TOWARDS STABLE AND SCALABLE EVENT-DRIVEN TRAINING OF SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) offer a promising route towards energy-efficient deep learning on account of their sparse, event-driven computation. Activation-based training achieves competitive performance but does not exploit the event-driven computation during training, leading to memory inefficiency. In contrast, event-driven (time-based) training performs credit assignment with more efficient memory utilization, but remains underexplored and has not been shown to scale favorably across architectures, datasets, and inference timesteps. In this work, we develop an event-driven training approach by identifying and addressing sources of instability in existing formulations, thereby enabling stable and scalable training across architectures and inference timesteps. Our approach retains the memory-efficient formulation of event-driven learning while enabling the effective use of batch normalization, which we show to be important for SNN training stability, without relying on ad-hoc methods such as supervisory spike injection used in previous approaches. We obtain state-of-the-art results among event-driven methods on both static and neuromorphic benchmarks, with a 5% improvement in accuracy at less than half the inference time steps on CIFAR100, and even extend event-driven training to Tiny-ImageNet. Overall, our results establish event-driven learning as a scalable and efficient alternative to activation-based training for deep SNNs.

1 INTRODUCTION

Deep artificial neural networks (ANNs) have achieved remarkable success for a wide range of applications, often matching or surpassing human-level performance in tasks such as image recognition (Krizhevsky et al., 2012), speech synthesis (Van den Oord et al., 2016), and natural language processing (Touvron et al., 2023). Despite these advances, ANNs rely on dense, continuous-valued computations that incur substantial energy and computational costs. In contrast, spiking neural networks (SNNs) communicate through discrete binary spikes in an event-driven manner, offering a more biologically plausible framework (Maass, 1997; Gerstner & Kistler, 2002). Unlike conventional ANNs that rely heavily on multiply-accumulate (MAC) operations, SNNs primarily operate using simple accumulate operations driven by incoming spikes, effectively reducing expensive multiplications to additions. This leads to improved computational efficiency and lower power consumption, particularly when deployed on neuromorphic hardware (Davies et al., 2018; Akopyan et al., 2015). However, direct training of SNNs is challenging due to the non-differentiable nature of spike generation. As a result, ANNs continue to outperform SNNs in terms of accuracy. To address this, previous work has been largely based on ANN-to-SNN conversion techniques (Ramesh & Srinivasan, 2025; Bu et al., 2022; You et al., 2024) and activation-based learning methods, which approximate the derivatives of the spiking function through surrogate gradients (Neftci et al., 2019; Diehl et al., 2015; Rueckauer et al., 2017; Sengupta et al., 2019; Deng et al., 2022).

An alternative approach is *event-driven* or time-based training, which leverages precise spike timings as the carrier of information. In activation-based approaches, information is encoded in spike counts over a time window, effectively discarding precise spike timing. In contrast, temporal coding schemes utilize the exact timing of spikes, which has been shown to convey richer information in both biological and computational settings (Maass, 1997; Thorpe et al., 2001; VanRullen et al., 2005). However, most existing approaches still rely on activation-based learning through surrogate-gradient formulations, where gradients are propagated at every timestep regardless of spike activity. A clear

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

performance hierarchy persists among these methods: ANNs outperform activation-based SNNs, which in turn outperform event-driven approaches. Recent works (Zhu et al., 2022; 2023) have made progress in event-driven training, but continue to face several key challenges, including gradient instability, reliance on supervisory signal injection, and limited scalability across architectures.

In this work, we identify and address these limitations and propose a robust event-driven training framework that substantially narrows the performance gap with activation-based methods. Our approach mitigates gradient-related issues and improves scalability, enabling consistent performance across a range of datasets, architectures, and timestep configurations. Notably, our method achieves state-of-the-art accuracy results among event-driven approaches while operating at significantly reduced timesteps. We further provide a comparative analysis of memory efficiency, demonstrating the advantages of event-driven training over activation-based methods.

2 Related Work

SNN training is an inherently challenging problem due to the non-differentiability of spike generation and discrete nature of neuronal firing. Existing approaches can be broadly categorized into *activation-based (rate-based)* and *event-driven (time-based)* training methods.

Activation-based training. Activation-based approaches treat SNNs as recurrent neural networks and apply backpropagation through time (BPTT) with surrogate gradients to approximate the derivative of the spiking function. These methods have demonstrated strong performance on large-scale benchmarks such as CIFAR10 and ImageNet (Neftci et al., 2019; Wu et al., 2018; Fang et al., 2021; Deng et al., 2022). However, they rely heavily on rate coding and require dense temporal computation at every timestep, compromising the event-driven nature and energy efficiency of SNNs.

Event driven training. To better exploit the temporal dynamics of spikes, event-driven training methods compute gradients with respect to spike timing rather than spike activation. Early work such as SpikeProp (Bohte et al., 2002) introduced gradient-based learning using spike times. More recent approaches extend this idea to deep architectures, enabling event-driven backpropagation where gradients propagate only through spike events in a framework compatible with the asynchronous computation paradigm of SNNs (Wunderlich & Pehle, 2021; Zhu et al., 2022). The compatibility of event-driven learning with neuromorphic hardware is evidenced by implementations of event-based backpropagation on platforms such as SpiNNaker2 (Béna et al., 2025).

Theoretical advances in event-driven training. Recent work by Zhu et al. (2022) provides a theoretical foundation for event-driven training. They show that temporal gradients propagate across layers while preserving their total sum, leading to stable gradient flow in deep SNNs. Furthermore, they establish a connection between event-driven and activation-based learning by demonstrating that commonly used rate-based loss functions can be mapped to equivalent time-based formulations (Zhu et al., 2023). Their analysis highlights the importance of maintaining balanced temporal gradients and proposes improved loss functions to address gradient imbalance and achieve effective training.

