Stabilizing Direct Training of Spiking Neural Networks: Membrane Potential Initialization and Threshold-robust Surrogate Gradient

Hyunho Kook Byeongho Yu Jeong Min Oh Eunhyeok Park
Pohang University of Science and Technology (POSTECH), Republic of Korea
{kookhh0827, bhyu418, ojm010130, eh.park}@postech.ac.kr

Abstract

Recent advancements in the direct training of Spiking Neural Networks (SNNs) have demonstrated high-quality outputs even at early timesteps, paving the way for novel energy-efficient AI paradigms. However, the inherent non-linearity and temporal dependencies in SNNs introduce persistent challenges, such as temporal covariate shift (TCS) and unstable gradient flow with learnable neuron thresholds. In this paper, we present two key innovations: ***MP-Init (Membrane Potential Initialization)*** *and* ***TrSG (Threshold-robust Surrogate Gradient)****. MP-Init addresses TCS by aligning the initial membrane potential with its stationary distribution, while TrSG stabilizes gradient flow with respect to threshold voltage during training. Extensive experiments validate our approach, achieving state-of-the-art accuracy on both static and dynamic image datasets. The code is available at: https://github.com/kookhh0827/SNN-MP-Init-TRSG*

1. Introduction

Spiking Neural Networks (SNNs) transmit information through discrete spikes over time, enabling efficient, event-driven processing [2, 4, 14, 34, 35, 37], especially on neuromorphic hardware [1, 6, 33]. Recent advances in direct training [43, 44] have significantly improved SNN accuracy with few timesteps, enhancing efficiency. Yet, SNNs still lag behind deep neural networks, and their complex dynamics often hinder performance on large-scale datasets [20, 54]. Thus, further innovations are crucial to ensure reliability and accuracy for real-world adoption.

One of the primary challenges in SNN training is **Temporal Covariate Shift (TCS)** [27], where internal covariate shifts in activations accumulate over time. While TEBN [11] and TAB [26] attempt to address this issue by introducing timestep-specific normalization parameters, they disrupt parameter homogeneity across time, increasing both model complexity and computational cost. More importantly, our analysis reveals that these methods fail to mitigate TCS in other critical components of SNNs, particularly the membrane potential. Since the membrane potential directly determines neuronal output, neglecting TCS in this component can lead to significant accuracy degradation.

The second challenge lies in **training the internal parameters** [16, 35, 42] of Leaky Integrate-and-Fire (LIF) neurons, which are the threshold voltage (V_{thr}) and leakage term (τ). Despite their crucial role in LIF neuron behavior, they are typically fixed or updated via gradient descent under strict constraints. We reveal that existing gradient flow heavily depends on the scale of V_{thr}, and unconstrained updates during training could lead to significant instability. Surprisingly, none of the existing studies has explicitly recognized this issue.

In this work, we introduce a comprehensive approach to overcoming these limitations. Our first contribution, **MP-Init (Membrane Potential Initialization)**, stems from our analysis of the root cause of TCS in SNNs. By addressing TCS in membrane dynamics, MP-Init ensures consistent neuron behavior across timesteps without using timestep-dependent parameters. Our second contribution, **TrSG (Threshold-Robust Surrogate Gradient)**, is a refined version of surrogate gradient that stabilizes the gradient flow with respect to the threshold V_{thr}. This modification allows for stable gradient-based updates of all internal parameters, leading to superior convergence.

We provide a rigorous mathematical analysis of our methods and demonstrate through extensive experiments that they significantly improve SNN performance. Our approach enables effective SNN training, achieving state-of-the-art results on both static and dynamic datasets.

2. Related Work

2.1. Addressing Temporal Covariate Shift

The TCS problem was first identified in BNTT [27] and later addressed by methods such as TEBN [11] and TAB [26], which adapt normalization layers on a per-timestep basis. While these methods alleviate TCS in ac-

tivations, they increase model complexity with additional parameters and, more critically, overlook the persistent shift in membrane potential, a primary source of TCS during inference (see Section 4.1).

IMP+LTS [39] proposed training the initial membrane potential for every neuron to enhance representational power. However, it does not directly address TCS and incurs a large parameter overhead scaling with the number of neurons $O(N)$. By contrast, our approach initializes membrane potentials at the layer level $O(L)$ ($L \ll N$), directly targeting TCS while remaining lightweight.

More recently, MPS [10] reinterpreted SNNs from an ensemble-learning perspective, arguing that excessive differences in membrane potential distributions across timesteps destabilize subnetworks. Their smoothing and guidance strategies improve temporal consistency and yield accuracy gains. However, these methods modify the LIF update rule itself and require additional smoothing coefficients and guidance losses, meaning both training and inference diverge from the standard LIF neurons. In contrast, our approach preserves the original inference pipeline: MP-Init adds negligible overhead during training while inference remains compatible with neuromorphic hardware [6].

2.2. Learning in Spiking Neural Networks

Surrogate Gradient (SG) descent enables end-to-end training of SNNs by approximating the non-differentiable firing process [29, 43, 44]. Some studies primarily focus on the impact of hyperparameters and the shape of surrogate derivatives [31, 51], while recent advancements introduce differentiable surrogate gradient functions [31] and Temporal Efficient Training (TET) losses [8]. Other approaches regulate membrane potential by adding loss terms [19, 21] or applying batch normalization [22]. Meanwhile, methods that modify the dynamics of LIF neurons [18, 24, 48] have been explored, though they could be incompatible with existing neuromorphic hardware [20, 25].

Recent studies have also explored training internal parameters, such as the threshold voltage and leakage term, via gradient descent [16, 35, 42]. For instance, PLIF [16] optimizes the leakage term, LTMD [42] adjusts the threshold voltage, and DIET-SNN [35] learns both. However, these works provide limited analysis of gradient flow related to these learnable parameters, leaving questions about training stability unanswered. In contrast, we find that when the threshold voltage becomes trainable, surrogate gradient methods can exhibit instability, leading to misaligned, exploding, or vanishing gradients, as discussed in Section 5.2. To overcome these issues, we propose a novel SG formulation that remains stable when learning the threshold voltage, ensuring robust convergence.

