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

Supplementary Material

8. General Details

8.1. Biological Plausibility of MP-Init

A neuroscience paper [50] shows that resting membrane potentials (RMPs) differ across distinct cortical regions. Specifically, neurons in the prefrontal cortex (PFC) show approximately 10 mV higher resting membrane potentials compared to neurons in the posterior parietal cortex (PPC), primarily driven by differences in recurrent synaptic strengths mediated by NMDA receptors. These regional differences imply that the stationary membrane potential distribution may inherently vary across neural populations and brain areas. Therefore, the layer-wise initialization of MP-Init is similar to the brain's intrinsic regional variability in membrane potentials, thus providing a biological plausibility and justification for our proposed MP-Init method.

8.2. Limitations

Despite the advancements presented in the paper, several limitations remain. MP-Init efficiently mitigates TCS with minimal overhead, but exploring alternative initialization strategies could provide further insights. While training internal parameters of LIF neurons may introduce challenges in deploying SNNs on neuromorphic hardware optimized for simpler neuron dynamics. Future work should focus on adapting these methods for broader architectures and exploring more hardware-compatible designs for real-world applications.

8.3. Extension to Advanced Architectures and Tasks

We further evaluate MP-Init and TrSG on **Transformer-based SNNs**. On **DVS-CIFAR10** (10 timesteps), our method improves SpikeFormer2 from 78.90% to **80.10%** and QKFormer from 83.80% to **84.80%**. On **ImageNet** (4 timesteps), SpikingResformer rises from 74.34% to **75.62%**, and QKFormer from 78.80% to **79.90%** (Tab. 4). These **+1.0–1.3 %pt** gains indicate that MP-Init and TrSG transfer cleanly to advanced Transformer backbones without architectural changes.

Beyond classification, we validate task generality on **COCO2017** detection with EMS-ResNet10 [41]. Using the same training configuration (100 epochs) for both baseline and ours, MP-Init + TrSG improves mAP from 0.291 to **0.305** (mAP@0.5) and from 0.138 to **0.147** (mAP@0.5:0.95), as summarized in Tab. 5. Together, these results show that our methods are broadly effective across architectures and tasks while preserving the efficient inference pipeline of standard LIF SNNs.

Dataset	Architecture	Accuracy (%)
DVS-CIFAR10	Spikformer [56]	80.90
	Spikingformer [53]	81.30
	S-Transformer [47]	80.00
	SpikeFormer2 [57]	78.90
	+ Ours	**80.10**
	QKFormer [55]	83.80
	+ Ours	**84.80**
ImageNet	Spikformer [56]	70.24
	Spikingformer [53]	72.45
	S-Transformer [47]	72.28
	SpikingResformer [40]	74.34
	+ Ours	**75.62**
	QKFormer [55]	78.80
	+ Ours	**79.90**

Table 4. Transformer-based SNNs with and without our method. DVS-CIFAR10 is evaluated at 10 timesteps; ImageNet at 4 timesteps. Baselines are selected with comparable parameter budgets; our variant applies MP-Init + TrSG with all other settings held fixed.

Architecture	COCO2017 (mAP)
EMS-ResNet10 [41]	0.291 / 0.138
+ Ours	**0.305 / 0.147**

Table 5. Object detection on COCO2017 with EMS-ResNet10 (timesteps=3). Baseline and ours are trained under the *same* configuration for 100 epochs; values are mAP@0.5 / mAP@0.5:0.95.

9. Implementation Details

9.1. Architecture

We employ various network architectures to validate the effectiveness of MP-Init and TrSG. For the CIFAR-10/100 [28] and ImageNet [7] datasets, we utilize ResNet-19 [52] and ResNet-34 [23]. On the DVS-CIFAR10 dataset [30], we apply a 7-layer CNN (64C3-AP2-128C3-128C3-AP2-256C3-256C3-AP2-1024FC-10FC) as outlined in TEBN [11] and TAB [26], in addition to ResNet-19. We integrate the temporal dimension into the batch dimension to ensure proper normalization, following the tdBN method [52].

In MP-Init and TrSG, the running mean of membrane potential $E[U[T]]$, the threshold potential (V_{thr}), and the time decay factor (τ) are layer-wise parameters. We also tried channel-wise versions of them, but found only a small difference in performance. We set the initial $E[U[T]]$ to 0.0, V_{thr} to 1.0, and τ to 2.0, which are commonly used in SNNs [8, 11, 15, 16, 32, 39, 40, 55] without further tuning on networks/datasets for a fair comparison. The network directly processes the input through the first layer, which acts as the stem layer to generate spikes, while the final layer does not generate spikes [8]. To enhance generalization on the CIFAR-10/100 and DVS-CIFAR10 datasets, we incorporate dropout layers before each fully connected layer, consistent with the approach used in TEBN and TAB [11, 26].

9.2. Algorithm

MP-Init: Running-mean initialization. To mitigate TCS, we initialize each layer's membrane potential at the beginning of a simulation window using a running mean of the final potential from the previous window. The statistic is computed only from *active neurons* (those with $S^l[t] > 0$ at least once), so silent neurons do not bias the estimate. The running mean is updated *during training only*; at inference it is fixed and reused. Unless otherwise stated, we set the EMA coefficient to $\beta = 0.9$.

Algorithm 1 MP-Init (per-layer initialization)

```
1:  Initialize per-layer running mean $\mu^l \leftarrow 0$ for each layer $l$
2:  **Training phase:**
3:  **for** each batch with simulation window of length $T$ **do**
4:      Set initial membrane $U^l[0] \leftarrow \mu^l$
5:      Initialize activity mask $\texttt{mask}^l \leftarrow 0$
6:      **for** $t = 1$ to $T$ **do**
7:          Update $M^l[t]$ by Eq. (1), compute spikes $S^l[t]$ by Eq. (2), and reset $U^l[t]$ by Eq. (3)
8:          $\texttt{mask}^l \leftarrow \texttt{mask}^l \vee (S^l[t] > 0)$
9:      **end for**
10:     $\mu^l_{\text{batch}} \leftarrow \text{mean}\big(U^l[T] \,\big|\, \texttt{mask}^l{=}1\big)$
11:     $\mu^l \leftarrow (1-\beta)\mu^l + \beta\,\mu^l_{\text{batch}}$
12: **end for**
13:
14: **Inference phase:**
15: Set initial membrane $U^l[0] \leftarrow \mu^l$
16: Run forward dynamics with Eqs. (1) to (3)
```

Stable parameterization of τ and V_{thr}. To keep LIF parameters within valid ranges, we reparameterize

$$\tau = \tfrac{1}{\sigma(w)} \in (1,\infty), \quad V_{\text{thr}} = \text{Softplus}(k) \in (0,\infty),$$

where w, k are unconstrained learnable variables and $\sigma(\cdot)$ is the logistic sigmoid [16]. This yields stable gradients w.r.t. (w, k) while ensuring feasible (τ, V_{thr}).