3 Preliminary

3.1 Forward and Backward Equations

We consider the following spike response model for neurons in the SNN (Gerstner, 1995; Gerstner & Kistler, 2002). The membrane potential of neuron i in layer l is given by

$$u_i^{(l)}(t) = \int_{t_{i,\text{last}}^{(l)}}^{t} I_i^{(l)}(\tau)\, \epsilon(t-\tau)\, d\tau, \quad \text{where} \quad I_i^{(l)}(\tau) = \sum_j w_{ij}^{(l)}\, s_j^{(l-1)}(\tau) \tag{1}$$

where $t_{i,\text{last}}$ denotes time of last spike, $w_{ij}^{(l)}$ denotes synaptic weights, $I_i^{(l)}(\tau)$ denotes the incoming current to neuron i in layer l at time τ and $\epsilon(\cdot)$ is the post-synaptic kernel. A spike is emitted when the membrane potential crosses a threshold θ:

$$s_i^{(l)}(t) = \delta(u_i^{(l)}(t) - \theta). \tag{2}$$

Following prior work (Zhu et al., 2022), gradients are computed with respect to spike times rather than spike activations. For a spike occurring at time t_k in layer l, the gradient is expressed as

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \sum_{i,\, t_k(s_i^{(l)})} \frac{\partial L}{\partial t_k(s_i^{(l)})} \cdot \frac{\partial t_k(s_i^{(l)})}{\partial u_i^{(l)}(t_k)} \cdot \frac{\partial u_i^{(l)}(t_k)}{\partial w_{ij}^{(l)}}, \tag{3}$$

$$\frac{\partial L}{\partial t_m(s_j^{(l-1)})} = \sum_{i,\, t_k(s_i^{(l)})} \frac{\partial L}{\partial t_k(s_i^{(l)})} \cdot \frac{\partial t_k(s_i^{(l)})}{\partial u_i^{(l)}(t_k)} \cdot \frac{\partial u_i^{(l)}(t_k)}{\partial t_m(s_j^{(l-1)})}, \tag{4}$$

where L is the loss function, $t_k(s_i^{(l)})$ and $t_m(s_j^{(l-1)})$ denote the spike timings of spikes $s_i^{(l)}$ (emitted by neuron i in layer l) and $s_j^{(l-1)}$ (emitted by neuron j in layer $l-1$), respectively.

3.2 Rate based loss function for event-driven learning

The enhanced Count loss function, Count+, was introduced by Zhu et al. (2023).

$$\mathcal{L}(s, \text{target}) = 2\lambda \sum_{i=1}^{C} f\left(\int_0^T s_i(t)\, dt,\ \text{target}_i \right), \qquad f(x, \text{target}) = \begin{cases} x - \text{target}\ln(x), & x > 0, \\ 0, & x \le 0. \end{cases} \tag{5}$$

It has been shown that a rate-based loss $L(\mathbf{s})$ admits an equivalent representation in terms of spike timings (Zhu et al., 2023) as:

$$L(\mathbf{t}(\mathbf{s})) = \sum_i \sum_{t_k(s_i)} \frac{\partial L}{\partial s_i(t_k)} \cdot t_k(s_i). \tag{6}$$

This result establishes a direct connection between rate-based objectives and temporal formulations, enabling optimization in the spike-timing domain.

3.3 Gradient Sum Invariance

A key property of event-driven training is that the sum of temporal gradients remains invariant across layers (Zhu et al., 2022):

$$\sum_i \sum_k \frac{\partial L}{\partial t_k^{(l)}} = \sum_j \sum_m \frac{\partial L}{\partial t_m^{(l-1)}}. \tag{7}$$

This indicates that gradients are redistributed across spikes but their total magnitude is preserved. Consequently, training stability depends critically on how gradients are distributed across spike times, rather than their absolute scale.

3.4 Problem Analysis

While the current state-of-the-art event-driven learning approach (Zhu et al., 2023) demonstrates promising results, the following critical limitations remain.

Supervisory signals. In event-driven training, network parameters are updated only through spike events. During training, weight updates can cause an output neuron to reach a parameter configuration in which its membrane potential no longer reaches the firing threshold. Once the neuron stops firing, it receives no further parameter updates, causing it to remain trapped in this inactive state and preventing the network from correctly recognizing the corresponding class. To address this issue, (Zhu et al., 2023) modify the forward-pass spike sequence by manually setting the ground-truth output neuron to fire at a selected timestep. Specifically, they identify the latest timestep at which the neuron's membrane potential is increasing while remaining below the firing threshold by a prescribed margin, and set the neuron's output at that timestep to a spike. This manually imposed spike event provides a training signal for the ground-truth class but alters the output generated by the network's original forward dynamics.

In our analysis, the original formulation exhibits the following failure modes:

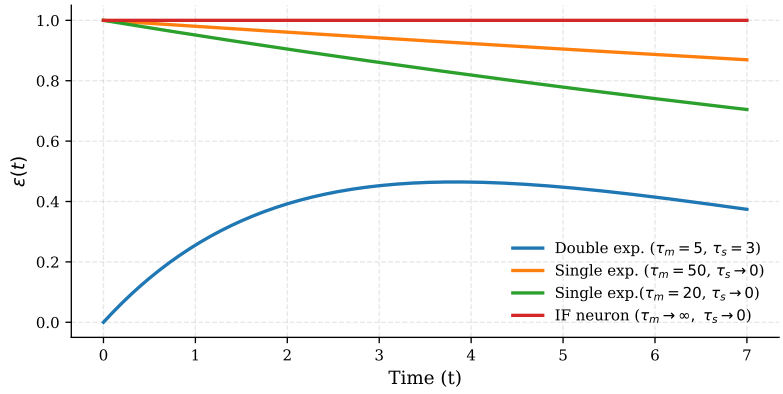


Figure 1: Different choices for kernel functions $\epsilon(t)$. The blue curve depicts the double-exponential kernel $\epsilon(t) = \frac{\tau_m}{\tau_m - \tau_s}\left(e^{-t/\tau_m} - e^{-t/\tau_s}\right)$. The orange and green curves show single-exponential kernels $\epsilon(t) = e^{-t/\tau_m}$ (obtained in the limit $\tau_s \to 0$). The red curve corresponds to the integrate-and-fire (IF) neuron $\epsilon(t) = 1$, which arises in the limiting case $\tau_s \to 0$ and $\tau_m \to \infty$.

1. **Forward/backward mismatch.** Backpropagation is performed through these artificially modified outputs rather than the network's true forward dynamics, breaking the causal consistency between the forward and backward passes. This mismatch can prevent convergence, with affected training runs exhibiting oscillatory behavior and accuracy remaining at approximately 1–2% on CIFAR100, as shown in Table 2.
2. **Heuristic supervision.** The method injects a spike only when the ground-truth neuron's membrane potential is increasing by more than the prescribed margin of 0.05 while remaining below threshold. This does not guarantee recovery: if an inactive neuron fails to meet these conditions, no corrective spike is injected. In ResNet-18 on CIFAR100, this resulted in dead output neurons, with a flat training loss and zero gradients throughout the network, as shown in Table 2.