3. Preliminaries

3.1. The LIF Neuron Model

LIF model [13] is a well-known spiking neural model that emulates key behaviors of biological neurons, including firing spikes and decaying membrane potential. The LIF neuron's behavior in discrete steps $t \in \mathbb{N}$ is described as the following equations:

$$M^l[t] = \left(1 - \frac{1}{\tau^l}\right) U^l[t-1] + \frac{1}{\tau^l} I^l_{\text{in}}[t], \tag{1}$$

$$S^l[t] = \mathcal{H}\left(M^l[t] - V^l_{\text{thr}}\right), \tag{2}$$

$$U^l[t] = \begin{cases} M^l[t] - V^l_{\text{thr}} \cdot S^l[t], & \text{(soft reset)} \\ M^l[t] \cdot (1 - S^l[t]). & \text{(hard reset)} \end{cases} \tag{3}$$

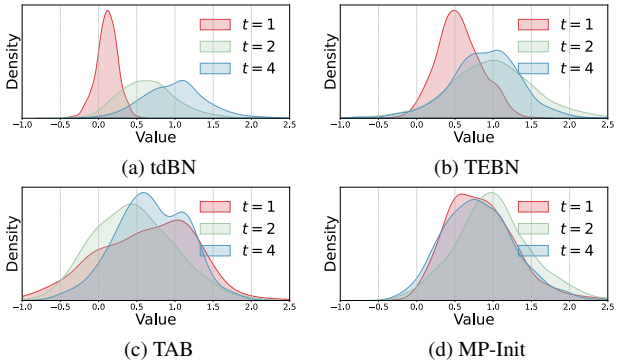
In these equations, l represents the layer index within the network, and $\mathcal{H}$ denotes the Heaviside step function. The presynaptic input current is defined as $I^l_{\text{in}}[t] = W^l_{l-1} O^{l-1}[t]$, where W^l_{l-1} is the synaptic weight matrix connecting layer $l-1$ to layer l, and $O^{l-1}[t]$ represents the output of the previous layer. In direct training, this output typically corresponds to the binary spike itself, i.e., $O^l[t] = S^l[t]$ [43]. The leakage constant τ^l can be absorbed into the synaptic weights [13], leading to a rescaling $W^l_{l-1} := \frac{1}{\tau^l} W^l_{l-1}$ in Eq. (1). The choice between a soft and hard reset in Eq. 3 influences the membrane potential dynamics after firing. Under a soft reset [38], the membrane potential decreases by the threshold voltage V^l_{thr}. In contrast, a hard reset [43] resets the membrane potential to zero. Notably, our two novel ideas are effective in both cases, leading to improved quality.

3.2. Exclusion of Silent Neurons

In our analysis of TCS, we focus only on *active neurons*, i.e., those that fire at least once during each simulation. Previous work [49] shows that a large portion of neurons in SNNs remain subthreshold throughout a simulation (i.e., *silent*). They are irrelevant to our analysis, as they neither contribute to information propagation nor exhibit statistical properties similar to those of active neurons. Consequently, we exclude silent neurons when analyzing the statistics of the membrane potential.

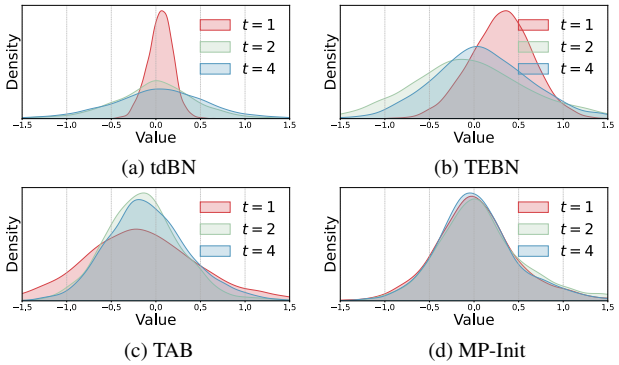
4. Membrane Potential Initialization

TCS [11, 26, 27] limits the performance of SNNs during inference, particularly when batch normalization (BN) is employed. Despite BN's capability to normalize activations during training, the fixed normalization statistics during inference result in activation drift across timesteps, causing accuracy degradation. Previous methods [11, 26] have tried



(a) tdBN (b) TEBN (c) TAB (d) MP-Init

Figure 1. Membrane potential distribution of the third layer's first spiking layer of ResNet-19 on CIFAR100



(a) tdBN (b) TEBN (c) TAB (d) MP-Init

Figure 2. Activation distribution of the convolutional layer after the third layer's first spiking layer of ResNet-19 on CIFAR100

to address this issue through timestep-dependent normalization, but these approaches remain superficial as they fail to resolve the underlying cause: drift in the membrane potential of LIF neurons.

To tackle this issue, we propose **MP-Init (Membrane Potential Initialization)**, an efficient solution that directly targets TCS in the membrane potential. MP-Init ensures stable activation distributions throughout inference by initializing the membrane potential of spiking neurons in each layer to match their stationary distribution. This eliminates the need for timestep-dependent normalization and enhances final performance by aligning training and inference statistics.

4.1. Root Cause of TCS

We first identify that TCS in activations fundamentally originates from TCS in membrane potentials. According to Eq. (2), spike outputs depend directly on membrane potentials. If membrane potentials significantly drift over timesteps, the resulting activation distribution shifts correspondingly.

We empirically visualize the membrane potential at each timestep to validate this claim. As illustrated in Fig. 1, the membrane potential in tdBN, which does not account for the TCS problem, exhibits a clear temporal drift, leading to TCS in activation (Fig. 2). While TEBN and TAB attempt to stabilize the distribution by timestep-dependent normalizations, the membrane potential itself continues to undergo transient drift, preserving TCS in activation.

4.2. Convergence of Membrane Potential

While we have empirically demonstrated the existence of TCS in the membrane potential, a theoretical perspective can provide a deeper understanding. In this section, we present the core principles explaining why the membrane potential drifts over time.

