymbol{x}_t^0$ for all $t \in [1, T]$.
if ***rate***$_M$ **then**
  for $l = 1$ *to* L **do**
    Compute input currents through linear operators $\boldsymbol{I}_t^l = \boldsymbol{W}^l \boldsymbol{s}_t^{l-1}$ for all $t \in [1, T]$;
    Initialize $\boldsymbol{\rho}_0^l = 0$, $\boldsymbol{g}_0^l = 0$, $\boldsymbol{e}_0^l = 0$.
    for $t = 1$ *to* T **do**
      Compute output spikes $\boldsymbol{s}_t^l$ from $\boldsymbol{I}_t^l$ following neural dynamics in Eq. (1);
      Compute the eligibility trace $\boldsymbol{\rho}_t^l = 1 + \boldsymbol{\rho}_{t-1}^l \left(\frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{u}_{t-1}^l} + \frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{s}_{t-1}^l} \frac{\partial \boldsymbol{s}_{t-1}^l}{\partial \boldsymbol{u}_{t-1}^l} \right)$ in Eq. (8);
      Accumulate $\boldsymbol{e}_t^l = \frac{1}{t}((t-1)\boldsymbol{e}_{t-1}^l + \boldsymbol{s}_t^l)$;
      Accumulate $\boldsymbol{g}_t^l = \frac{1}{t}((t-1)\boldsymbol{g}_{t-1}^l + \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l}\boldsymbol{\rho}_t)$.
    end
    Save $\boldsymbol{e}_T^l$, $\boldsymbol{g}_T^l$ and $\boldsymbol{W}^l$ for backwards, and free intermediate variables.
  end
else
  Initialize $\boldsymbol{\rho}_0^l = 0$, $\boldsymbol{g}_0^l = 0$, $\boldsymbol{e}_0^l = 0$ for all $l \in [1, L]$.
  for $t = 1$ *to* T **do**
    for $l = 1$ *to* L **do**
      Compute input currents through linear operators $\boldsymbol{I}_t^l = \boldsymbol{W}^l \boldsymbol{s}_t^{l-1}$;
      Initialize $\boldsymbol{\rho}_0^l = 0$, $\boldsymbol{g}_0^l = 0$, $\boldsymbol{e}_0^l = 0$;
      Compute output spikes $\boldsymbol{s}_t^l$ from $\boldsymbol{I}_t^l$ following neural dynamics in Eq. (1);
      Compute the eligibility trace $\boldsymbol{\rho}_t^l = 1 + \boldsymbol{\rho}_{t-1}^l \left(\frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{u}_{t-1}^l} + \frac{\partial \boldsymbol{u}_t^l}{\partial \boldsymbol{s}_{t-1}^l} \frac{\partial \boldsymbol{s}_{t-1}^l}{\partial \boldsymbol{u}_{t-1}^l} \right)$ in Eq. (8);
      Accumulate $\boldsymbol{e}_t^l = \frac{1}{t}((t-1)\boldsymbol{e}_{t-1}^l + \boldsymbol{s}_t^l)$;
      Accumulate $\boldsymbol{g}_t^l = \frac{1}{t}((t-1)\boldsymbol{g}_{t-1}^l + \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{u}_t^l}\boldsymbol{\rho}_t)$;
      Save $\boldsymbol{u}_t^l$, $\boldsymbol{s}_t^l$ for neuron states;
      Save $\boldsymbol{g}_t^l$, $\boldsymbol{e}_t^l$, $\boldsymbol{\rho}_t^l$ as eligibility traces.
    end
  end
end
Compute the outputs gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^L}$ from the objective function.
for $l = L - 1$ *to* 1 **do**
  Compute error backpropagated through the linear part $\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^l} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^{l+1}} \boldsymbol{W}^{l+1^\top}$;
  Compute error backpropagated through the neuron part $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^l} \boldsymbol{g}_T^l$;
  Compute the weight gradients $\nabla_{\boldsymbol{W}^l} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{c}^l} (\boldsymbol{e}_T^{l-1})^\top$;
  Update parameters $\{\boldsymbol{W}^l\}_{l\leq L}$ based on the gradient-based optimizer.
end