Gradient reversal. A common model for neuronal dynamics in event-driven SNNs is the double-exponential Leaky Integrate-and-Fire (LIF) kernel:

$$\epsilon(t) = \frac{\tau_m}{\tau_m - \tau_s}\left(e^{-t/\tau_m} - e^{-t/\tau_s}\right), \tag{8}$$

which captures a fast synaptic rise governed by τ_s and a slower membrane decay governed by τ_m, as shown in Figure 1. During backpropagation (equation 4), temporal gradients depend on the membrane-potential derivative $\frac{\partial u(t)}{\partial t}$, making the kernel derivative important for determining their direction and stability:

$$\frac{\partial u_i(t_k)}{\partial t_m} \approx \sum_m w_m \frac{\partial \epsilon}{\partial t} = \sum_m w_m \left(-\frac{1}{\tau_m} e^{-(t_k - t_m)/\tau_m} + \frac{1}{\tau_s} e^{-(t_k - t_m)/\tau_s}\right), \tag{9}$$

where t_k and t_m denote the spike times at the evaluated layer and preceding input source, respectively. However, the double-exponential kernel is non-monotonic, first increasing and then decreasing after its peak. Consequently, $\frac{d\epsilon(t)}{dt}$ changes sign with spike timing, so delaying a presynaptic spike can either increase or decrease the postsynaptic response depending on its position relative to the peak. Thus, nearby synaptic interactions can produce opposing gradient directions, making optimization difficult. Prior work (Zhu et al., 2022) identifies this as the *gradient reversal problem*. To address this issue, Zhu et al. (2022) replace $\frac{d\epsilon(t)}{dt}$ in the backward computation with a continuous positive function:

$$h(t) = e^{-t/\tau}, \tag{10}$$

while retaining the double-exponential kernel in the forward pass. Although this eliminates sign changes in the backward signal, it introduces a mismatch between the forward dynamics and backward computation: the forward pass follows the non-monotonic $\epsilon(t)$, whereas the backward pass assumes monotonic decay $h(t)$. Consequently, the computed gradients no longer correspond to the derivative of the actual forward synaptic dynamics.

Scalability. Existing methods have been validated predominantly on a single architecture (VGG11). When extended to other deeper architectures such as ResNet-18 or ResNet-19, performance degrades significantly (as shown in Table 2), raising concerns about generalizability.

Higher timesteps. Current results are reported for relatively higher timesteps (e.g. $T = 16$ for CIFAR100 and $T = 12$ for CIFAR10), while activation-based methods achieve superior accuracy with much lower timesteps (e.g. $T = 4$ or $T = 6$). The memory efficiency advantage of event-driven training is meaningful only when compared under equivalent temporal budgets. Increasing the number of steps to achieve competitive accuracy undermines this advantage and leads to higher computational cost.

Violation of gradient invariance. Current formulations enforce invariance of the sums of temporal gradients across layers, as described in Section 3.3. While theoretically appealing, this condition does not constrain individual gradient terms, which may become unbounded. In practice, gradients are often clipped to ensure stability, violating this invariance assumption. We analyze this condition in Appendix Section D.2 and find that, despite these limitations, the estimated gradient sums remain bounded and consistently close to, though not exactly, unity throughout training, with no systematic evidence of temporal gradient collapse or divergence.

4 METHODS

We introduce the following modifications to improve the stability and performance of event-driven training.

4.1 OUTPUT LAYER

Prior work addresses the silent output neuron problem by manually injecting a supervisory spike into the ground-truth output neuron during the forward pass. The problems associated with this heuristic are discussed in the Section 3.4. In order to address the silent output neuron problem, we instead use a non-spiking linear readout for the output layer. Let $s_j^{(l-1)}(t)$ denote the binary spike activation of neuron j in the penultimate layer at timestep t. The readout maps the temporal spike activity directly to the output logits:

$$s_i^{(l)}(t) = \sum_j w_{ij} \left(s_j^{(l-1)}(t) \right) + b_i, \tag{11}$$

where w_{ij} and b_i are the weight and bias of output neuron i. The output layer is therefore non-spiking, keeping gradients available without requiring output spikes or supervisory signals, thereby avoiding the dead-output-neuron failure mode. All preceding layers remain spiking and event-driven. Similar non-spiking output layers have been used in prior SNN training approaches, such as TET (Deng et al., 2022).

4.2 NEURON KERNEL

Non-monotonic kernels break the consistency between temporal causality and gradient direction. In this work, we propose using a monotonic single-exponential kernel (Fig. 1) in both the forward and use its true derivative in the backward passes, as specified by

$$\epsilon(t) = e^{-t/\tau_m} \Rightarrow \frac{d\epsilon(t)}{dt} = \frac{-1}{\tau_m} e^{-t/\tau_m}. \tag{12}$$

A single exponential can be considered a double exponential in the limit $\tau_s \to 0$. The monotonic kernel ensures that $\frac{d\epsilon(t)}{dt}$ does not change sign, eliminating the reverse-gradient region at its source. Consequently, the derivative used for optimization exactly matches the forward dynamics, making temporal gradients more stable and optimization consistent with the model dynamics. We additionally evaluate an IF neuron with $\tau_m \to \infty$, which is also monotonic and therefore free of gradient reversal. IF neurons are widely studied for SNN training and ANN-to-SNN conversion due to the equivalence between their firing rates and ReLU activations (Lu & Xu, 2022; Stanojevic et al., 2023).

4.3 Normalization Layers

Prior work on event-driven SNN training has employed weight normalization (Zhu et al., 2022) and threshold normalization (Zhu et al., 2023) to stabilize training. However, in conventional deep ANNs, batch normalization (BN) (Ioffe & Szegedy, 2015; He et al., 2016) is widely used after convolutional layers to stabilize intermediate representations and improve optimization. BN has also shown benefits in activation-based SNN training (Deng et al., 2022), motivating us to investigate whether a similar normalization principle can improve the stability of event-driven training.

In an SNN, the input current determines membrane-potential dynamics and thus whether and when neurons spike. Parameter updates can therefore substantially shift the input-current distribution, causing excessive or insufficient spiking and abrupt changes in the spike manifold, i.e., the set of all spikes generated by the network. This is particularly relevant for event-driven learning, which assigns credit only to existing events and does not predict spike generation or the lack thereof. Unlike activation-based learning, which directly optimizes aggregate model activity, default event-driven learning is locally unaware of layer-wise activity, which may change drastically and unpredictably as the spike manifold shifts. Batch normalization constrains these fluctuations by normalizing input currents across batches, helping prevent mode collapse and keeping membrane-potential dynamics within a controlled range.