Training vs. inference. *MP-Init:* During training, we track spikes $S^l[t]$ to update μ^l as in Algorithm 1. At inference, μ^l is no longer updated; it simply initializes $U^l[0]$, leaving the LIF forward rule unchanged.

TrSG: As detailed in Sec. 5.3, threshold multiplication is a *training-only* trick; at test time inference remains binary ($S^l[t] \in \{0,1\}$), and V^l_{thr} is absorbed into the next layer by rescaling $W^{l+1}_l \leftarrow V^l_{\text{thr}} W^{l+1}_l$. Thus, both MP-Init and TrSG keep the inference pipeline identical to standard LIF.

Practical note on efficiency. SpikingJelly [17] currently lacks CUDA-optimized kernels for jointly training V_{thr} and τ. We therefore implemented CUDA kernels supporting AS-SG, RS-SG, and TrSG, which accelerates end-to-end training substantially. For ResNet-34 on ImageNet with 8×A100-80GB, the total wall-clock for 120 epochs drops from $\sim$ 32.5 hours to $\sim$ 16.7 hours (about $1.94\times$ speed-up).

9.3. Dataset

CIFAR-10/100: We train ResNet-19 with timesteps of 2, 4, and 6, using one RTX3090 GPU. The optimization is performed using the SGD optimizer with a weight decay of 5e-4 and a CosineAnnealing scheduler, starting with an initial learning rate of 0.02 and a batch size of 64. We apply random cropping to 32x32 pixels, random horizontal flips, and data normalization.

DVS-CIFAR10: We train the 7-layer CNN and ResNet-19 with timesteps of 4 and 10, respectively, using one RTX3090 GPU. The training setup includes an SGD optimizer with a weight decay of 5e-4 and a CosineAnnealing scheduler, starting with an initial learning rate of 0.02 and a batch size of 64. The images are randomly cropped to 48x48 pixels and horizontally flipped, following the method described in [11, 26].

ImageNet: We train ResNet-34 with 6 timesteps and report results at multiple inference timesteps without additional fine-tuning. The optimizer is SGD with a weight decay of 4×10^{-5}, paired with a CosineAnnealing scheduler that starts at 0.10. The batch size is 512, distributed across 8 NVIDIA A100-80GB GPUs with synchronized batch statistics. SEW-ResNet-34 follows the same procedure and hyperparameters. Data augmentation includes random cropping and resizing to 224x224 pixels, random horizontal flips, and data normalization.

9.4. Parameter Overhead

If timestep-dependent parameters are introduced for each neuron, the parameter overhead would scale as $O(LTN)$ at maximum, where L, T, and N represent the number of layers, timesteps, and neurons per layer, respectively. This represents the worst-case scenario, assuming every neuron at every timestep requires independent parameters. By leveraging layerwise statistics, we significantly reduce this complexity. The total parameter overhead is reduced to $O(L)$, as V_{thr}, τ, and the running mean of the membrane potential are single parameters at each layer.

In comparison, prior works such as TEBN [11] and TAB [26] employ timestep-dependent parameters, resulting in an overhead of $O(LT)$. IMP+LTS [39], which trains the initial membrane potential per neuron, incurs an overhead of $O(N)$, where $(N >> L, T)$. In comparison, our method provides superior efficiency and achieves higher accuracy simultaneously.

10. Additional Details: MP-Init

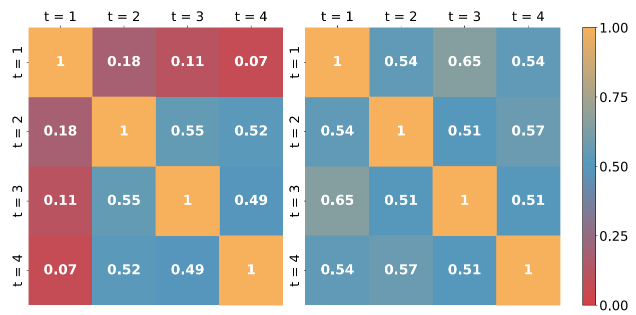
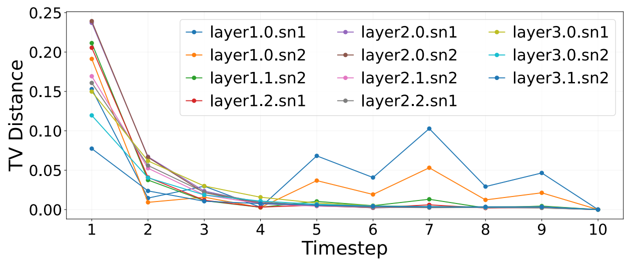


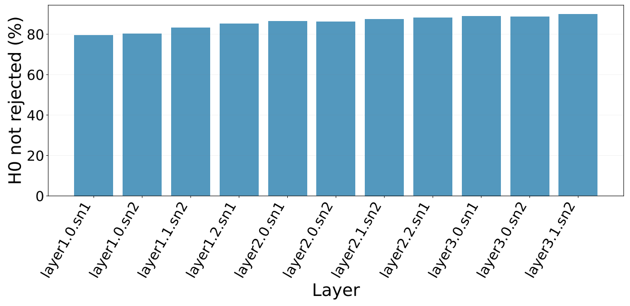
Figure 10. **Gradient conflicts across timesteps with and without MP-Init.** We visualize the cosine similarity of gradients across timesteps in ResNet-19 on CIFAR100. *Left:* Without MP-Init, the gradient from $t = 1$ strongly conflicts with later timesteps, reflecting the temporal inconsistency highlighted by MPS [10]. *Right:* With MP-Init, conflicts are largely resolved and similarities improve, indicating that MP-Init aligns temporal ensemble members and stabilizes training.

10.1. How does MP-Init impact training?

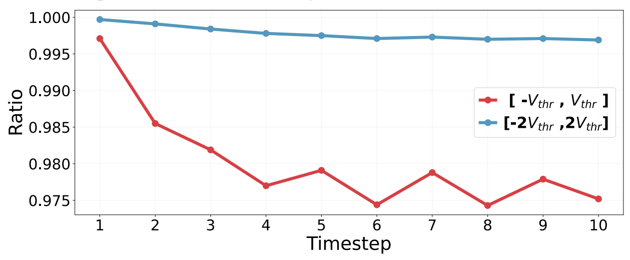
TEBN [11] and TAB [26] have shown that temporal covariate shift (TCS) indeed occurs, and our paper has already demonstrated that MP-Init effectively resolves it. However, prior studies did not provide clear evidence of *how* TCS directly impairs training. Inspired by the ensemble interpretation of SNNs in MPS [10], we revisited this question and conducted an experiment on ResNet-19 with CIFAR100. Specifically, we computed the gradients applied to a given layer at each timestep and measured the cosine similarity across timesteps. As shown in Fig. 10, without MP-Init (left), the gradients from $t = 1$ exhibit strong conflicts with later timesteps, indicating severe inconsistency across the temporal ensemble members. With MP-Init (right), these conflicts are largely alleviated, and the overall cosine similarity improves. This analysis provides the direct evidence that TCS not only degrades inference consistency but also hinders gradient alignment during training, and highlights how MP-Init effectively stabilizes optimization.