B.3 Implementation of Batch Normalization in Rate-based Backpropagation

In the forward pass, batch normalization (BN) precedes neuron activation, scaling inputs $\boldsymbol{I}_t^l$ and introduces a bias in the average inputs $\boldsymbol{c} = \mathbb{E}[\boldsymbol{I}_t]$. We denote $\tilde{\boldsymbol{c}}$ to represent the biased average inputs as $\tilde{\boldsymbol{c}} = \mathbb{E}[\tilde{\boldsymbol{I}}_t] = \mathbb{E}[\text{BN}(\boldsymbol{I}_t)]$ instead of $\boldsymbol{c}$. Note that BN acts as a linear operation during inference, where $\tilde{\boldsymbol{c}} = \mathbb{E}[\tilde{\boldsymbol{I}}_t] = \mathbb{E}[\text{BN}(\boldsymbol{I}_t)] = \text{BN}(\mathbb{E}[\boldsymbol{I}_t]) = \text{BN}(\boldsymbol{c})$. Implementing rate-based propagation requires considering how gradients pass through the BN layers and affect their intrinsic parameters during training. Initially, we explore the spatial BN [48] design for the single-step mode, which

computes mean and variance statistics independently at each time step t:

$$\tilde{\boldsymbol{I}}_t = \mathrm{BN}(\boldsymbol{I}_t) = \boldsymbol{\gamma}\left(\frac{\boldsymbol{I}_t - \boldsymbol{\mu}_t}{\sqrt{\boldsymbol{\sigma}_t^2 + \epsilon}}\right) + \boldsymbol{\beta}, \text{ where } \boldsymbol{\mu}_t = \frac{1}{B}\sum_b \boldsymbol{I}_t^{(b)} \text{ and } \boldsymbol{\sigma}_t^2 = \frac{1}{B}\sum_b (\boldsymbol{I}_t^{(b)} - \boldsymbol{\mu}_t)^2. \quad (17)$$

Defining $\boldsymbol{\chi}_t^{(\boldsymbol{I})} = \frac{\partial \tilde{\boldsymbol{I}}_t}{\partial \boldsymbol{I}_t}, \boldsymbol{\chi}_t^{(\boldsymbol{\gamma})} = \frac{\partial \tilde{\boldsymbol{I}}_t}{\partial \boldsymbol{\gamma}}, \boldsymbol{\chi}_t^{(\boldsymbol{\beta})} = \frac{\partial \tilde{\boldsymbol{I}}_t}{\partial \boldsymbol{\beta}}$, the following expressions are obtained:

$$\boldsymbol{\chi}_t^{(\boldsymbol{I})} = \boldsymbol{\gamma}\frac{1}{\sqrt{\boldsymbol{\sigma}_t^2 + \epsilon}} + \frac{\partial \tilde{\boldsymbol{I}}_t^l}{\partial \boldsymbol{\sigma}_t^2}\frac{\partial \boldsymbol{\sigma}_t^2}{\partial \boldsymbol{I}_t^l} + \frac{\partial \tilde{\boldsymbol{I}}_t^l}{\partial \boldsymbol{\mu}_t}\frac{\partial \boldsymbol{\mu}_t}{\partial \boldsymbol{I}_t^l}, \quad \boldsymbol{\chi}_t^{(\boldsymbol{\gamma})} = \frac{\boldsymbol{I}_t^l - \boldsymbol{\mu}_t}{\sqrt{\boldsymbol{\sigma}_t^2 + \epsilon}}, \quad \boldsymbol{\chi}_t^{(\boldsymbol{\beta})} = Id. \quad (18)$$

For the backward derivation of BN in a rate-based setting based on mean estimations through time, we implement $\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}} = \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{c}}}\mathbb{E}[\boldsymbol{\chi}_t^{(\boldsymbol{I})}], \frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} = \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{c}}}\mathbb{E}[\boldsymbol{\chi}_t^{(\boldsymbol{\gamma})}], \frac{\partial L}{\partial \boldsymbol{\beta}} = \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{c}}}\mathbb{E}[\boldsymbol{\chi}_t^{(\boldsymbol{\beta})}] = \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{c}}}$. Since gradient computation at each timestep is independent, the dynamic estimations of $\mathbb{E}[\boldsymbol{\chi}_t^{(\boldsymbol{I})}]$ and $\mathbb{E}[\boldsymbol{\chi}_t^{(\boldsymbol{\gamma})}]$ are performed in the same manner of $\{\boldsymbol{e}_t^l\}_{t\leq T}$ and $\{\boldsymbol{g}_t^l\}_{t\leq T}$.