We therefore apply BN to the input current immediately before the spike-generation function:

$$\mathbf{s}^{(l-1)} \rightarrow \mathrm{Conv} \rightarrow \mathrm{BN} \rightarrow \mathrm{NeuronSpikeGeneration} \rightarrow \mathbf{s}^{(l)}. \tag{13}$$

Importantly, BN does not replace the event-driven credit-assignment mechanism. During the backward pass, gradients through the spiking layers are computed according to the event-driven formulation described in equation 4. These gradients are then propagated through the BN backward operation according to the chain rule, which only transforms and redistributes the temporal gradients before they are passed to preceding layers. Furthermore, the BN operation can be absorbed into the original layer for inference. We provide the corresponding weight-folding derivation and implementation details in the in Appendix Section C. Hence, BN can stabilize event-driven training while preserving the event-driven inference pathway.

5 Experiments

5.1 Training Details

Datasets. We evaluated our method on several standard image classification benchmarks, including CIFAR10, CIFAR100, DVS-CIFAR10, and Tiny-ImageNet. CIFAR10 and CIFAR100 (Krizhevsky, 2009) consist of 32×32 RGB images with 10 and 100 classes, respectively. DVS-CIFAR10 (Deng et al., 2022) is a neuromorphic version of CIFAR10, providing temporally sparse spike streams. Tiny-ImageNet (Le & Yang, 2015) is a more challenging dataset with 200 classes and greater intra-class variability. All experiments are conducted for 500 training epochs.

Architectures. We evaluate our approach in standard deep architectures adapted for spiking neural networks, including ResNet-18 (He et al., 2016), ResNet-19 (Deng et al., 2022), VGG-11 (Zhu et al., 2022), VGG-16 and VGGSNN (Deng et al., 2022). These model architectures provide a diverse set of depths and design choices, allowing us to assess the robustness of our method in different network configurations.

Input Encoding. We directly inject real-valued pixel intensities as input currents at every timestep, following Zhu et al. (2022) for a consistent comparison. We use a fixed firing threshold of unity for all spiking neurons, which remains constant throughout the training. Beyond the input layer, all subsequent layers operate on spike timings and use event-driven temporal credit assignment.

Initialization. In event-driven training, gradient flow is inherently tied to spike activity; layers that do not fire block the propagation of gradients. To address this, we initialize the network weights and adapt the Batch Normalization scaling parameter γ to maintain the desired firing rate, thereby allowing stable and effective gradient propagation. Scaling factors are determined using a binary search procedure (Zhu et al., 2022), ensuring adequate spike generation and stable training in the early stages. We use the enhanced count+ loss function to calculate the loss.

Table 1: Accuracy comparison with existing works across multiple datasets and architectures. * denotes self-implementation results and + denotes data augmentation, as described in Section 5.1. T denotes the number of timesteps. Mean ± standard deviation is reported for configurations evaluated across 3 seeds.

Dataset	Method	Gradient	Architecture	T	Acc. (%)
CIFAR10	Zhu et al. (2022)	Temporal	VGG11	12	92.10
	. Park & Yoon (2021)	Temporal	VGG16	160	93.05
	Zhu et al. (2023)	Temporal	VGG11	12	93.54
	Ours	Temporal	ResNet-19	2 / 6	93.60 ± 0.22 / 94.17 ± 0.05
	Ours	Temporal	VGG11	6	94.35 ± 0.22
	Ours$^+$	Temporal	VGG16	6	**94.57**
	TET (Deng et al., 2022)	Rate	ResNet-19	2 / 6	94.16 ± 0.03 / 94.50 ± 0.07
	ANN (Deng et al., 2022)	–	ResNet-19	1	94.97
CIFAR100	Zhu et al. (2022)	Temporal	VGG11	16	63.97
	Park & Yoon (2021)	Temporal	VGG16	160	69.66
	Zhu et al. (2023)	Temporal	VGG11	16	70.50
	Ours	Temporal	ResNet-19	2 / 6	67.57 ± 0.16 / 71.60 ± 0.12
	Ours$^+$	Temporal	ResNet-19	6	**76.9**
	TET (Deng et al., 2022)	Rate	ResNet-19	2 / 6	72.87±0.10 / 74.72±0.28
	ANN (Deng et al., 2022)	–	ResNet-19	1	75.35
DVS-CIFAR10	Zhu et al. (2023)	Temporal	VGG11	20	76.30
	Ours	Temporal	VGGSNN	10 / 20	76.57 ± 0.15 / **78.33 ± 0.38**
	TET (Deng et al., 2022)	Rate	VGGSNN	10	77.33 ± 0.21
Tiny-ImageNet	Ours	Temporal	ResNet-18	4	**48.52**
	TET (Deng et al., 2022)*	Rate	ResNet-18	4	52.40

Data augmentation. In addition to standard data normalization, we incorporate a set of preprocessing and augmentation techniques including random cropping, Cutout (DeVries & Taylor, 2017), and AutoAugment (Cubuk et al., 2019) in all static datasets. These augmentations improve both training stability and empirical performance.

5.2 Results

We compare our method with existing time-based SNN training frameworks, TET (Deng et al., 2022) (a representative activation-based SNN approach), and ANNs in Table 1. We extend event-driven SNN training to operate at as few as 2 timesteps, a setting not previously demonstrated in prior event-driven work. In the temporal regime, our method achieves state-of-the-art performance even without augmentation, outperforming previous temporal SNN methods by +1.1% in CIFAR100 (ResNet-19) with less than half the time steps (6 versus 16) and by +0.81% in CIFAR10 (VGG11) while using half the time steps (6 versus 12). In DVS-CIFAR10, we achieve a gain of +2.03% using the same timestep setting. Compared to the TET and ANN baselines, our method remains competitive. Specifically, on CIFAR10, our method (without augmentation, ResNet-19, T=6) is within 0.33% and 0.8% of the performance of TET and ANN, respectively. In DVS-CIFAR10 (T=10), our method is within 0.76% of TET. With augmentation, we achieve results comparable to those of ANN.