(a) TV distance between input distributions at different timesteps. Each curve compares $t = 1 \dots 9$ against the reference at $t = 10$.



(b) Ljung-Box test at lag=1. Bars show the percentage of neurons for which independence could not be rejected.



(c) Boundedness ratio of membrane potentials $U[t]$.

Figure 11. Empirical validation of Assumption 1 using ResNet-19 trained on **CIFAR100**. (a) TV distance across timesteps demonstrates near-identical input distributions. (b) Ljung-Box test confirms approximate input independence. (c) Boundedness ratio shows that membrane potentials remain within symmetric ranges relative to the threshold.

10.2. Validation of Assumption 1

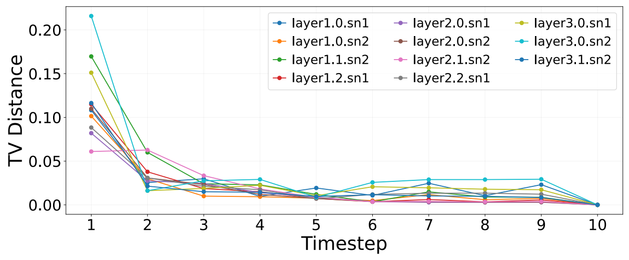
We empirically verify that inputs are approximately *i.i.d.* and that the membrane potentials of active neurons remain bounded, in line with Assumption 1. Using a trained ResNet-19 on CIFAR100, we report three complementary metrics:

- **Identical distribution (Fig. 11a).** We compute the TV distance between input distributions at timesteps $t = 1 \dots 9$ and the reference at $t = 10$. Distances are consistently small (< 0.1 after $t = 2$), indicating that the distribution quickly stabilizes and remains nearly identical across timesteps.
- **Independence (Fig. 11b).** We applied the Ljung-Box test

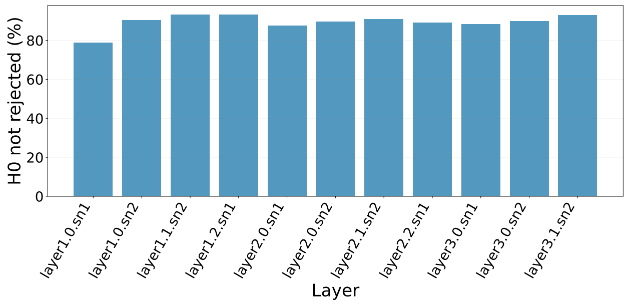
for autocorrelation with significance level $\alpha = 0.05$ using lag $= 1$. It illustrates 80–90% of neurons do not reject H_0 (independence). This suggests that temporal correlations are minor, supporting practical independence.

- **Boundedness (Fig. 11c).** Over 97% of membrane potentials remain within $\pm V_{\text{thr}}$ at all timesteps, and nearly all values lie within $\pm 2V_{\text{thr}}$, confirming that the dynamics are effectively bounded in practice.

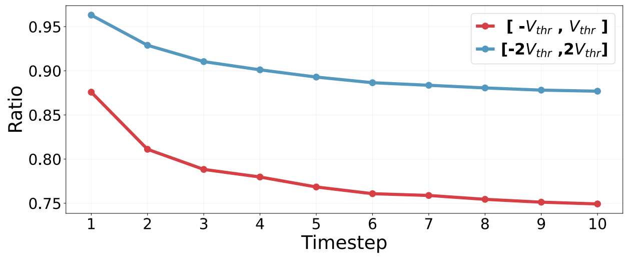
Taken together, these results confirm that Assumption 1 holds empirically: the inputs to spiking layers are nearly i.i.d., and the membrane potentials of active neurons remain bounded, thereby validating the theoretical conditions. Consistent findings are observed on DVS-CIFAR10 as well, as shown in Fig. 12, where ResNet-19 exhibits the same behavior satisfying all parts of Assumption 1.



(a) TV distance between input distributions at different timesteps. Each curve compares $t = 1 \dots 9$ against the reference at $t = 10$.

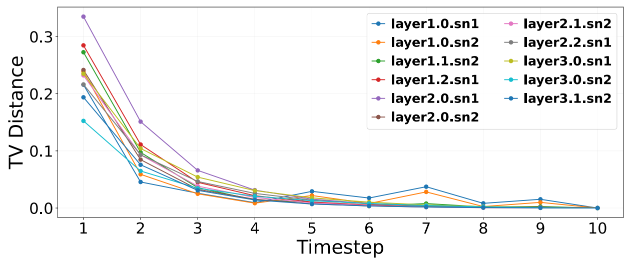


(b) Ljung-Box test at lag=1. Bars show the percentage of neurons for which independence could not be rejected.

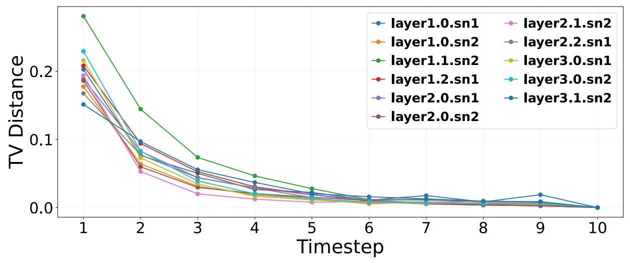


(c) Boundedness ratio of membrane potentials $U[t]$.

Figure 12. Empirical validation of Assumption 1 using ResNet-19 trained on **DVS-CIFAR10**. (a) TV distance across timesteps demonstrates near-identical input distributions. (b) Ljung-Box test confirms approximate independence. (c) Boundedness ratio shows that membrane potentials remain within symmetric ranges relative to the threshold.



(a) CIFAR100 (ResNet-19).



(b) DVS-CIFAR10 (ResNet-19).

Figure 13. TV distance between membrane potentials $U^l[t]$ at each timestep and the reference at $t = 10$. Across both CIFAR100 and DVS-CIFAR10, TV distance is initially large (> 0.3 at $t = 1$) but decreases sharply by $t = 2$, and falls below 0.1 by $t = 3$, indicating rapid convergence of membrane dynamics.

Method	**w/o MP-Init (%)**	**w/ MP-Init (%)**
tdBN (timestep=2)	72.35 ± 0.18	$\mathbf{74.01 \pm 0.33}$

Table 6. Effect of MP-Init at early timesteps on CIFAR100 using ResNet-19

10.3. Convergence of Membrane Potentials

As predicted by Theorem 1, the empirical TV distance curves in Fig. 13 clearly exhibit *exponential convergence*: deviations are large at $t = 1$ but shrink rapidly, already becoming negligible (< 0.1) by $t = 3$. This rapid convergence is highly desirable for SNNs, since reducing the number of timesteps directly improves both energy efficiency and latency. Notably, the benefit is especially pronounced at very low timesteps: at timestep=2, MP-Init delivers a **1.66%pt** accuracy improvement over the baseline (Tab. 6), highlighting that our method enables reliable inference even under aggressive latency constraints.