We prove that the layer-wise distribution of membrane potential converges to a distribution possibly differing from its initial state as time progresses. This was proven on continuous-time LIF models with stable input statistics [12]. Here, we extend this idea to the discrete-time model and show it has the same property. Let's start with the following assumption:

Assumption 1. Assume $\{I_{\text{in}}[t]\}_{t\geq 1}$ be a sequence drawn from an i.i.d. distribution. Furthermore, suppose there exist real constants $u^- < u^+$ where the membrane potential $U[t]$ of active neurons within each layer remains within $[u^-, u^+]$ for all t.

This i.i.d. assumption is reasonable, as inputs $I_{\text{in}}[t]_{t\geq 1}$ are repeatedly drawn from a dataset or a normalized layer. Although soft reset LIF dynamics do not strictly prevent unbounded growth, well-trained SNNs rarely exhibit it in active neurons. The empirical validation of this assumption is provided in the supplementary material (Sec. 10.2). Based on Assumption 1, we present the following theorem:

Theorem 1. Under Assumption 1, the sequence $\{U[t]\}_{t\geq 0}$ forms a Markov chain with transition kernel P. This chain satisfies Doeblin's minorization condition, ensuring the existence of a unique stationary distribution π. Moreover, the Total Variation (TV) distance between $U[t]$ and π decreases exponentially.

Detailed proof is provided in supplementary material (Sec. 10.4). This theorem indicates that TCS in the membrane potential is inevitable when the initial membrane potential differs from its stationary distribution. Traditionally, the membrane potential is initialized to zero, i.e., $U[0] = 0$, but this zero initialization is a primary contributor to TCS. Therefore, to mitigate TCS, we must reduce the difference between the initial membrane potential and its stationary distribution.

4.3. Membrane Potential Initialization

Theorem 1 implies that eliminating the mismatch between the initial state and this stationary distribution is crucial. However, directly computing π is intractable; hence, we propose a more practical approach to initialize the mem-

brane potential with a **per-layer constant** that approximates the mean of π, as supported by the following lemma:

Lemma 1. Among all constants $c \in \mathbb{R}$, $c = E[\pi]$ minimizes expected square difference $E[(\pi - c)^2]$.

The proof is provided in supplementary material (Sec. 10.5). This lemma suggests that initializing the membrane potential to the mean of its stationary distribution is optimal for eliminating drift and minimizing variance. However, the next challenge lies in approximating $E[\pi]$.

One simple solution is leveraging Theorem 1, which states that a membrane potential distribution exponentially converges to π over time. Therefore, we approximate π using the membrane potential at the final timestep ($U[T]$), which is the closest to π. Based on this, we estimate the per-layer statistics of $E[\pi]$ using $E[U[T]]$.

To implement this efficiently, we compute the average membrane potential $U[T]$ at the final timestep T for each layer during training. This average is then used to update a running mean r, which is initialized to zero at the beginning of training. Before feeding each batch into the network (during both training and inference), we initialize the membrane potential of every layer to this running mean, i.e., $U[0] = r$. This technique is called **MP-Init**, which requires minimal extra memory or computation.

As illustrated in Figs. 1 and 2, MP-Init effectively resolves both TCS in membrane potential and activation, resulting in consistent distributions over time. Importantly, MP-Init is straightforward to implement and does not require modifications to BN layers or involve complex training pipelines, unlike methods such as TEBN [11] or TAB [26]. This simplicity and ease of implementation significantly enhance the usability of SNNs across various applications without altering existing recipes.

5. Threshold-robust Surrogate Gradient

Surrogate Gradient (SG) [43] enables gradient-based optimization of SNNs by replacing the non-differentiable spiking function with a differentiable surrogate near the threshold. This allows effective gradient descent in SNNs, often improving performance in early timesteps. However, our analysis shows that SG used for internal neuronal parameters [16, 35, 42], particularly V_{thr}, makes training highly sensitive to the *scale* of V_{thr}. While careful tuning can yield strong accuracy, this dependency exposes a fundamental limitation of current SG-based training.

In this section, we identify, *to our knowledge for the first time*, that prior SG designs exhibit strong sensitivity of gradient flow to V_{thr}. To address this, we introduce **TrSG (Threshold-robust SG)**, which maintains stable gradients regardless of the threshold value. TrSG is simple to implement, adds negligible overhead, and is compatible with a wide range of surrogate functions.

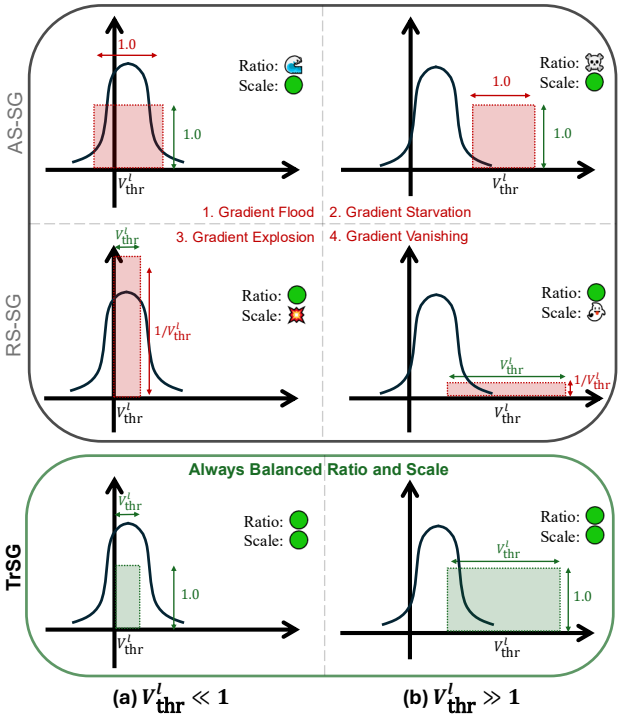


Figure 3. **AS-SG vs. RS-SG vs. TrSG with a rectangular surrogate gradient ($\gamma = 1$).** The curve depicts the membrane potential distribution, while the colored boxes represent the surrogate gradient region: the horizontal span indicates the active gradient window, and the vertical extent indicates its magnitude. Panels (a) and (b) compare the behaviors when the threshold is small ($V^l_{\text{thr}} \ll 1$) and large ($V^l_{\text{thr}} \gg 1$), respectively. *AS-SG* uses a fixed window of width γ, which leads to *gradient flood* or *starvation*. *RS-SG* normalizes the window by V^l_{thr} but scales the magnitude as $1/V^l_{\text{thr}}$, causing *explosion* or *vanishing*. **TrSG** multiplies the threshold forward during training, canceling the $1/V^l_{\text{thr}}$ factor and keeping both window and magnitude balanced across thresholds.