5.3 Ablation Studies

We perform a series of ablation experiments to quantify the contribution of each component in our framework in Table 2 across architectures, where our method shows consistent improvements. In particular, without augmentation, ResNet-18 and ResNet-19 achieve gains of +4.69% and +4.0% on CIFAR100, respectively (comparing columns B and E). With augmentation ResNet-18 and ResNet-19 achieve added gains of +11.1% and +9.2% respectively (comparing columns B and F). Furthermore, our VGG-11 model with $T = 6$ already exceeds the performance of previous work (Zhu et al., 2023) using $T = 12$ timesteps on CIFAR10, highlighting the efficiency of our approach.

Table 2: Comparison with previous work by Zhu et al. (2023) on CIFAR10 and CIFAR100 (T=6). Kernel refers to the neuronal kernel modification. BN denotes batch normalization, and Aug denotes data augmentation. The columns correspond to (A) original implementation, (B) baseline, (C) baseline + Kernel, (D) baseline + BN, (E) baseline + BN + Kernel, and (F) baseline + Kernel + BN + Aug.

Dataset	Model	**Original (A)**	**Baseline (B)**	**+ Kernel (C)**	**+ BN (D)**	**+ BN + Kernel (E)**	**+ Kernel + BN + Aug (F)**
CIFAR10	VGG11	11.71	92.30	93.50	92.45	**94.60**	94.30
	VGG16	11.29	91.00	93.10	93.13	93.70	**94.57**
	ResNet-18	13.53	88.60	90.73	88.72	92.11	**93.13**
	ResNet-19	12.83	91.60	92.91	89.16	94.17	**94.54**
CIFAR100	VGG11	1.05	68.52	69.41	68.88	69.00	**73.63**
	VGG16	1.10	64.80	69.40	67.83	70.30	**74.10**
	ResNet-18	5.08	61.41	63.80	62.10	66.10	**72.51**
	ResNet-19	1.00	67.70	69.70	65.30	71.70	**76.90**

1. **Original (Zhu et al., 2023).** The original framework reports results only for select architectures and typically at higher timesteps. When evaluated under a reduced timestep budget ($T = 6$), it did not train reliably, as discussed in Section 3.4. Although only a single spike is injected, its relative contribution increases as the timestep budget decreases because fewer genuine output spikes are available. Consequently, the artificial supervisory event has a greater influence on the forward and backward passes, exacerbating the forward/backward mismatch at low timesteps.

2. **Baseline.** Since the original configuration does not train reliably at reduced timesteps, we replace the spiking output layer with a linear layer to avoid supervisory signal injection, resulting in more stable and reliable optimization. We also employ the Adam optimizer to improve convergence. We use this configuration as the baseline for subsequent ablations.

To isolate the effects of BN and the Kernel modification, we perform individual ablations by adding BN to the Baseline (D) and the Kernel modification to the Baseline (C). We then combine both modifications (E) and subsequently add the data augmentation described in Section 5.1 (F). Batch normalization improves accuracy over the baseline across most evaluated settings, with ResNet-19 being an exception. In Table 2, we observe a consistent increase in accuracy for each cumulative modification, supporting the effectiveness of each component in our method, with only two minor deviations: a 0.3% drop on CIFAR10 with VGG11 (E vs F) on data augmentation and a 0.41% drop on CIFAR100 with VGG11 upon introducing batch normalization (C vs E). These differences are small in magnitude and do not affect the overall trends or conclusions of our study. We provide a detailed comparison between the single-exponential kernel and the IF neuron in Appendix Section B.

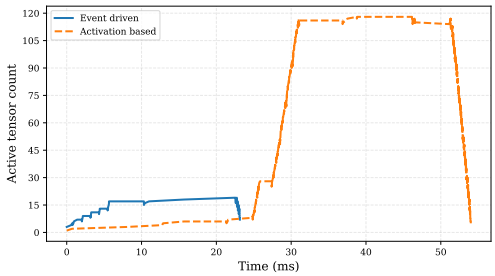
5.4 Hardware Implications

The compatibility of event-driven learning with neuromorphic hardware has already been demonstrated by implementations of event-based backpropagation on platforms such as SpiNNaker2 (Béna et al., 2025). Complementing these findings, we compare the memory utilization of activation-based and event-driven training on GPUs.

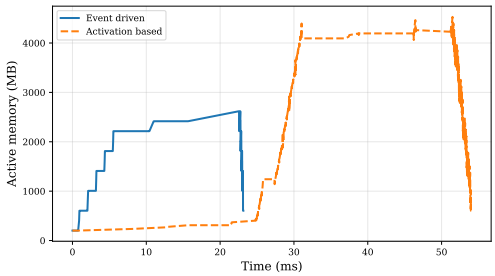
Activation-based methods such as TET model neuron dynamics as a recurrent process, requiring gradients to be propagated through the full temporal unrolling:

$$\frac{\partial L}{\partial I_t} = \sum_{k=t}^{T} \frac{\partial L}{\partial s_k} \prod_{j=t}^{k} \frac{\partial u_j}{\partial u_{j-1}}, \tag{14}$$

which introduces multiplicative dependencies across time steps, resulting in RNN-style backpropagation through time (BPTT) and memory complexity scaling linearly with time steps T. In contrast, the event-driven formulation enables a temporally local gradient decomposition. For a single neuron



(a) Shows the number of active tensors during training. Event-driven training maintains a peak of 19 compared to 118 for activation-based methods.



(b) Shows the memory trace recorded during training. Event-driven achieves a peak memory usage of 2.4 GB, compared to 4.2 GB for activation-based training.

Figure 2: Memory trace comparison on random input tensors between event-driven and activation-based learning.

(omitting indices for clarity), the gradient with respect to an input spike time t_m is the following.

$$\frac{\partial L}{\partial t_m} = \sum_{t_k} \frac{\partial L}{\partial t_k} \cdot \frac{\partial t_k}{\partial u(t_k)} \cdot \frac{\partial u(t_k)}{\partial t_m}, \tag{15}$$

where t_m and t_k denote spike times in layers l and $l+1$, respectively, and the summation is restricted to the first spike t_k that occurs after t_m. Since gradients are evaluated only at spike events, the memory cost depends primarily on the number of spikes rather than the number of timesteps. This formulation eliminates the need for multiplicative temporal chains and enables recursive gradient accumulation, avoiding full BPTT.