10.4. Proof of Theorem 1

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 TV distance between $U[t]$ and π decreases exponentially.

Step 1: Markov Property

We begin by confirming that $\{U[t]\}_{t\geq 0}$ forms a Markov chain. The dynamics of $U[t]$ are given by:

$$U[t+1] = f(U[t], I_{in}[t+1]), \tag{12}$$

where $f(\cdot)$ is a deterministic function dependent on the current membrane potential $U[t]$ and the new input $I_{in}[t+1]$. Since the inputs $\{I_{in}[t]\}_{t\geq 0}$ follow i.i.d, the future state $U[t+1]$ depends solely on the current state $U[t]$ and not on any earlier states.

Thus, the probability of transitioning to $U[t+1]$ given the entire history $\{U[s]\}_{s\leq t}$ depends only on $U[t]$:

$$P(U[t{+}1] \in A \mid U[t], \ldots, U[0]) = P(U[t{+}1] \in A \mid U[t]), \tag{13}$$

which confirms that $\{U[t]\}_{t\geq 0}$ is indeed a Markov chain.

Step 2: The Transition Kernel $P(x, A)$ and Its Relation
The transition kernel $P(x, A)$ describes the probability of moving from state x at time t to the set A at time $t+1$. Specifically, for the Markov chain $\{U[t]\}_{t\geq 0}$, $P(x, A)$ is the probability that the membrane potential at time $t+1$, denoted by $U[t+1]$, falls within the set A given that the membrane potential at time t is $U[t] = x$. This is formulated as:

$$P(x, A) = P(U[t+1] \in A \mid U[t] = x). \tag{14}$$

The transition kernel can be represented using the transition density $p(x, y)$, which is the probability density function for transitioning from state x to state y:

$$P(x, A) = \int_A p(x, y)dy, \tag{15}$$

where $p(x, y)$ encapsulates the dynamics governed by the function $f(U[t], I_{in}[t+1])$ and the input $I_{in}[t+1]$.

Step 3: Doeblin's Minorization Condition
We now show that the Markov chain $\{U[t]\}_{t\geq 0}$ satisfies the Doeblin's minorization condition [36]. We need to demonstrate the existence of a constant $\epsilon > 0$ and a probability measure $\mu(\cdot)$ such that for any $x \in S$ and any measurable set $A \subseteq S$, the transition probability satisfies:

$$P(x, A) \geq \epsilon\mu(A). \tag{16}$$

Given that the state space S is bounded by $[u^-, u^+]$, and assuming the transition density $p(x, y)$ is smooth and continuous (as the inputs are i.i.d. Gaussian), there exists a positive lower bound δ for $p(x, y)$ over the state space S. This implies that for any measurable set $A \subseteq S$:

$$P(x, A) = \int_A p(x, y)dy \geq \delta \cdot \mu(A), \tag{17}$$

where $\mu(\cdot)$ is a probability measure (e.g., uniform measure) over S. Setting $\epsilon = \delta$, we can validate that Eq. (16) is satisfied.

Step 4: Exponential Decrease in TV Distance
As a result of the Doeblin's minorization condition, the TV distance between the distribution $P^n(x, \cdot)$ of the Markov chain after n steps, starting from any state x, and the stationary distribution $\pi(\cdot)$ satisfies:

$$\|P^n(x, \cdot) - \pi(\cdot)\|_{TV} \leq C(1-\epsilon)^n, \tag{18}$$

for some constant $C > 0$. This inequality shows that the TV distance decreases exponentially as n increases, implying that the Markov chain converges rapidly to the stationary distribution π. This also implies the uniqueness and existence of the stationary distribution.

10.5. Proof of Lemma 1

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

We begin by calculating the expected value of the L2 norm between U^* and a constant c:

$$E[|U^* - c|^2] = E[(U^* - c)^2] = E[U^{*2}] - 2E[U^*]c + c^2. \tag{19}$$

Next, we take the derivative of this expression with respect to c:

$$\frac{\partial E[|U^* - c|^2]}{\partial c} = -2E[U^*] + 2c. \tag{20}$$

Setting the derivative to zero to find the minimum, we obtain:

$$c = E[U^*]. \tag{21}$$

Thus, the L2 norm is minimized when $c = E[U^*]$, proving the lemma.

11. Additional Details: TrSG

11.1. How does TrSG impact training?

Protocol. We analyze gradient flow of ResNet-19 on CIFAR100 with timestep=4 for 200 epochs. To isolate the effect of the SG formulation, we *fix* τ=2.0 and do not train V_{thr} or τ. We use a rectangular surrogate with γ=1 and the soft-reset mechanism.

Metrics. We quantify stability with three complementary metrics computed on convolutional weights of a given layer:

$$\text{AbsStr} = \operatorname*{mean}_i |g_i|,$$

$$\text{RatioAG} = \frac{1}{N}\sum_i \mathbb{1}\left(|x_i| < \tfrac{\gamma}{2}\right),$$

$$\text{GradCV} = \frac{\text{std}_i(|g_i|)}{\text{mean}_i(|g_i|)},$$

Epoch	V_{thr}=1.0	AbsStr	RatioAG (%)	GradCV
1	any SG	0.0122	6.72	0.818
100	any SG	0.0211	18.97	0.571

Table 7. Reference case where all SG variants behave same (ResNet-19, CIFAR100, timestep=4, τ=2.0).

Epoch	V_{thr}=0.1	AbsStr	RatioAG (%)	GradCV
1	AS-SG	0.2756	31.25	5.01
	RS-SG	7.8191	0.01	37.11
	TrSG	0.0046	3.05	**0.861**
100	AS-SG	0.0145	90.20	1.22
	RS-SG	NaN	0.00	NaN
	TrSG	0.0153	21.45	**0.865**

Table 8. Low-threshold stress test (CIFAR100, timestep=4, τ=2.0). RS-SG explodes (diverges), AS-SG floods, while TrSG remains stable. See Tab. 10 for accuracy.

where $g_i = \partial\mathcal{L}/\partial W_i$ and x_i is the surrogate argument.

AbsStr measures the average gradient magnitude. Very low values indicate vanishing gradients, where updates stall, while excessively high values point to exploding gradients, leading to erratic updates. Stable training requires AbsStr to remain in a moderate range. **RatioAG** quantifies the fraction of neurons receiving nonzero gradients. If RatioAG is too low, most neurons fail to receive updates (gradient starvation); if too high, too many neurons receive updates, causing gradient flood and poor selectivity. An intermediate range is thus ideal.

GradCV captures the variability of gradient magnitudes. Low variance (CV < 1) implies smooth, balanced updates that can be handled by a single learning rate, whereas high variance (CV $\gg 1$) means some weights update disproportionately faster than others, destabilizing training. TrSG is designed to maintain AbsStr at moderate levels, keep RatioAG in the healthy range, and reduce GradCV, together ensuring stable optimization.