5.1. Background: Surrogate Gradient Descent

As in Eq. 2, an LIF neuron emits a spike when the membrane potential crosses the threshold. Let $H(x)$ denote the Heaviside step. Then the spike can be written as

$$S^l[t] \;=\; H\big(x(M^l[t], V^l_{\text{thr}})\big),$$

where the *argument* $x(\cdot)$ encodes how the membrane potential $M^l[t]$ and the threshold V^l_{thr} are combined (e.g., $x = M^l[t] - V^l_{\text{thr}}$). Because $H(\cdot)$ is non-differentiable and $H'(\cdot)$ is a Dirac delta function, SG training replaces H with a smooth surrogate f whose derivative locally mimics H':

$$H'(x) \;\approx\; f'(x) \quad \text{near } x = 0.$$

Applying the chain rule to the spike yields the working gradient used in training,

$$\frac{\partial S^l[t]}{\partial M^l[t]} \;=\; H'(x)\,\frac{\partial x}{\partial M^l[t]} \;\approx\; f'(x)\,\frac{\partial x}{\partial M^l[t]}, \qquad (4)$$

so the *choice of* $x(\cdot)$ is crucial—especially when V_{thr}^l is learnable—because it determines both the active region of nonzero gradients and the scaling of their magnitudes.

Upon reviewing the literature, SG-based methods fall into two families according to how they define $x(M^l[t], V_{\text{thr}}^l)$: *Absolute-Scale SG (AS-SG)* and *Relative-Scale SG (RS-SG)*. This distinction becomes significant once the threshold is trained.

5.2. AS-SG vs. RS-SG

To make the scaling effects explicit, we analyze a representative surrogate with a rectangular derivative

$$f'(x) \;=\; \frac{1}{\gamma}\,\mathbb{1}\Big(|x| < \tfrac{\gamma}{2}\Big),$$

where $\gamma > 0$ controls the width of the region where gradients flow. The phenomena we highlight are *not* specific to this choice; any surrogate exhibits the same issues.

For a better illustration, Fig. 3 visualizes a membrane potential distribution under two different threshold scales: small ($V_{\text{thr}}^l \ll 1$) and large ($V_{\text{thr}}^l \gg 1$).

1. AS-SG: Absolute-Scale Surrogate Gradient. Many works [8, 16, 42, 52] take

$$x \;=\; M^l[t] - V_{\text{thr}}^l, \tag{5}$$

which gives

$$f'\big(M^l[t] - V_{\text{thr}}^l\big) \;=\; \frac{1}{\gamma}\,\mathbb{1}\Big(\big|M^l[t] - V_{\text{thr}}^l\big| < \tfrac{\gamma}{2}\Big), \tag{6}$$

and thus

$$\frac{\partial S^l[t]}{\partial M^l[t]} \;\approx\; f'\big(M^l[t] - V_{\text{thr}}^l\big). \tag{7}$$

Here the active window $\{M^l[t] \in (V_{\text{thr}}^l - \frac{\gamma}{2},\, V_{\text{thr}}^l + \frac{\gamma}{2})\}$ has a *fixed* width γ, and the magnitude is fixed, independent of V_{thr}^l. When the V_{thr}^l is small, the window becomes excessively wide relative to the membrane potential spread, so gradients keep flowing even far from the true firing boundary (*gradient flood*). Conversely, when the V_{thr}^l is large, the fixed-width window covers only a tiny fraction of the distribution, and most gradients vanish (*gradient starvation*).

2. RS-SG: Relative-Scale Surrogate Gradient. Other works normalize by the threshold; either by $\gamma = V_{\text{thr}}^l$ [32] or by defining $x = \frac{M^l[t]}{V_{\text{thr}}^l} - 1$ as in [35]. In this case,

$$f'\Big(\tfrac{M^l[t]}{V_{\text{thr}}^l} - 1\Big) \;=\; \frac{1}{\gamma}\,\mathbb{1}\Big(\Big|\tfrac{M^l[t]}{V_{\text{thr}}^l} - 1\Big| < \tfrac{\gamma}{2}\Big), \tag{8}$$

so the active window $\{M^l[t] \in ((1-\frac{\gamma}{2})V_{\text{thr}}^l,\, (1+\frac{\gamma}{2})V_{\text{thr}}^l)\}$ scales with V_{thr}^l. However, the chain rule introduces a sensitivity of the gradient magnitude to the threshold:

$$\frac{\partial S^l[t]}{\partial M^l[t]} \;\approx\; \frac{1}{V_{\text{thr}}^l}\, f'\Big(\tfrac{M^l[t]}{V_{\text{thr}}^l} - 1\Big). \tag{9}$$

Thus, small thresholds amplify gradients excessively (*gradient explosion*), whereas large thresholds suppress them severely (*gradient vanishing*).

5.3. Threshold-robust Surrogate Gradient

We propose **TrSG**, a simple modification that preserves the desirable *relative* window while canceling the problematic $1/V_{\text{thr}}^l$ factor in the gradient. During training we (i) keep the relative-scale argument,

$$x \;=\; \frac{M^l[t]}{V_{\text{thr}}^l} - 1,$$

and (ii) multiply the spike by the threshold on the forward path:

$$O^l[t] \;=\; V_{\text{thr}}^l \cdot S^l[t]. \tag{10}$$

Then,

$$\frac{\partial O^l[t]}{\partial M^l[t]} \;=\; V_{\text{thr}}^l\,\frac{\partial S^l[t]}{\partial M^l[t]} \;\approx\; V_{\text{thr}}^l \cdot \Big(\frac{1}{V_{\text{thr}}^l} f'(x)\Big) \;=\; f'(x), \tag{11}$$

so the gradient magnitude becomes *threshold-invariant* while the active window still adapts with V_{thr}^l.