We evaluate memory usage by repeatedly applying the spike activation function five times on random inputs of shape $(B, C, H, W) = (128, 64, 32, 32)$ over $T = 6$ timesteps. Figure 2 shows the resulting memory trace for two methods, (a) event-driven learning (ours) and (b) activation-based learning (Deng et al., 2022). We observe that the activation-based method generates a significantly larger number of intermediate tensors, leading to a higher memory consumption as a result of its inherently recurrent computation. On the other hand, our method achieves performance comparable to activation-based approaches while reducing the peak memory usage by approximately 40%. In addition, the event-driven method runs faster due to the CUDA backend support provided by Zhu et al. (2022). To compare more practically, we repeat the same analysis on the scale of an entire model in the supplementary section. The results confirm the memory efficiency benefits of the proposed event-driven training method.

6 Conclusion

SNNs are rapidly approaching ANN performance while retaining inherent energy efficiency. Although activation-based methods have matured, they incur substantial training memory overhead. We systematically identify key failure modes and limitations in prior state-of-the-art event-driven SNN training and address them through targeted modifications. The resulting framework generalizes across datasets, architectures, and timesteps, achieving state-of-the-art performance among event-driven methods and competitive accuracy with activation-based approaches, while retaining the memory advantages of event-driven training. Our findings provide a stronger foundation for scalable and efficient event-driven SNN training.

AI use statement

In this work, we used generative AI tools solely for assistance with formatting and language editing. Generative AI was not used for theoretical formulation, development of the proposed methods, experimental design, data analysis, interpretation of results, or generation of research claims. We have not used generative AI tools for any other tasks requiring disclosure under the ICLR 2027 AI Policy, and the remaining recommended disclosure tasks are not applicable to this work. All AI-assisted text

and formatting were reviewed and verified by the authors. The authors take full responsibility for the final content of this work, including all text, claims, mathematical formulations, experimental results, and other artifacts.

Reproducibility statement

We provide the complete source code used for the main results as supplementary material, including the implementation of the proposed method, training scripts and configuration files, and scripts for generating the reported results. Environment and dependency specifications are also provided. The main paper and the Appendix describe the model, training procedure, experimental settings, and procedures needed to understand and reproduce the results. The main results reported in the paper can be reproduced using the code and configurations provided.

References

Filipp Akopyan, Jun Sawada, Andrew S. Cassidy, Alvarez-Icaza Rodrigo, John V. Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi Joon Nam, Brian Taba, Michael P. Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan L. Jackson, and Dharmenda S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. In *IEEE Transactions on Computer-Aided Design og Integrated Circuits and Systems (TCADIS)*, 2015.

Sander M Bohte, Joost N Kok, and Han La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 2002.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7B3IJMM1k_M.

Gabriel Béna, Timo Wunderlich, Mahmoud Akl, Bernhard Vogginger, Christian Mayr, and Hector Andres Gonzalez. Event-based backpropagation on the neuromorphic platform spinnaker2, 2025. URL https://arxiv.org/abs/2412.15021.

D Ekin Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. *In IEEE Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, and Sri Harsha Choday. Loihi: A neuromorphic manycore processor with on-chip learning. In *IEEE MICRO, Vol. 38*, 2018.

Song Deng, Yunzhan Li, Shiwei Zhang, and Shi Gu. Temporal efficient training of spiking neural networks via gradient re-weighting. In *International Conference on Learning Representations (ICLR)*, 2022.

Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017. URL https://arxiv.org/abs/1708.04552.

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 *International Joint Conference on Neural Networks (IJCNN)*, 2015.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, and Tiejun Huang. Deep residual learning in spiking neural networks. In *NeurIPS*, 2021.

Wulfram Gerstner. Time structure of the activity in neural network models. *Physical Review E*, 51 (1):738–758, 1995. doi: 10.1103/physreve.51.738.

Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

I C Kleppe and H P Robinson. Determining the activation time course of synaptic ampa receptors from openings of colocalized nmda receptors. *Biophysical journal*, 77(3):1418–27, 1999. ISSN 0006-3495 (Print). doi: 10.1016/S0006-3495(99)76990-0.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.

Ya Le and Xuan Yang. Tiny imagenet challenge. 2015.

Sijia Lu and Feng Xu. Linear leaky-integrate-and-fire neuron model based spiking neural networks and its mapping relationship to deep neural networks. *Frontiers in Neuroscience*, Volume 16 - 2022, 2022. ISSN 1662-453X. doi: 10.3389/fnins.2022.857513. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2022.857513.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(97)00011-7. URL https://www.sciencedirect.com/science/article/pii/S0893608097000117.

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Seongsik Park and Sungroh Yoon. Training energy-efficient deep spiking neural networks with time-to-first-spike coding, 2021. URL https://arxiv.org/abs/2106.02568.

Pranav Ramesh and Gopalakrishnan Srinivasan. PASCAL: Precise and efficient ann- snn conversion using spike accumulation and adaptive layerwise activation. In *Transactions on Machine Learning Research (TMLR)*, 2025.

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. In *Frontiers in Neuroscience*, 2017.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. In *Frontiers in Neuroscience*, 2019.

Ana Stanojevic, Stanisław Woźniak, Guillaume Bellec, Giovanni Cherubini, Angeliki Pantazi, and Wulfram Gerstner. An exact mapping from relu networks to spiking neural networks. *Neural Networks*, 168:74–88, 2023. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2023.09.011. URL https://www.sciencedirect.com/science/article/pii/S0893608023005051.

Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14(6):715–725, 2001. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(01)00083-1. URL https://www.sciencedirect.com/science/article/pii/S0893608001000831.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron Van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Rufin VanRullen, Rudy Guyonneau, and Simon J. Thorpe. Spike times make sense. *Trends in Neurosciences*, 28(1):1–4, 2005. ISSN 0166-2236. doi: https://doi.org/10.1016/j.tins.2004.10.010. URL https://www.sciencedirect.com/science/article/pii/S0166223604003546.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. In *Frontiers in Neuroscience*, 2018.

Timo C. Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):12829, Jun 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-91786-z. URL https://doi.org/10.1038/s41598-021-91786-z.

Kang You, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. SpikeZIP-TF: conversion is all you need for transformer-based SNN. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. In *NeurIPS*, 2022.

Yaoyu Zhu, Wei Fang, Tiejun Huang, Xiaodong Xie, and Zhaofei Yu. Exploring loss functions for time-based training strategy in spiking neural networks. In *NeurIPS*, 2023.