Reference case (V_{thr}=1.0, Tab. 7). This widely used threshold serves as a baseline where all SG variants behave same (though 1.0 is not necessarily optimal). From epoch 1 to 100 we observe steadily improving yet stable gradients.

Low threshold (V_{thr}=0.1, Tab. 8). This stress test exposes small-threshold failure modes. At epoch 1, AS-SG floods (wide window), RS-SG explodes due to the $1/V_{\text{thr}}$ factor, while TrSG remains stable. By epoch 100, AS-SG still shows instability and RS-SG diverges; TrSG trains smoothly. These trends align with accuracy outcomes in Tab. 10.

Epoch	V_{thr}=2.0	AbsStr	RatioAG (%)	GradCV
1	AS-SG	0.0058	0.78	17.14
	RS-SG	0.0066	1.64	0.639
	TrSG	0.0092	1.62	**0.961**
100	AS-SG	0.0000	0.00	0.000
	RS-SG	0.0021	0.10	2.247
	TrSG	0.0323	6.90	**0.731**

Table 9. High-threshold stress test (CIFAR100, timestep=4, τ=2.0). AS-SG starves, RS-SG vanishes, TrSG preserves stable updates. See Tab. 10 for accuracy.

Reset	SG	V_{thr}				
		0.1	0.5	1.0	1.5	2.0
Soft	AS-SG	61.59	75.96	—	69.97	18.09
	RS-SG	*div.*	76.12	—	71.43	5.43
	TrSG	**73.99**	**76.82**	**75.56**	**71.97**	**64.27**
Hard	AS-SG	67.71	**76.34**	—	68.42	12.27
	RS-SG	10.36	75.85	—	72.51	52.10
	TrSG	**68.94**	76.33	**75.72**	**73.86**	**68.42**

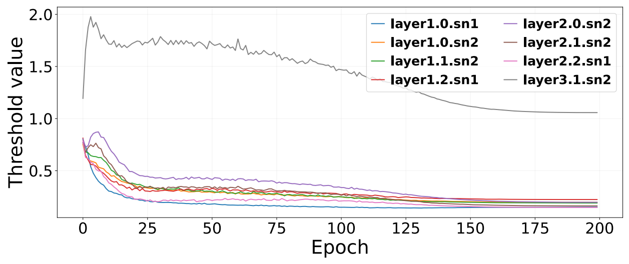
Table 10. Accuracy (%) with fixed τ=2.0 and *frozen* V_{thr} and τ to isolate SG effects. We vary $V_{\text{thr}} \in \{0.1, 0.5, 1.0, 1.5, 2.0\}$ under soft/hard resets. Conventional SGs fail outside a narrow range (explosion at 0.1, starvation/vanishing at 2.0), while **TrSG** remains robust across thresholds.

High threshold (V_{thr}=2.0, Tab. 9). Here the absolute window of AS-SG is too narrow and RS-SG's $1/V_{\text{thr}}$ suppresses signals. AS-SG starves, RS-SG vanishes with high variance, while TrSG stays trainable. The accuracy pattern is consistent with Tab. 10.

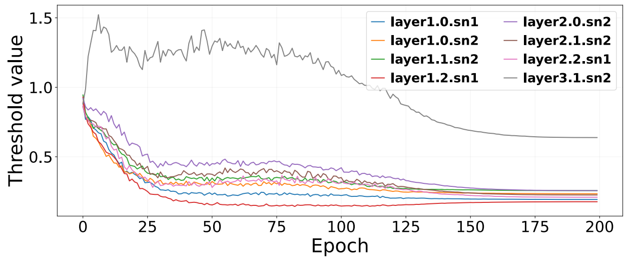
Do real trainings visit such extreme thresholds? Yes. When V_{thr} is trainable, many layers drift toward *lower* thresholds (often $\ll 1$), while late/deep layers move *upward* toward larger values (around ~ 2 on CIFAR100 and even higher on ImageNet). Figure 14 summarizes these trajectories: CIFAR100 and DVS-CIFAR10 show several early layers decreasing below 1 and the last layer increasing, whereas ImageNet exhibits a wider spread with some deep layers rising well beyond 2. These observations confirm that the brittle regimes identified for AS-SG/RS-SG are indeed encountered during training, precisely where TrSG's threshold-robustness is most beneficial.

Implications:

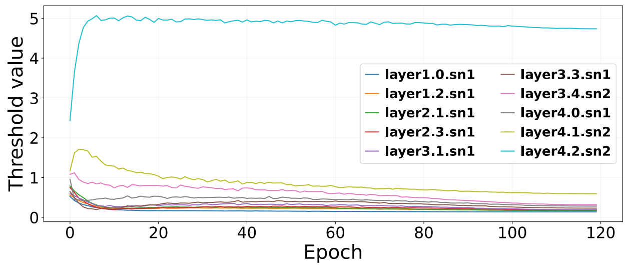
- Tabs. 7 to 9 explain *why* TrSG is superior: it maintains moderate magnitude (AbsStr), healthy selectivity (RatioAG), and low update variance (GradCV), avoiding flood/starvation and explosion/vanishing.



(a) ResNet-19 on CIFAR100



(b) ResNet-19 on DVS-CIFAR10



(c) ResNet-34 on ImageNet

Figure 14. **Trajectories of trainable V_{thr} during training.** Early layers tend to settle at lower thresholds, while late/deep layers increase toward larger thresholds.

- The threshold trajectories in Fig. 14 show that real trainings *do* visit the very regimes that break AS-SG/RS-SG, so threshold-robustness is not a corner case but a practical necessity.
- **Even when V_{thr} and τ are not trained**, TrSG remains the best choice across thresholds (Tab. 10); using TrSG is strictly beneficial, with no downside in practice.

11.2. Ablation on Advanced Architecture

Method	Top-1 (%)
QKFormer (HST-10-384)	78.80
+ AS-SG	77.09
+ RS-SG	78.70
+ TrSG	**79.90**

Table 11. Ablation on surrogate gradient choice for QKFormer (HST-10-384) on ImageNet at 4 timesteps. All runs follow the official training recipe; only the SG variant differs. MP-Init is enabled in all cases.

We ablate the surrogate gradient formulation on a large-scale Transformer SNN, QKFormer [55], using the official training protocol while keeping all settings fixed and changing only the SG type. With MP-Init enabled across the board, **TrSG** is the only variant that reliably improves over the baseline ($+1.10$ %pt), whereas **AS-SG** underperforms (-1.71 %pt) and **RS-SG** essentially matches or slightly underperforms (-0.10 %pt).

These results demonstrate that even on advanced Transformer-based SNNs, TrSG provides consistent benefits. More importantly, they show that using conventional SGs not only fails to deliver improvements but can actually degrade performance, underscoring TrSG as the superior and safer choice for training modern large-scale SNN architectures.

11.3. Validation on More Surrogate Functions

In Sec. 5, we introduced **TrSG (Threshold-robust Surrogate Gradient)** using a Rectangular-shaped surrogate as our primary example. In Sec. 6.2, we extended experiments to include a Triangular-shaped surrogate. Here, we further demonstrate TrSG's generality by evaluating it with an Arctan-based [16] and Sigmoid-based [43] surrogate functions.