Inference remains binary. The scaling is training-only: at test time we revert to $S^l[t] \in \{0, 1\}$ and absorb V_{thr}^l into the next layer via a simple rescaling $W_l^{l+1} \leftarrow V_{\text{thr}}^l\, W_l^{l+1}$.

In summary, **TrSG** unifies a relative-scale argument with threshold multiplication on the forward path, ensuring stable gradients across thresholds. This design eliminates the flood/starvation issue of AS-SG and the vanish/explode problem of RS-SG, as further demonstrated in Sec. 11.1.

6. Experiment

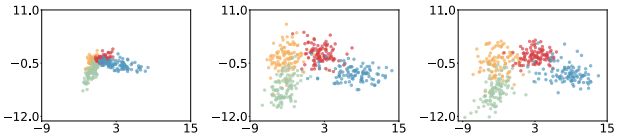
We evaluate **MP-Init** and **TrSG** on both static and event-based vision benchmarks, including CIFAR10, CIFAR100 [28], ImageNet [7], and DVS-CIFAR10 [30], using diverse backbones such as ResNet-19 [52], a 7-layer CNN [11, 26], ResNet-34 [23], and SEW-ResNet-34 [15]. Following tdBN [52], batch normalization is applied across both temporal and batch dimensions. All baselines are trained under their original configurations, and we strictly follow their reported preprocessing pipelines for fairness. We avoid strong augmentations such as AutoAugment [5] or Cutout [9] to ensure that observed gains come solely from our proposed methods. We run three independent trials and report the results as $\mathbf{mean} \pm \mathbf{std}$ across runs. Detailed descriptions of architectures, preprocessing, and hyperparameters are provided in the supplementary material (Sec. 9).

6.1. Analysis of MP-Init

Effectiveness of MP-Init We begin by assessing whether MP-Init effectively enhances the final performance of SNNs. To do this, we train ResNet-19 on CIFAR100 and

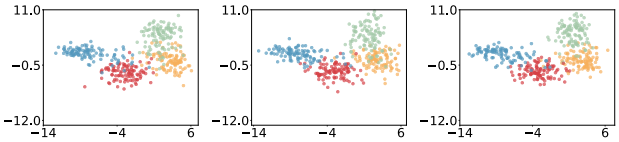
Dataset	Method	Timestep	Accuracy (%)
CIFAR100	tdBN	4	75.55 ± 0.06
	+ MP-Init		**76.09 ± 0.04**
	TEBN	4	75.96 ± 0.68
	+ MP-Init		**76.45 ± 0.24**
	TAB	4	76.25 ± 0.49
	+ MP-Init		**77.24 ± 0.12**
DVS-CIFAR10	tdBN	10	76.60 ± 0.20
	+ MP-Init		**77.37 ± 0.38**
	TEBN	10	75.17 ± 0.50
	+ MP-Init		**76.70 ± 0.78**
	TAB	4	76.43 ± 0.25
	+ MP-Init		**76.93 ± 0.38**

Table 1. Validation of performance improvements from MP-Init using ResNet-19 on CIFAR100 and DVS-CIFAR10 datasets.



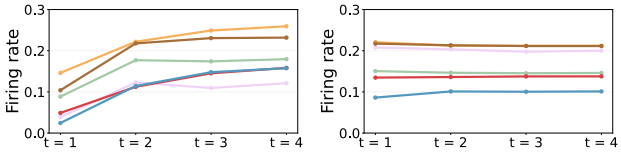
(a) t = 1 (b) t = 2 (c) t = 4

Figure 4. PCA of logits without MP-Init. Class clusters stay mixed in early timesteps, showing strong temporal covariate shift.



(a) t = 1 (b) t = 2 (c) t = 4

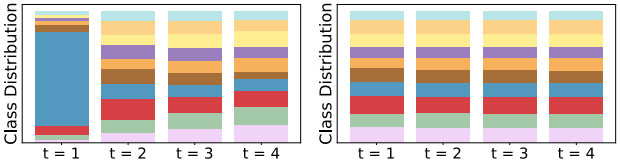
Figure 5. PCA of logits with MP-Init. Clear class separation emerges from t=1, indicating reduced temporal drift.



(a) w/o MP-Init (b) w/ MP-Init

Figure 6. Firing rate of all neurons at each timestep in arbitrarily selected 6 layers of ResNet-19 on CIFAR10.

DVS-CIFAR10 datasets and measure accuracy with and without MP-Init, as shown in Table 1. We observe consistent performance improvements with MP-Init, not only in tdBN, which does not account for TCS, but also in TCS-aware methods such as TEBN and TAB. This consistency suggests that previous approaches may only partially address the TCS issue, whereas MP-Init offers a more fundamental solution. Additional analysis on how MP-Init affects training is provided in Sec. 10.1.



(a) w/o MP-init (b) w/ MP-init

Figure 7. Class prediction distributions over timesteps in ResNet-19 on CIFAR10. Without MP-Init, early steps collapse to a few classes; MP-Init yields balanced outputs from t=1.