In the multi-step mode, tdBN [87] accounts for mean and variance statistics over the entire time horizon:

$$\tilde{\boldsymbol{I}}_t = \mathrm{BN}(\boldsymbol{I}_t) = \boldsymbol{\gamma}\left(\frac{\boldsymbol{I}_t - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}\right) + \boldsymbol{\beta}, \text{where } \boldsymbol{\mu} = \frac{1}{BT}\sum_t\sum_b \boldsymbol{I}_t^{(b)}, \boldsymbol{\sigma}^2 = \frac{1}{BT}\sum_t\sum_b (\boldsymbol{I}_t^{(b)} - \boldsymbol{\mu})^2. \quad (19)$$

The rate-based representation integrates the input across the time dimension, with the mean $\boldsymbol{\mu}_c = \sum_b \boldsymbol{c}^{(b)}$, and variance $\boldsymbol{\sigma}_c^2 = \frac{1}{B}\sum_b (\boldsymbol{c}^{(b)} - \boldsymbol{\mu}_c)^2$. Since $\boldsymbol{c}$ is the temporal mean of inputs, it is clear that $\boldsymbol{\mu}_c = \boldsymbol{\mu}$ and $\boldsymbol{\sigma}_c^2 \leq \boldsymbol{\sigma}^2$. Note that $\frac{\partial \boldsymbol{\sigma}^2}{\partial \boldsymbol{I}_t} = \frac{1}{BT}\sum_t\sum_b (\boldsymbol{I}_t^{(b)} - \boldsymbol{\mu}) = \frac{1}{B}\sum_b (\boldsymbol{c}^{(b)} - \boldsymbol{\mu}_c) = \frac{\partial \boldsymbol{\sigma}_c^2}{\partial \boldsymbol{c}}$. Assuming $\frac{\partial \boldsymbol{I}_t}{\partial \boldsymbol{c}} = Id$, we derive $\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{I}_t} = \frac{\partial \boldsymbol{\mu}_c}{\partial \boldsymbol{c}}$. For the forward approximation specifically tailored for tdBN in rate-based backpropagation, we define:

$$\tilde{\boldsymbol{c}} = \hat{\mathrm{BN}}(\boldsymbol{c}) = \boldsymbol{\gamma}\left(\frac{\boldsymbol{c} - \boldsymbol{\mu}}{\sqrt{\hat{\boldsymbol{\sigma}}_c^2 + \epsilon}}\right) + \boldsymbol{\beta}, \quad (20)$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ refer to the same intrinsic parameters shared with $\mathrm{BN}(\boldsymbol{I}_t)$, and $\hat{\boldsymbol{\sigma}}_c^2$ is defined distinctly in forward and backward passes: $\hat{\boldsymbol{\sigma}}_c^2 = \boldsymbol{\sigma}^2$ in forward and $\frac{\partial \hat{\boldsymbol{\sigma}}_c^2}{\partial \boldsymbol{\sigma}_c^2} = Id$ in backward. The implementation utilizes gradient replacement with the detach operation in PyTorch: $\hat{\boldsymbol{\sigma}}_c^2 = detach(\boldsymbol{\sigma}^2 - \boldsymbol{\sigma}_c^2) + \boldsymbol{\sigma}_c^2$. Thus, in the forward phase, $\tilde{\boldsymbol{c}} = \hat{\mathrm{BN}}(\boldsymbol{c}) = \mathbb{E}[\mathrm{BN}(\boldsymbol{I}_t)]$, and in the backward phase, $\frac{\partial \tilde{\boldsymbol{c}}}{\partial \boldsymbol{c}} = \mathbb{E}[\frac{\partial \tilde{\boldsymbol{c}}}{\partial \boldsymbol{I}_t}]$, aligning perfectly with the foundational principles of rate-based backpropagation.

C Experimental Settings

C.1 Datasets

CIFAR-10 and CIFAR-100. The CIFAR-10 and CIFAR-100 [37] datasets contain 32x32 color images across different classes, licensed under MIT. CIFAR-10 includes 60,000 images across 10 classes, with 50,000 for training and 10,000 for testing, whereas CIFAR-100 is spread over 100 classes. Both datasets have been normalized for zero mean and unit variance. Image data augmentation is applied using AutoAugment [9] and Cutout [14] strategies, similar to the implementations in recent studies [42, 7, 24, 69, 13]. The pixel values are directly fed into the input layer at each timestep as direct encoding [55].