Arctan Surrogate Function For an Arctan-based neuron, the derivative of the surrogate firing function is expressed as:

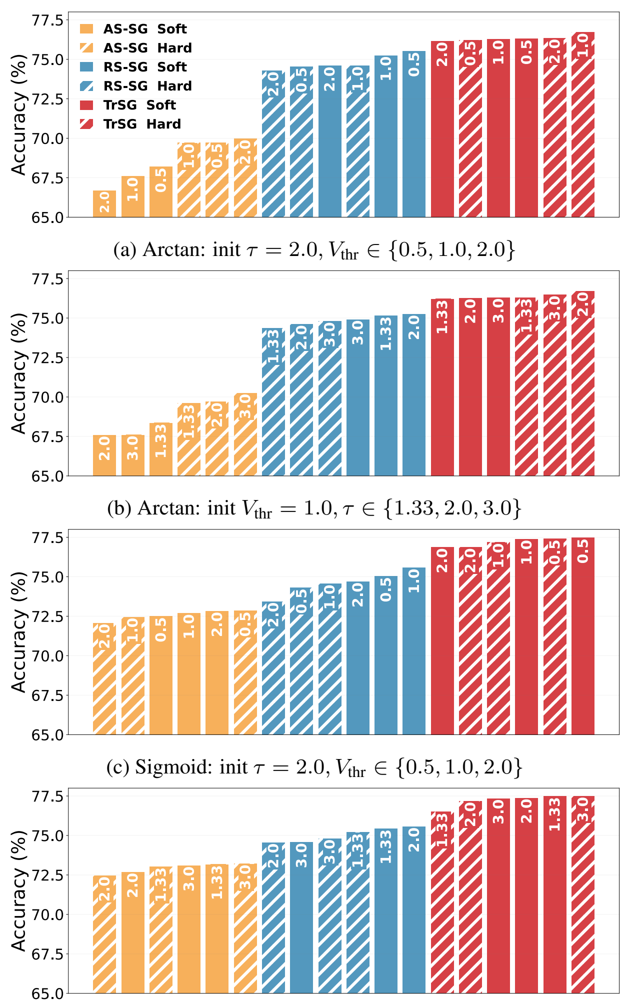
$$\frac{\partial f_{sr}(x)}{\partial x} = \frac{\gamma}{2\Big(1 + \big(\frac{\pi}{2}\,\gamma\, x\big)^2\Big)}.$$

Sigmoid Surrogate Function For a Sigmoid-based neuron, the derivative of the surrogate firing function is expressed as:

$$\frac{\partial f_{sr}(x)}{\partial x} = \frac{1}{\gamma}\frac{e^{\frac{x}{\gamma}}}{(1 + e^{\frac{x}{\gamma}})^2}.$$

where γ is the steepness parameter. Similar to our earlier analyses, we compare three SG methods (AS-SG, RS-SG, and **TrSG**) under both soft and hard reset neuron models.

Fig. 15 shows that **TrSG** consistently outperforms the conventional **absolute-scale** (AS-SG) and **relative-scale** (RS-SG) approaches, regardless of the initial threshold or time constant. This robust behavior parallels our findings in Sec. 6.2 for Rectangular and Triangular surrogates. TrSG maintains reliable performance across a wide range of initial conditions, emphasizing its capacity to provide stable gradient flow even for varied neuron parameters.



(a) Arctan: init $\tau = 2.0, V_{\text{thr}} \in \{0.5, 1.0, 2.0\}$

(b) Arctan: init $V_{\text{thr}} = 1.0, \tau \in \{1.33, 2.0, 3.0\}$

(c) Sigmoid: init $\tau = 2.0, V_{\text{thr}} \in \{0.5, 1.0, 2.0\}$

(d) Sigmoid: init $V_{\text{thr}} = 1.0, \tau \in \{1.33, 2.0, 3.0\}$

Figure 15. **Accuracy under different initial values of τ and V_{thr} when learning both parameters, using the *Arctan* and *Sigmoid* surrogate function.** **(Top)**: The results using Arctan surrogate function. **(Bottom)**: The results using Sigmoid surrogate function. We compare AS-SG, RS-SG, and **TrSG** on a ResNet-19 (CIFAR100, 4 timesteps), evaluating both soft- and hard-reset LIF neurons.

11.4. Surrogate Gradient Through V^l_{thr} and τ^l

In the main text, our analysis of threshold-related instabilities centered primarily on

$$\frac{\partial S^l[t]}{\partial M^l[t]},$$

i.e., how the spike output changes with respect to the membrane potential. However, similar issues arise when considering partial derivatives w.r.t. V^l_{thr} and τ^l. Below, we illustrate how AS-SG and RS-SG can each fail to provide stable gradients in these pathways and then show how **TrSG** maintains a consistent and well-scaled gradient flow.

1. Gradient w.r.t. V^l_{thr}. Recall from Eq. (2) that $S^l[t] = \mathcal{H}(\,M^l[t] - V^l_{\text{thr}}\,)$. When approximating $\mathcal{H}$ by a differentiable surrogate f_{sr}, existing methods define the argument x in either **absolute-scale** or **relative-scale** form:

- **AS-SG (Absolute-Scale).** $x = M^l[t] - V^l_{\text{thr}}$. Then,

$$\frac{\partial S^l[t]}{\partial V^l_{\text{thr}}} = f'_{sr}(x) \times \left(\frac{\partial M^l[t]}{\partial V^l_{\text{thr}}} - 1\right),$$

  whose update does not adapt its magnitude to V^l_{thr}. Consequently, if $|V^l_{\text{thr}}|$ becomes very large or very small, the gradient can vanish or dominate excessively.
- **RS-SG (Relative-Scale).** $x = \frac{M^l[t]}{V^l_{\text{thr}}} - 1$. By the chain rule,

$$\frac{\partial S^l[t]}{\partial V^l_{\text{thr}}} \approx f'_{sr}\big(\tfrac{M^l[t]}{V^l_{\text{thr}}} - 1\big) \times \frac{\partial}{\partial V^l_{\text{thr}}}\Big(\frac{M^l[t]}{V^l_{\text{thr}}} - 1\Big),$$

  which can introduce factors proportional to $\frac{1}{V^l_{\text{thr}}}$. If V^l_{thr} is tiny, the gradient risk is explosion; if V^l_{thr} is huge, updates can vanish.

TrSG circumvents both extremes by using a relative-scale definition for x and multiplying V^l_{thr} forward during training:

$$O^l[t] = V^l_{\text{thr}}\, S^l[t].$$

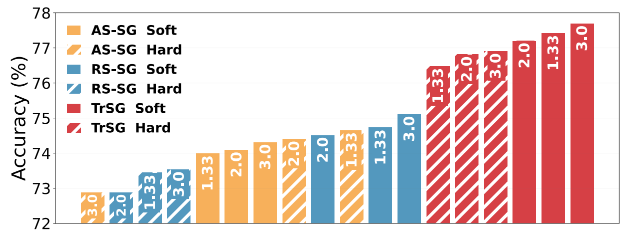
As discussed in the main text, this multiplication cancels out the problematic $\frac{1}{V^l_{\text{thr}}}$ factor during backpropagation, keeping gradients w.r.t. V^l_{thr} stable throughout training.