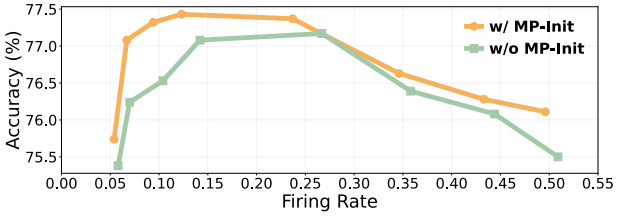
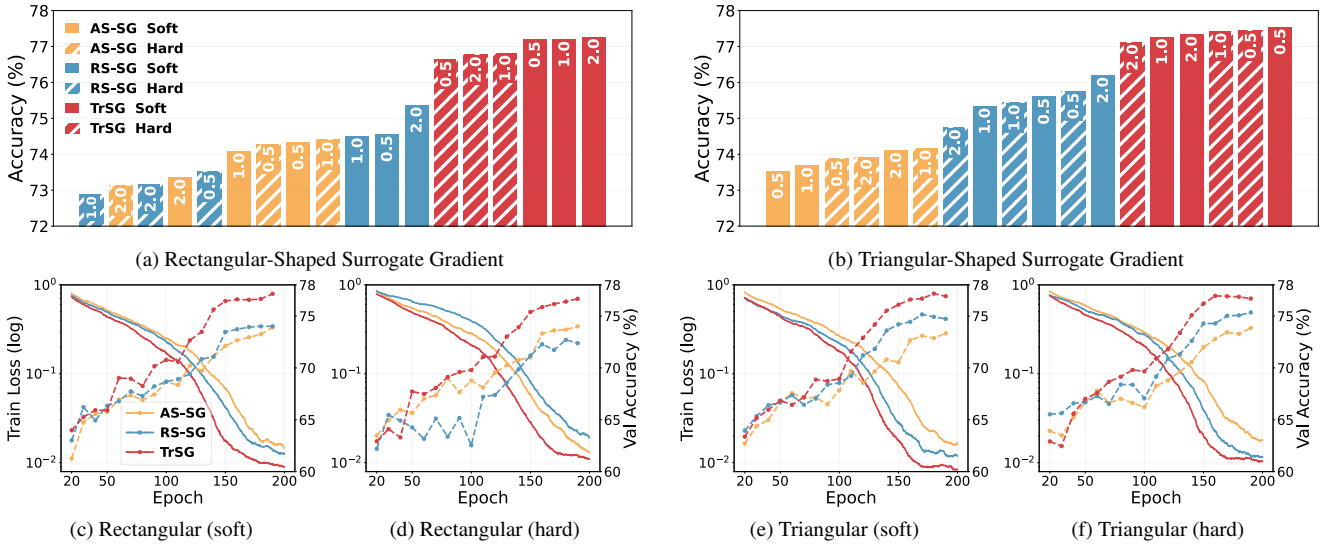


Figure 8. Accuracy vs. firing rate. MP-Init consistently achieves higher accuracy at matched sparsity levels, forming a better Pareto frontier.

Impact of TCS and Role of MP-Init To visualize TCS effects, we train ResNet-19 on CIFAR10 and project the final logits of four randomly selected classes onto two principal components (Figs. 4 and 5). Without MP-Init, these classes remain entangled in early steps and only gradually separate, reflecting TCS-induced output bias. With MP-Init, class separation emerges from the first timestep. This aligns with the firing-rate dynamics in Fig. 6: MP-Init stabilizes neuron activations across timesteps, whereas the baseline accumulates activity over time. Class-distribution analysis (Fig. 7) further confirms this: networks without MP-Init produce heavily skewed class predictions at early timesteps, often collapsing to a single dominant class. This illustrates the impact of temporal covariate shift on output stability. In contrast, MP-Init yields balanced class distributions from the first timestep, confirming that it mitigates early-step bias and enables reliable low-latency inference.

Firing Rate vs. Accuracy A natural concern after Fig. 6 is whether MP-Init simply increases firing and thereby harms efficiency. To test this, we apply a regularization loss [45] for ResNet-19 on CIFAR100 that directly controls spike sparsity and evaluate performance across a wide range of firing rates (Fig. 8). The results show that MP-Init does not rely on excessive spiking: it consistently forms a Pareto frontier, achieving higher accuracy than the baseline at the same or even lower firing rates. Remarkably, at a firing rate as low as 0.06, MP-Init still outperforms all baseline settings. This indicates that MP-Init not only enhances accuracy but also enables more energy-efficient SNN operation.



(a) Rectangular-Shaped Surrogate Gradient

(b) Triangular-Shaped Surrogate Gradient

(c) Rectangular (soft)

(d) Rectangular (hard)

(e) Triangular (soft)

(f) Triangular (hard)

Figure 9. **Comparison of SG approaches under different surrogate function shapes (rectangular and triangular), reset types (soft and hard). (Top Row)** shows final accuracy on CIFAR100 (ResNet-19, timestep = 4) at various initial $V_{\text{thr}} \in \{0.5, 1.0, 2.0\}$ and an initial $\tau = 2.0$. **(Bottom Row)** plots the training loss and validation accuracy curves with initial $\tau = 2.0$ and $V_{\text{thr}} = 1.0$. In all experiments, **TrSG** consistently outperforms the AS-SG and RS-SG methods, demonstrating robustness to different initial values.

6.2. Analysis of TrSG

In this section, we compare three distinct ways of defining the SG: **AS-SG** (Absolute-Scale), **RS-SG** (Relative-Scale), and our proposed **TrSG** when training both V_{thr} and τ. To ensure fair evaluation, we use the same training recipes across all methods, as explained in the supplementary material (Sec. 9.3). To assess initialization sensitivity, we also provide results with different initial values of V_{thr} and τ.

Effectiveness of TrSG. We first evaluate three SG approaches using both soft and hard reset LIF neurons, along with two widely adopted SG shapes for $f'_{sr}(x)$: **(i) Rectangular** [35, 42, 52]: $f'_{sr}(x) = \frac{1}{\gamma}\mathbb{1}\left(|x| < \frac{\gamma}{2}\right)$, **(ii) Triangular** [8, 26, 32]: $f'_{sr}(x) = \frac{1}{\gamma^2}\max\left(0, \gamma - |x|\right)$. We train each configuration to learn both V_{thr} and τ under varying initial V_{thr} values. Figure 9 (a)–(b) (top row) shows the final accuracies on the CIFAR100 dataset using ResNet-19.

Across all conditions, **TrSG** consistently outperforms both AS-SG and RS-SG, exhibiting robust performance with various initial values of V_{thr}. Overall, TrSG converges to better optima than previously used SG methods, reflecting its stable gradient flow. Additional analysis on how TrSG affects training is provided in Sec. 11.1.

Training Loss and Validation Accuracy. We further examine how the training loss and validation accuracy evolve over epochs. Figure 9 (c)–(f) (bottom row) shows that, for both soft and hard reset neurons, TrSG achieves faster convergence in training loss compared to AS-SG or RS-SG and maintains a higher validation accuracy throughout training. This improvement demonstrates TrSG's ability to circumvent the gradient issues inherent in AS-SG and RS-SG. Thus, TrSG not only reaches higher final accuracy but also fosters a more stable learning process from the early stages.