ImageNet. The ImageNet-1K dataset [11] comprises 1,281,167 training images and 50,000 validation images distributed across 1,000 classes, licensed for non-commercial use. ImageNet-1K images are normalized for zero mean and unit variance. Training images undergo random resized cropping to 224x224 pixels and horizontal flipping, while validation images are resized to 256x256 and then center-cropped to 224x224. The images are transformed into time sequences through direct encoding [55], following the approach used for CIFAR datasets.

CIFAR10-DVS. The CIFAR10-DVS dataset [41] is a neuromorphic version of CIFAR-10, which includes 10,000 event-based images captured by the DVS camera with pixel dimensions expanded to

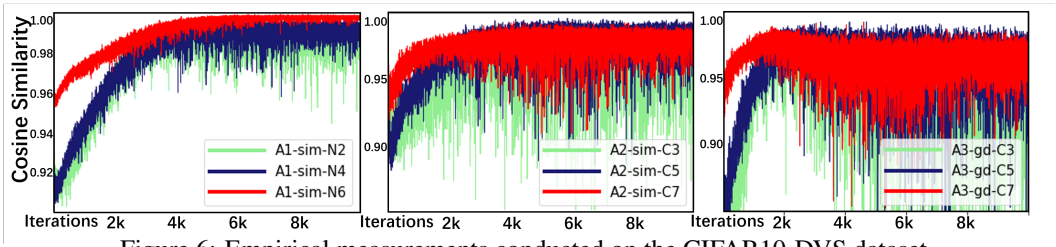


Figure 6: Empirical measurements conducted on the CIFAR10-DVS dataset.

128×128, licensed under CC BY 4.0. We split the whole dataset into 9000 training images and 1000 testing images. Data preprocessing involves integrating events into frames [21, 19] and reducing the spatial resolution to 48x48 through interpolation. Additional data augmentation includes random horizontal flips and random rolls within a 5-pixel range, mirroring previous methods [76, 48].

C.2 Training Setup

Network Architectures. For the CIFAR-10, CIFAR-100, and CIFAR10-DVS datasets, our method is tested on standard network architectures, including ResNet-18, ResNet-19, and VGG-11 [63, 32, 87, 76, 19, 69]. On the ImageNet dataset, we adapt two variations on ResNet architecture [32], SEW-ResNet-34 [87] specially proposed for SNNs, and ResNet-34 with pre-activation residual blocks [33], aligning with previous works [76, 48, 89]. While OTTT [76] and SLTT [48] frameworks utilize normalization-free techniques under the ResNet-34 framework [5], Zhu et al. [89] substitute these with their custom-designed batch normalization. We directly employ tdBN [87] instead of normalization-free methods in our experiments.

Training Details. This work utilizes the widely adopted sigmoid-based surrogate gradient [19] to approximate the Heaviside step function using $h(x, \alpha) = \frac{1}{1+e^{\alpha x}}$ and sets $\alpha = 4$ to ensure the maximum derivative of the surrogate function is 1 for preventing gradient explosion. All implementations are based on the PyTorch [53] and SpikingJelly [19] frameworks. The experiments on CIFAR-10, CIFAR-100, and CIFAR10-DVS datasets run on one NVIDIA GeForce RTX 3090 GPU. For ImageNet, distributed data parallel processing is utilized across eight NVIDIA GeForce RTX 4090 GPUs. We use the SGD optimizer [58] with a momentum of 0.9 for all tasks, integrating a cosine annealing strategy [45] for the learning rate schedule. Other hyperparameters are listed in Table 3.

Table 3: Training hyperparameters.

	CIFAR-10	CIFAR-100	ImageNet	CIFAR10-DVS
Epoch	300	300	100	300
Learning rate	0.1	0.1	0.2	0.1
Batch size	128	128	512	128
Weight decay	5e-4	5e-4	2e-5	5e-4

D More Results

D.1 Empirical Validation on CIFAR10-DVS

As shown in Figure 6, we extend conduct empirical experiments on CIFAR10-DVS as avalidation in the case of dynamic datasets. The observations confirm that, even in data with a degree of temporal information, the empirical validation of the assumptions remains consistent with expectations. This alignment emphasizes that the approximate relationship between rate-based backpropagation and BPTT remains substantially consistent. As a result, this stability ensures that our approach continues to effectively extract rate-based representations from neuromorphic datasets with a degree of temporal dynamics, thereby maintaining robust performance across diverse data scenarios.