2. Gradient w.r.t. τ^l. Next, consider the discrete LIF update from Eq. (1): $M^l[t] = \left(1 - \frac{1}{\tau^l}\right) U^l[t-1] + \frac{1}{\tau^l} I^l_{\text{in}}[t]$. Since τ^l appears in denominators, it affects $S^l[t]$ indirectly via $M^l[t]$. Hence,

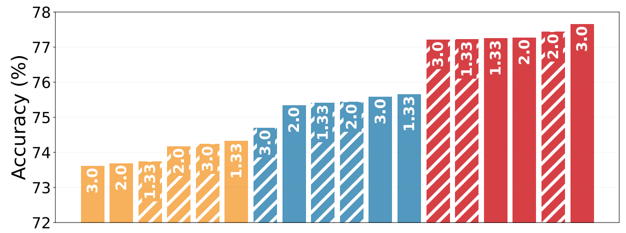
$$\frac{\partial S^l[t]}{\partial \tau^l} = \frac{\partial S^l[t]}{\partial M^l[t]} \times \frac{\partial M^l[t]}{\partial \tau^l}.$$

If $\frac{\partial S^l[t]}{\partial M^l[t]}$ is not well-scaled (as in standard AS-SG or RS-SG), the $\frac{\partial S^l[t]}{\partial \tau^l}$ may yield exploding or vanishing updates. **TrSG**, by ensuring a stable gradient path for $\frac{\partial S^l[t]}{\partial M^l[t]}$, similarly prevents instability when learning τ^l.

In conclusion, TrSG's threshold-multiplication mechanism and relative-scale formulation ensure that gradients remain appropriately scaled not only w.r.t. $M^l[t]$ but also along the V^l_{thr} and τ^l pathways. This provides a broad foundation for stable, end-to-end training of spiking neurons, even when multiple internal parameters are learnable.



(a) Rectangular-shaped Surrogate Gradient



(b) Triangular-shaped Surrogate Gradient

Figure 16. **Accuracy results with different initial τ values with initial $V_{\text{thr}} = 1.0$.** We compare three SG types **(i) AdSG (i) RdSG (i) TrSG** under two shapes **(a)** Rectangular and **(b)** Triangular, each with both soft and hard resets. We vary initial $\tau \in \{1.33, 2.0, 3.0\}$, indicated in the bars, and train ResNet-19 on CIFAR100 using 4 timesteps.

12. Additional Experimental Results

12.1. Varying Initial τ with a Fixed Initial V_{thr}

To further complement the experiments in Sec. 6.2, we conduct additional tests where we fix the initial threshold $V_{\text{thr}} = 1.0$ and instead vary the initial time constant $\tau \in \{1.33,\ 2.0,\ 3.0\}$. In Fig. 16, the top panel (Fig. 16a) reports accuracy under the Rectangular-shaped surrogate gradient, while the bottom panel (Fig. 16b) corresponds to the Triangular-shaped surrogate. Each panel compares three SG types (AdSG, RdSG, and TrSG) and two reset modes (soft vs. hard).

Our proposed **TrSG** consistently outperforms AS-SG and RS-SG across all initial τ values. TrSG maintains high accuracy in every setting, demonstrating its robust learning dynamics and further validating the overall trend observed in Sec. 6.2.

12.2. Impact of Trainable Parameters on Accuracy

To analyze the impact of training the threshold potential V_{thr} and the time decay constant τ, we conduct an ablation study. The results are summarized in Tab. 12.

From Tab. 12, we observe that training only τ results in a slight degradation in accuracy compared to fixed parameters. In contrast, training V_{thr} alone leads to a noticeable improvement in accuracy. Moreover, training τ alongside V_{thr} yields the best results, with a final accuracy of 77.22%. This indicates that while training τ alone may not be beneficial, its effect becomes synergistic when combined with V_{thr} training. By allowing both parameters to be learned, the model can dynamically adjust them to achieve a balance between convergence speed and network stability, thereby enhancing performance.

τ	V_{thr}	Accuracy (%)
✗	✗	75.56
✓	✗	74.61
✗	✓	76.98
✓	✓	77.22

Table 12. Ablation study of trainable parameters on ResNet-19 trained on CIFAR-100 with a timestep of 4. The initial (or fixed) values for V_{thr} and τ are set to 1.0 and 2.0, respectively. ✓ indicates trainable, while × means not trainable.

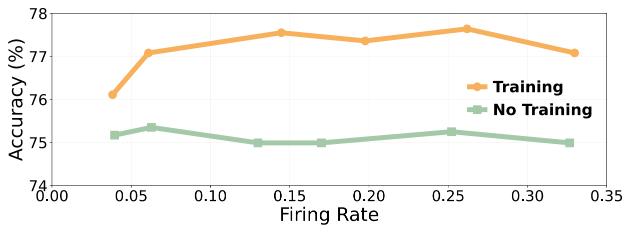


Figure 17. **Accuracy vs. firing rate under TrSG.** We sweep a sparsity loss [45] to match global firing rates and compare two settings: *Training* (learn V_{thr} and τ with TrSG) vs. *No Training* (both frozen).

12.3. Accuracy vs. Firing Rate

As shown in Sec. 6.1, MP-Init forms a Pareto front over firing rate. A natural question is whether training internal parameters (V_{thr}, τ) with TrSG merely increases spikes to boost accuracy. To decouple these effects, we vary a sparsity regularization loss [45] that directly controls spike counts and evaluate accuracy across matched firing rates, comparing *Training* (TrSG updates V_{thr}, τ) to *No Training* (both frozen). As seen in Fig. 17, the **Training** curve strictly dominates the **No Training** curve at all firing rates, indicating that the improvement stems from well-adapted neuron dynamics rather than excessive spikes. This is consistent with the threshold trajectories in Fig. 14, where layers evolve toward operating points that favor stable, informative spikes.

12.4. Energy Cost

Since firing rate tightly correlates with ACs (and thus energy), the analysis in Secs. 6.1 and 12.3 can be viewed as an energy–accuracy trade-off as well. To quantify this, we measure ACs and MACs and compute energy following [3]; the results in Tab. 13 show that, at similar firing rates (hence

MP-Init	TrSG	ACs / MACs(G)	Energy (mJ)	Acc (%)
✗	✗	1.03 / 0.14	1.57	75.58
✓	✗	1.01 / 0.14	1.57	76.20
✗	✓	1.13 / 0.14	1.68	77.13
✓	✓	1.12 / 0.14	1.67	77.68

Table 13. **Energy at comparable firing rates.** We report accumulate operations (ACs), multiply–accumulate operations (MACs), and the calculated energy costs.

similar energy), training internal neuron parameters with TrSG yields consistently higher accuracy. In short, TrSG improves accuracy at nearly the same energy cost, reinforcing that the gains come from better adaptation of V_{thr} and τ, not from increasing spike counts.