A similar trend appears when varying the initial value of τ, and when using different surrogate function shapes (e.g., Sigmoid), which can be found in the supplementary material (Secs. 11.3 and 12.1).

Dataset	MP-Init	TrSG	Accuracy (%)
CIFAR100 (T=4)	✗	✗	75.55 ± 0.06
	✓	✗	76.09 ± 0.04
	✗	✓	77.15 ± 0.06
	✓	✓	**77.66 ± 0.15**
DVS-CIFAR10 (T=10)	✗	✗	76.60 ± 0.20
	✓	✗	77.37 ± 0.20
	✗	✓	80.83 ± 0.64
	✓	✓	**81.43 ± 0.40**

Table 2. Ablation study of MP-Init and TrSG on ResNet-19.

6.3. Ablation Study and Final Configuration

Building on earlier findings, given their superior quality and stability, we select the **rectangular-shaped** surrogate function and **soft reset** neuron for subsequent experiments.

Next, we validate the individual effects of MP-Init and TrSG through an ablation study on CIFAR100 and DVS-CIFAR10. From Table 2, both MP-Init and TrSG clearly improve performance over the baseline. These results confirm the complementary contributions of both MP-Init and TrSG, justifying the adoption of both as our default setup

Dataset	Method	Network	Timestep	Accuracy (%)
CIFAR10	tdBN [52]	ResNet-19	2 / 4 / 6	92.34 / 92.92 / 93.16
	TET [8]		2 / 4 / 6	94.16 / 94.44 / 94.50
	TEBN [11]		2 / 4 / 6	94.57 / 94.70 / 94.71
	TAB [26]		2 / 4 / 6	94.73 / 94.76 / 94.81
	Ours		2 / 4 / 6	**95.05 ± 0.15 / 95.34 ± 0.06 / 95.50 ± 0.16**
CIFAR100	TET [8]	ResNet-19	2 / 4 / 6	72.87 / 74.47 / 74.72
	TEBN [11]		2 / 4 / 6	75.86 / 76.13 / 76.41
	TAB [26]		2 / 4 / 6	76.31 / 76.81 / 76.82
	Ours		2 / 4 / 6	**76.87 ± 0.16 / 77.66 ± 0.15 / 77.91 ± 0.32**
ImageNet	tdBN [52]	ResNet-34	6	63.72
	Dspike [31]		6	68.19
	TEBN [11]		4	64.29
	TAB [26]		4	67.78
	SEW-ResNet [15]	SEW-ResNet-34	4	67.04
	TET [8]		4	68.00
	TEBN [11]		4	68.28
	IMP+LTS [39]		4	68.90
	MPS [10]		4	69.03
	EAGD [46]		4	68.12
	Ours	ResNet-34	4 / 6	67.15 ± 0.03 / **68.73 ± 0.09**
		SEW-ResNet-34	4	**69.67 ± 0.08**
DVS-CIFAR10	tdBN [52]	ResNet-19	10	67.80
	MPBN [22]		10	74.40 ± 0.20
	RMP-Loss [21]		10	76.20 ± 0.20
	BNTT [27]	7-layer CNN	20	63.20
	TEBN [11]		10	75.10
	TAB [26]		4	76.70
	Ours	ResNet-19	10	**81.43 ± 0.40**
		7-layer CNN	4	**79.27 ± 0.75**

Table 3. Comprehensive comparison of our methods on static and dynamic datasets.

for subsequent experiments.

6.4. Comparison with State-of-the-arts

We compare our methods, **TrSG** and **MP-Init**, against state-of-the-art methods on both static datasets (CIFAR10/100, ImageNet) and dynamic datasets (DVS-CIFAR10), using standard LIF neurons and various backbones. Comprehensive results are summarized in Table 3.

On **CIFAR10 and CIFAR100**, our method achieves the best accuracy across all evaluated timesteps. On CIFAR10, we reach 95.50% at 6 timesteps, improving over the previous best (94.81% with TAB) by 0.69%pt. On CIFAR100, our method obtains 77.91% at 6 timesteps, surpassing the strongest baseline (76.82% with TAB) by 1.09%pt. Moreover, our results at fewer timesteps outperform baselines at larger timesteps, highlighting improved efficiency.

On **ImageNet**, our approach achieves 68.73% top-1 accuracy with ResNet-34 at 6 timesteps. With SEW-ResNet-34, we further push performance to 69.67% at only 4 timesteps, outperforming prior strong methods such as IMP+LTS [39] (68.90%) and MPS [10] (69.03%). Unlike these approaches, which require training additional neuron-specific parameters or complex optimization, our method introduces negligible overhead.

On the dynamic dataset **DVS-CIFAR10**, our method delivers 81.43% accuracy with ResNet-19 at 10 timesteps, exceeding the previous best (76.20% with RMP-Loss). With a lightweight 7-layer CNN, we also achieve 79.27% at only 4 timesteps, demonstrating that the benefits of TrSG and MP-Init generalize broadly to both static and event-based data.

7. Conclusion

In this paper, we introduced two key methods, MP-Init and TrSG, to tackle core challenges in training SNNs. MP-Init reduces TCS by aligning membrane potentials with their stationary distribution. At the same time, TrSG stabilizes the training process by stabilizing gradient flow with respect to the threshold voltage, allowing us to achieve state-of-the-art accuracy on static and dynamic datasets. Together, they contribute to improving the stability, efficiency, and accessibility of SNN training with minimal overhead.