D.2 Extended Performance Comparisons with BPTT

We conduct additional experiments to illustrate the comparative performance of rate-based back-propagation versus BPTT, as presented in Table 4 for CIFAR-10 and Table 5 for CIFAR-100. These experiments span various configurations, including different network architectures—ResNet-18,

Table 4: Performance comparison of rate-based backpropagation and BPTT on CIFAR-10.

Training	Model	Timesteps	Top-1 Acc (%)
BPTT_S	ResNet-18	2	95.02
		4	95.53
		6	95.68
	ResNet-19	2	96.12
		4	96.38
		6	96.57
	VGG-11	2	95.27
		4	95.61
		6	95.63
$\textbf{rate}_S$	ResNet-18	2	94.82±0.07(94.89)
		4	95.42±0.11(95.56)
		6	95.73±0.03(95.78)
	ResNet-19	2	96.11±0.05(96.18)
		4	96.32±0.04(96.38)
		6	96.38±0.06(96.45)
	VGG-11	2	95.44±0.02(95.46)
		4	95.57±0.08(95.68)
		6	95.64±0.12(95.76)
BPTT_M	ResNet-18	2	94.93
		4	95.64
		6	96.03
	ResNet-19	2	96.16
		4	96.49
		6	96.70
	VGG-11	2	95.31
		4	95.67
		6	95.64
$\textbf{rate}_M$	ResNet-18	2	94.75±0.05(94.82)
		4	95.61±0.02(95.64)
		6	95.90±0.07(96.01)
	ResNet-19	2	96.23±0.10(96.33)
		4	96.26±0.03(96.29)
		6	96.38±0.02(96.40)
	VGG-11	2	95.17±0.12(95.35)
		4	95.30±0.06(95.37)
		6	95.23±0.06(95.32)

Table 5: Performance comparison of rate-based backpropagation and BPTT on CIFAR-100.

Training	Model	Timesteps	Top-1 Acc (%)
BPTT_S	ResNet-18	2	76.24
		4	77.72
		6	78.65
	ResNet-19	2	79.33
		4	80.12
		6	80.77
	VGG-11	2	77.37
		4	77.82
		6	78.13
$\textbf{rate}_S$	ResNet-18	2	75.89±0.11(75.97)
		4	77.73±0.28(77.93)
		6	78.86±0.08(78.94)
	ResNet-19	2	79.71±0.02(79.74)
		4	80.41±0.14(80.54)
		6	80.75±0.05(80.79)
	VGG-11	2	77.34±0.04(77.37)
		4	77.87±0.35(78.13)
		6	78.23±0.03(78.27)
BPTT_M	ResNet-18	2	77.09
		4	77.93
		6	78.35
	ResNet-19	2	80.01
		4	81.07
		6	81.12
	VGG-11	2	77.42
		4	77.96
		6	78.25
$\textbf{rate}_M$	ResNet-18	2	75.97±0.20(76.27)
		4	78.26±0.12(78.38)
		6	79.02±0.11(79.16)
	ResNet-19	2	79.87±0.03(79.90)
		4	80.71±0.12(80.84)
		6	80.83±0.07(80.94)
	VGG-11	2	77.40±0.05(77.46)
		4	77.86±0.03(77.89)
		6	77.99±0.11(78.11)

Table 6: Comparison results of performance and training costs across various timesteps. All units for time measurements are in seconds per batch. Experiments were conducted on NVIDIA GeForce RTX 4090, with training settings consistent with other experiments.