A Experimental Settings

A.1 CIFAR 10 and CIFAR100

Experiments are run using a CPU setup with 2 workers and RTX 4090 GPU. For CIFAR10 and CIFAR100 experiments, a GPU with at least 24GB memory is required. For larger-scale datasets such as Tiny-ImageNet and DVS-CIFAR10, a GPU with at least 48GB memory is needed due to increased temporal and model complexity. Unless otherwise specified, we use the Adam optimizer with cosine annealing with learning rate LR for all layers ($0.1 \times LR$ for the final continuous layer), and no weight decay. We adopt adjusted average pooling (Zhu et al., 2022), with the spike initialization set to approximately $T/4$ (e.g., 1.5 for $T = 6$). We conduct all experiments for 500 epochs with a batchsize of 128.

Table 3: Hyperparameter settings across different variants for T=6 for CIFAR10/100 datasets.

Method	**Optimizer**	τ_m	τ_s	τ_{grad}	**Spike Count** (Desired/Undesired)	**LR**
Zhu et al. (original)	AdamW	5	3	2.5	$(4, 1)$	5×10^{-4}
Zhu + Linear Last	Adam	5	3	2.5	$(4/5, 1)$	1×10^{-4}
+ Kernel Modification (IF)	Adam	∞	–	–	$(4/5, 1)$	1×10^{-4}
+ Batch Normalization (IF)	Adam	∞	–	–	$(4/5, 1)$	1×10^{-4}
+ Augmentation (single exp.)	Adam	20 / 50	–	–	$(4/5, 1)$	1×10^{-4}

A.2 Other Datasets

Table 4: Hyperparameter settings for DVS-CIFAR10 and Tiny-ImageNet datasets. Here, the leaky factor τ_m is set to 20.

Dataset	**Batch size**	**LR**	**Weight decay**	**Pooling**	**Spike Count (d,u)**
DVS-CIFAR10	16	1e-3	5×10^{-4}	max	(9/19, 1)
Tiny-ImageNet	64	1e-4	0	adjusted_avg	(3, 1)

B Ablation on leak factor - IF vs LIF

We conduct experiments with different values of τ_m in the presence of batch normalization on the CIFAR100 dataset with $T = 6$. We do not observe any consistent or conclusive trend favoring either LIF (finite τ_m) or IF ($\tau_m \to \infty$) dynamics. Instead, τ_m behaves primarily as a tunable hyperparameter, with performance varying across settings rather than following a clear pattern.

Bio-Plausibility of single exponential kernels A single-exponential kernel can be interpreted as the limiting case where synaptic dynamics are instantaneous relative to membrane dynamics, i.e.,

Table 5: Ablation on leak factor

Model	**T**	τ_m	**Accuracy (%)**
		CIFAR100	
ResNet-18	6	20	65.0
	6	50	66.1
	6	∞ (IF)	65.4
VGG16	6	20	70.59
	6	50	70.51
	6	∞ (IF)	70.3

the neuron integrates incoming spikes with negligible synaptic delay and subsequently decays with time constant τ_m. Such a separation of timescales is consistent with several biological and modeling scenarios. For example, fast excitatory synapses (e.g., AMPA-mediated currents) exhibit rise times on the order of sub-milliseconds, while membrane time constants are typically an order of magnitude larger (tens of milliseconds) (Kleppe & Robinson, 1999). In these settings, the synaptic current can be approximated as instantaneous relative to the membrane integration dynamics, justifying the use of a single-exponential kernel.

C Batch Normalization

In our work, we have inserted a batch normalization layer between the input current and the spike function, as indicated below.

$$\mathbf{s}^{(l-1)} \;\rightarrow\; \text{Conv} \;\rightarrow\; \text{BN} \;\rightarrow\; \text{Neuron} \;\rightarrow\; \mathbf{s}^{(l)} \tag{16}$$

Let $\mathbf{s}^{(l-1)} \in \{0,1\}^{B\times T\times C\times H\times W}$ denote the binary spike tensor from layer $l-1$, where B is the batch size, T the number of timesteps, and (C, H, W) are the spatial dimensions. The input current to layer l is obtained by convolving the spikes with learnable weights $\mathbf{w}^{(l)}$:

$$\mathbf{I}^{(l)} = \mathbf{w}^{(l)} * \mathbf{s}^{(l-1)}, \quad \mathbf{I}^{(l)} \in \mathbb{R}^{B\times T\times C\times H\times W}. \tag{17}$$

Batch normalization is applied over the (B, T, H, W) dimensions (for each c):

$$\hat{\mathbf{I}}^{(l)}_{b,t,c,h,w} = \frac{\mathbf{I}^{(l)}_{b,t,c,h,w} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}, \tag{18}$$

where μ_c and σ_c^2 are computed across (b, t, h, w). The normalized activations are then scaled and shifted using learnable parameters:

$$\tilde{\mathbf{I}}^{(l)}_{b,t,c,h,w} = \gamma_c \hat{\mathbf{I}}^{(l)}_{b,t,c,h,w} + \beta_c, \tag{19}$$

where γ_c and β_c are channel-wise learnable scale and shift parameters. The membrane potential is updated sequentially for T timesteps according to Eq. 1, using $\tilde{\mathbf{I}}^{(l)}$ in place of $\mathbf{I}^{(l)}$. The resulting spike tensor is $\mathbf{S}^{(l)} \in \{0,1\}^{B\times T\times C\times H\times W}$. During inference, BN can be folded into the synaptic weights and firing threshold. No constant input current is injected during inference, so neurons are still updated only when spike events occur, preserving event-driven computation and sparsity. Let the input current be

$$I = \sum_j w_j s_j, \tag{20}$$

and let Batch Normalization be defined as

$$\text{BN}(I) = \gamma \frac{I - \mu}{\sigma} + \beta. \tag{21}$$

The neuron fires when

$$\gamma \frac{I - \mu}{\sigma} + \beta \geq \theta. \tag{22}$$

Substituting $I = \sum_j w_j s_j$ and rearranging gives

$$\sum_j \frac{\gamma}{\sigma} w_j s_j \geq \theta - \beta + \frac{\gamma\mu}{\sigma}. \tag{23}$$

Defining

$$w'_j = \frac{\gamma}{\sigma} w_j, \qquad \theta' = \theta - \beta + \frac{\gamma\mu}{\sigma}, \tag{24}$$

the firing condition becomes

$$\sum_j w'_j s_j \geq \theta'. \tag{25}$$

D Empirical Analysis of Temporal Gradient Sum Invariance

A key property of event-driven learning is the temporal gradient sum invariance introduced in prior work (Zhu et al., 2022). For an output spike time t_k and its causal pre-synaptic spike times $\{t_m\}$, the temporal Jacobian satisfies

$$S_k = \sum_m \frac{\partial t_k}{\partial t_m}. \tag{26}$$

Under idealized continuous-time assumptions, this quantity is equal to unity, implying that temporal gradients are conserved during propagation and do not suffer from multiplicative collapse as in recurrent backpropagation through time.