Acknowledgement This work was supported by IITP and NRF grant funded by the Korea government(MSIT) (RS-2019-II191906, RS-2023-00213611, RS-2024-00457882)

References

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. 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*, 34:1537–1557, 2015. 1

[2] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015. 1

[3] Guangyao Chen, Peixi Peng, Guoqi Li, and Yonghong Tian. Training full spike neural networks via auxiliary accumulation pathway, 2023. 20

[4] Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, Johannes Schemmel, and Friedemann Zenke. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America*, 119:e2109194119, 2022. 1

[5] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2018. 5

[6] Mike Davies, Narayan Srinivasa, Tsung Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38:82–99, 2018. 1, 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 5, 12

[8] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv*, 2022. 2, 5, 7, 8, 13

[9] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017. 5

[10] Yongqi Ding, Lin Zuo, Mengmeng Jing, Pei He, and Hanpu Deng. Rethinking spiking neural networks from an ensemble learning perspective. In *ICLR*, 2025. 2, 8, 14

[11] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 34377–34390, 2022. 1, 2, 4, 5, 8, 12, 13, 14

[12] Grégory Dumont and Pierre Gabriel. The mean-field equation of a leaky integrate-and-fire neural network: measure solutions and steady states. *Nonlinearity*, 33:6381–6420, 2017. 3

[13] Jason Kamran Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *CoRR*, abs/2109.12894, 2021. 2

[14] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo Di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America*, 113:11441–11446, 2016. 1

[15] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In *Advances in neural information processing systems*, pages 21056–21069. Curran Associates, Inc., 2021. 5, 8, 13

[16] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2641–2651, Los Alamitos, CA, USA, 2021. IEEE Computer Society. 1, 2, 4, 5, 13, 18

[17] 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. 13

[18] Wei Fang, Zhaofei Yu, Zhaokun Zhou, Ding Chen, Yanqi Chen, Zhengyu Ma, Timothée Masquelier, and Yonghong Tian. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. *Advances in Neural Information Processing Systems*, 2023. 2

[19] Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2022. 2

[20] Yufei Guo, Xuhui Huang, and Zhe Ma. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023. 1, 2

[21] Yufei Guo, Xiaode Liu, Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Rmploss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2, 8

[22] Y. Guo, Y. Zhang, Y. Chen, W. Peng, X. Liu, L. Zhang, X. Huang, and Z. Ma. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19363–19373, Los Alamitos, CA, USA, 2023. IEEE Computer Society. 2, 8

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016. 5, 12

[24] Yulong Huang, Xiaopeng Lin, Hongwei Ren, Haotian Fu, Yue Zhou, Zunchang Liu, Biao Pan, and Bojun Cheng. Clif: Complementary leaky integrate-and-fire neuron for spiking neural networks. *Proceedings of Machine Learning Research*, 235:19949–19972, 2024. 2

[25] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv*, 2015. 2

[26] Haiyan Jiang, Vincent Zoonekynd, Giulia De Masi, Bin Gu, and Huan Xiong. TAB: Temporal accumulated batch normalization in spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2, 4, 5, 7, 8, 12, 13, 14

[27] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 15, 2020. 1, 2, 8

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 5, 12

[29] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10, 2016. 2

[30] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 2017. 5, 12

[31] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 23426–23439. Curran Associates, Inc., 2021. 2, 8

[32] 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*, 2023. 5, 7, 13

[33] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345:668–673, 2014. 1

[34] Emre O. Neftci. Data and power efficient intelligence with neuromorphic learning machines. *iScience*, 5:52–68, 2018. 1

[35] 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, 2023. 1, 2, 4, 5, 7

[36] Jeffrey Rosenthal. Minorization conditions and convergence rates for markov chain monte carlo. *Journal of the American Statistical Association*, 90:558–566, 1995. 16, 21

[37] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature 2019 575:7784*, 575:607–617, 2019. 1

[38] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv*, 2016. 2

[39] Hangchi Shen, Qian Zheng, Huamin Wang, and Gang Pan. Rethinking the membrane dynamics and optimization objectives of spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 92697–92720. Curran Associates, Inc., 2024. 2, 8, 13, 14

[40] Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2024. 12, 13

[41] Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 12

[42] SIQI WANG, Tee Hiang Cheng, and Meng-Hiot Lim. Ltmd: Learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout. In *Advances in neural information processing systems*, pages 28350–28362. Curran Associates, Inc., 2022. 1, 2, 4, 5, 7

[43] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12, 2018. 1, 2, 4, 18

[44] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1311–1318, 2019. 1, 2

[45] Yulong Yan, Haoming Chu, Yi Jin, Yuxiang Huan, Zhuo Zou, and Lirong Zheng. Backpropagation with sparsity regularization for spiking neural network learning. *Frontiers in Neuroscience*, 16:760298, 2022. 6, 20

[46] Shu Yang, Chengting Yu, Lei Liu, Hanzhi Ma, Aili Wang, and Erping Li. Efficient ann-guided distillation: Aligning rate-based features of spiking neural networks through hybrid block-wise replacement. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2025. 8

[47] Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems*, pages 64043–64058. Curran Associates, Inc., 2023. 12

[48] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks. In *Advances in Neural Information Processing Systems*, 2022. 2

[49] Ruokai Yin, Abhishek Moitra, Abhiroop Bhattacharjee, Youngeun Kim, and Priyadarshini Panda. Sata: Sparsity-aware training accelerator for spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42:1926–1938, 2022. 2

[50] Minsu Yoo, Yoon Sil Yang, Jong Cheol Rah, and Joon Ho Choi. Different resting membrane potentials in posterior parietal cortex and prefrontal cortex in the view of recurrent synaptic strengths and neural network dynamics. *Frontiers in Cellular Neuroscience*, 17:1153970, 2023. 12

[51] 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. 2

[52] 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*, pages 11062–11070, 2021. 5, 7, 8, 12

[53] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Han Zhang, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, 2023. 12

[54] Chenlin Zhou, Han Zhang, Liutao Yu, Yumin Ye, Zhaokun Zhou, Liwei Huang, Zhengyu Ma, Xiaopeng Fan, Huihui Zhou, and Yonghong Tian. Direct training high-performance deep spiking neural networks: a review of theories and methods. *Frontiers in Neuroscience*, 18:1383844, 2024. 1

[55] Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Qkformer: Hierarchical spiking transformer using q-k attention. In *Advances in Neural Information Processing Systems*, 2024. 12, 13, 18

[56] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 12

[57] Zhaokun Zhou, Kaiwei Che, Wei Fang, Keyu Tian, Yuesheng Zhu, Senior Member, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer v2: Join the high accuracy club on imagenet with an snn ticket. In *International Conference on Learning Representations*, 2024. 12