Datasets	Network	Method		Timesteps				
				T=1	T=2	T=4	T=8	T=16
CIFAR100	ResNet-18	**rate**$_M$	Time of Eligibility Track	0.003	0.004	0.007	0.015	0.027
			Time of Backward	0.034	0.035	0.036	0.034	0.036
			Time of both	0.037	0.039	0.043	0.049	0.063
			Memory Allocated	1.8492	1.8488	1.8473	1.8496	1.8483
			Top-1 Acc [%]	74.60	76.04	78.24	79.24	79.37
		BPTT$_M$	Time of Backward	0.023	0.044	0.098	0.199	0.564
			Memory Allocated	1.4272	2.4454	4.4804	8.0460	15.685
			Top-1 Acc [%]	74.38	76.65	78.49	78.35	
	ResNet-19	**rate**$_M$	Time of Eligibility Track	0.006	0.012	0.020	0.041	
			Time of Backward	0.083	0.083	0.082	0.083	
			Time of both	0.089	0.095	0.102	0.124	
			Memory Allocated [GB]	4.4787	4.4798	4.4788	4.4784	
			Top-1 Acc [%]	78.3	80.00	80.65	81.31	
		BPTT$_M$	Time of Backward	0.046	0.111	0.285	0.552	
			Memory Allocated [GB]	3.2556	5.6636	10.8978	20.3862	
			Top-1 Acc [%]	78.39	80.06	81.11	81.13	
	VGG11	**rate**$_M$	Time of Eligibility Track	0.003	0.003	0.006	0.011	0.020
			Time of Backward	0.017	0.017	0.017	0.017	0.018
			Time of both	0.020	0.020	0.023	0.028	0.038
			Memory Allocated [GB]	1.3624	1.3607	1.3619	1.3613	1.3601
			Top-1 Acc [%]	76.13	77.59	77.75	78.34	78.65
		BPTT$_M$	Time of Backward	0.010	0.021	0.054	0.135	0.384
			Memory Allocated [GB]	0.9911	1.6784	3.7363	6.6141	12.3768
			Top-1 Acc [%]	76.34	77.20	77.98	78.26	78.37
ImageNet	SEW-ResNet-34	**rate**$_M$	Time of Eligibility Track	0.012	0.014	0.023		
			Time of Backward	0.074	0.074	0.074		
			Time of both	0.086	0.088	0.097		
			Memory Allocated [GB]	5.7887	5.7898	5.7883		
		BPTT$_M$	Time of Backward	0.046	0.095	0.233		
			Memory Allocated [GB]	3.9858	6.8654	12.5597		
	PreAct-ResNet-34	**rate**$_M$	Time of Eligibility Track	0.007	0.009	0.020		
			Time of Backward	0.072	0.071	0.072		
			Time of both	0.079	0.080	0.092		
			Memory Allocated [GB]	5.4995	5.4982	5.4942		
		BPTT$_M$	Time of Backward	0.046	0.088	0.211		
			Memory Allocated [GB]	3.7017	6.4778	11.969		

ResNet-19, and VGG-11—and timesteps (T=2, 4, 6). The results demonstrate that rate-based back-propagation maintains competitive accuracy with BPTT across different architectures and timestep settings on benchmark datasets.

D.3 Comprehensive Evaluation of Training Costs

To enhance the understanding of the scalability of the proposed method, we extended our analysis to include training costs across the CIFAR-100 and ImageNet datasets, utilizing additional network architectures as detailed in Table 6. This comprehensive evaluation aimed to assess the impact of varying time steps on performance, memory, and time costs. We integrated the computation of eligibility traces during the forward process, ensuring a fair comparison by incorporating these

iterative computations into the overall cost assessment. The results reveal that the total cost of rate-based backpropagation demonstrates a clear advantage over BPTT when timesteps $T \geq 2$, which underscores the efficiency of the proposed method approach in managing computational resources while maintaining comparative performance across various datasets and network architectures.

E Social Impacts and Limitations

There is no direct negative societal impact since this work centers on enhancing the training efficiency of SNNs. SNNs inherently require less energy for inference compared to ANNs, helping reduce carbon dioxide emissions. The methods developed in this work further optimize SNNs training by improving both memory and time efficiency, potentially reducing the overall resource consumption and environmental footprint of training processes. Regarding limitations, this work primarily compares with BPTT baselines, and there is potential for incorporating strategies from state-of-the-art techniques in future work. Moreover, the proposed method is tailored for tasks that utilize rate-coding, designed to efficiently capture spatial rate-based feature representations to enhance training; therefore, it necessitates further adaptation to effectively manage sequential tasks. Future efforts may need to delve deeper into adapting the dynamic characteristics of spikes and robustly designing training hyperparameters, ensuring compatibility with rate-based backpropagation and extending applicability to a wider range of applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract and Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix E.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Section 4 and Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 4.4, Appendix B and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Supplementary Materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Appendix C.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We do not use new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We use the existing common datasets.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We use the existing common datasets.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.