D.1 Practical Estimation of the Gradient Sum

To empirically evaluate this property in our implementation, we estimate S_k during backpropagation using the decomposition

$$\frac{\partial t_k}{\partial t_m} = \frac{\partial t_k}{\partial u(t_k)} \cdot \frac{\partial u(t_k)}{\partial t_m}, \tag{27}$$

where $u(t_k)$ denotes the membrane potential evaluated at the output spike time. Using the implicit threshold condition $u(t_k) = \vartheta$, the temporal derivative is approximated as

$$\frac{\partial t_k}{\partial u(t_k)} \approx -\frac{1}{\dot{u}(t_k)}, \tag{28}$$

where $\dot{u}(t_k)$ is the membrane potential derivative at threshold crossing (Bohte et al., 2002). The second term,

$$\frac{\partial u(t_k)}{\partial t_m}, \tag{29}$$

is obtained from the synaptic response kernel during the backward pass. In practice, our implementation computes temporal gradients recursively in discrete time. Let $\epsilon(\Delta t)$ denote the temporal kernel evaluated at relative delay $\Delta t = t_k - t_m$. For each output spike t_k, we estimate

$$\hat{S}_k = \sum_{m \in \mathcal{P}(k)} \left(-\frac{1}{\dot{u}(t_k)} \right) w_m \epsilon'(t_k - t_m), \tag{30}$$

where $\mathcal{P}(k)$ denotes the set of causal pre-synaptic spikes contributing to the membrane potential at t_k. The quantity $\hat{S}_k$ is accumulated during the custom backward pass and averaged across neurons and mini-batches:

$$\bar{S}^{(l)} = \frac{1}{N_l} \sum_{k=1}^{N_l} \hat{S}_k, \tag{31}$$

where N_l denotes the number of spikes observed in layer l.

D.2 Observed Behavior

Figure 3 shows the evolution of the estimated gradient sum across layers during training of a VGG16 model on CIFAR. For visibility, only the first 8 layers have been depicted. Importantly, the estimated gradient sums remain consistently bounded and close to 1 throughout training.

Several implementation factors contribute to deviations from the ideal invariant:

1. discrete-time approximation of continuous spike dynamics,
2. truncation of the temporal response kernel,
3. clamping of $\dot{u}(t_k)^{-1}$ for numerical stability,
4. normalization-induced perturbations to membrane dynamics.

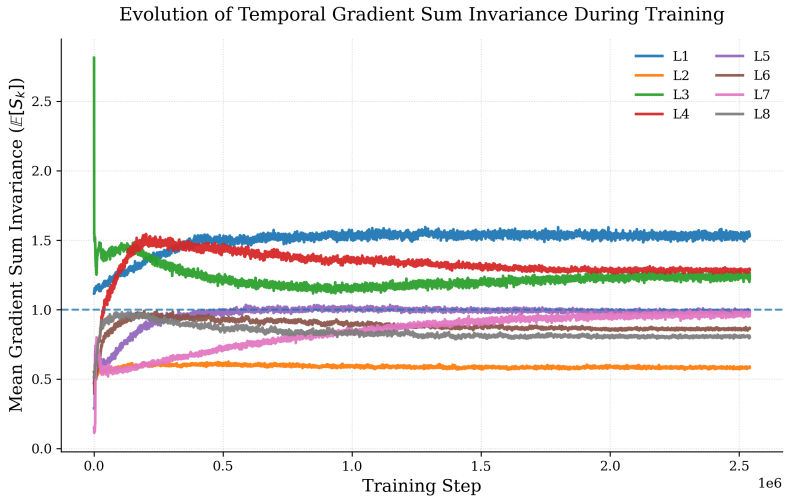


Figure 3: Estimated temporal gradient means in different layers.

Table 6: Memory comparison (in GB) of event-driven vs activation-based training methods.

Dataset	Batch Size	Model	T	Activation-based	Event-driven
CIFAR10	128	VGG16	6	15.29	**10.49**
CIFAR10	128	ResNet-18	6	20.70	**9.84**
DVS-CIFAR10	128	VGGSNN	10	32.32	**25.54**

These effects introduce a small layer-dependent bias in the empirical estimate of S_k. In particular, early layers tend to underestimate the invariant due to truncated temporal contributions, while deeper layers can mildly overestimate it when membrane derivatives near threshold become small. Despite these deviations, we do not observe systematic collapse or divergence of temporal gradients. Instead, the estimated gradient sums remain converge to a consistent value near unity, indicating that temporal gradient propagation remains well-conditioned during training.

D.3 Implications for Stable Event-Driven Learning

Unlike recurrent backpropagation through time, where gradients are propagated through products of temporal Jacobians,

$$\prod_t \frac{\partial u_t}{\partial u_{t-1}}, \tag{32}$$

event-driven learning propagates gradients through additive temporal interactions governed by equation 26. As a result, optimization is substantially less susceptible to exponential vanishing or exploding gradients. Our empirical observations support this interpretation: although the estimated invariant is not exactly satisfied in practice, temporal gradients remain stable throughout training.

E Memory Efficiency on Model Scale

To compare more practically, we repeated the same analysis on the scale of an entire model. The results, described in Table 6, confirm the memory efficiency benefits of the proposed event-driven training method

F Societal Impacts and Limitations

The proposed work on efficient event-driven spiking neural networks may positively contribute to the development of energy-efficient machine learning systems and low-power edge AI hardware. Since

this work focuses on foundational training methodologies for spiking neural networks, we do not identify any direct negative societal impacts beyond those generally associated with machine learning technologies.

A primary limitation of the current work is, while the proposed approach was successfully evaluated up to the Tiny-ImageNet dataset, we were unable to train and evaluate the method on ImageNet-scale benchmarks within available computational resources. Extending the method to larger-scale datasets and architectures remains an important direction for future research. Other promising future work includes research on time-domain loss formulations like phase coding where event-driven learning is potentially better suited.