12.5. Final Values of Neuron Parameters

In addition to learning the network weights, our approach also optimizes the neuron parameters τ and V_{thr}, while MP-Init tracks an internal running mean of the membrane potential at the last timestep, $U[T]$. To investigate how these three quantities evolve through training, we visualize their final layer-wise values for both CIFAR100 and DVS-CIFAR10 in Fig. 18, considering soft vs. hard resets.

Running Mean of $U[T]$. The running mean of $U[T]$ typically settles at about half of V_{thr} in the soft-reset case (Figs. 18a and 18b) and roughly a quarter of V_{thr} for hard-reset neurons (Figs. 18c and 18d). This pattern indicates that the average post-synaptic potential remains below the learned threshold in most layers yet remains sufficiently high to generate spikes as needed.

Threshold V_{thr}. Across both datasets and reset types, the learned V_{thr} commonly falls below 0.5 in earlier and intermediate layers, suggesting that moderate thresholds facilitate stable spiking activity.

Decay Constant τ. We observe a consistent trend of τ converging around 1.5–2.0 in most layers for CIFAR100, while on DVS-CIFAR10, τ tends to reach even larger values (e.g., near 2.5–3.0). This discrepancy implies that dynamic event-driven data may benefit from neurons accumulating input more slowly, whereas CIFAR100's static images require a shorter effective integration time.

Last-Layer Behavior. Interestingly, in the final layer on both datasets, τ commonly drops close to 1, which makes it become a binary layer. The layer may act as a gating mechanism, helping the model to classify features decisively without excessive temporal integration.

12.6. Convergence Rate of Membrane Potential

To analyze the factors influencing the convergence rate described in Theorem 1, we conduct a mathematical analysis complemented by simple experiments, revealing that the time decay constant τ and the threshold voltage V_{thr} play pivotal roles in determining the convergence speed, thereby influencing the TCS problem.

As outlined in Eq. (16) and Eq. (18), Doeblin's Minorization Condition ensures not only the convergence of distributions but also provides insights into the convergence speed. Consider the uniform measure μ, which represents a uniform distribution over the interval $[u^-, u^+]$. For simplicity, let $U[t-1] = 0$, such that $U[t] = \frac{1}{\tau} I_{\text{in}}[t]$. Under these assumptions, Eq. (16) can be expressed as:

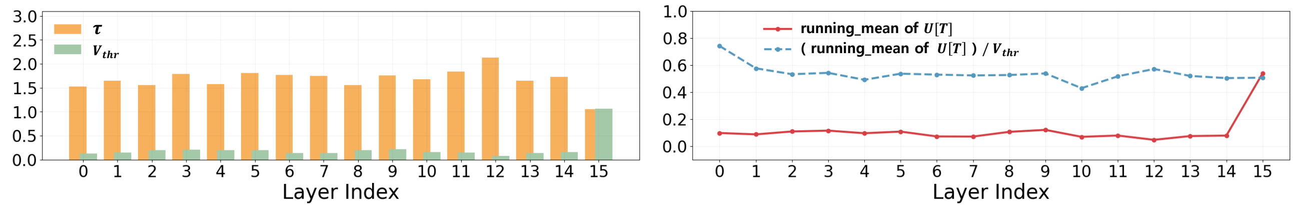
$$P\left(U[t] = \frac{1}{\tau} I_{\text{in}}[t] \in A\right) \geq \epsilon \mu(A), \tag{22}$$

From this expression with the Assumption. 1, it becomes evident that decreasing τ increases the maximum permissible value of ϵ, which, in turn, accelerates the convergence speed as per Eq. (18). Similarly, reducing the threshold V_{thr} results in the membrane potential reaching the threshold more frequently, triggering resets. This frequent resetting narrows the range $[u^-, u^+]$ of the membrane potential, thereby increasing ϵ and further accelerating convergence.

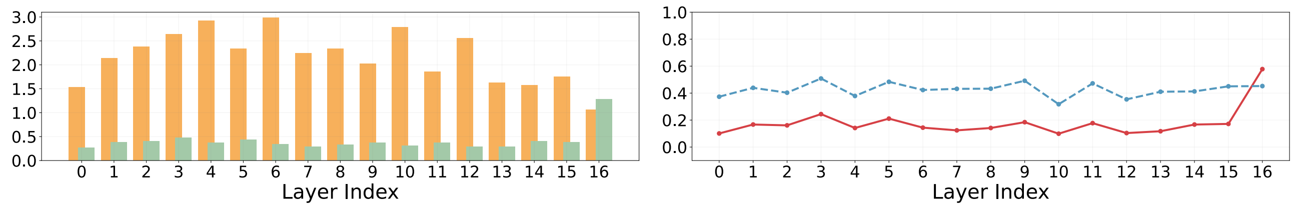
We conduct experiments utilizing soft reset LIF neurons with Gaussian input to validate the theoretical findings. The parameters V_{thr} and τ are varied, and the TV distance [36] is computed between the distribution at each timestep and the final distribution, as illustrated in Fig. 19. The results confirm that smaller values of V_{thr} and τ can alleviate the TCS problem. However, it is crucial to note that setting these parameters to excessively low values is not universally advantageous.

For instance, a threshold V_{thr} that is too low can result in excessive firing rates, leading to unstable network dynamics. Similarly, an overly small τ degrades the neuron's ability to accumulate membrane potential over time, effectively reducing its behavior to binary quantization. Therefore, the selection of V_{thr} and τ requires a careful balance to achieve both rapid convergence and stable network dynamics.

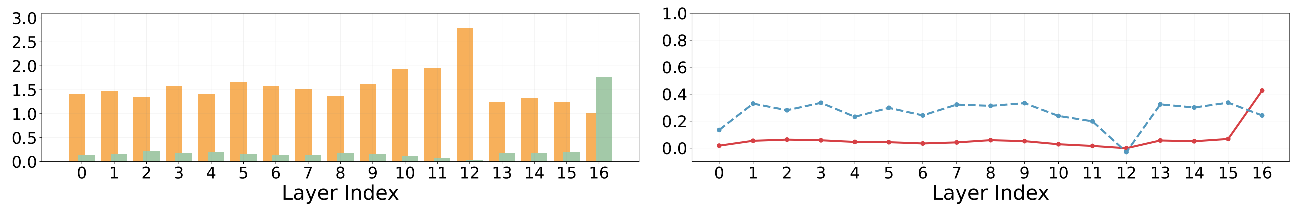
This analysis might establish a connection between MP-Init and TrSG. Rather than relying on manual tuning, training V_{thr} and τ as learnable parameters within the network can be more effective. As shown in Sec. 12.5, TrSG successfully optimizes V_{thr} and τ, leading them to moderately low values and achieving superior accuracy.



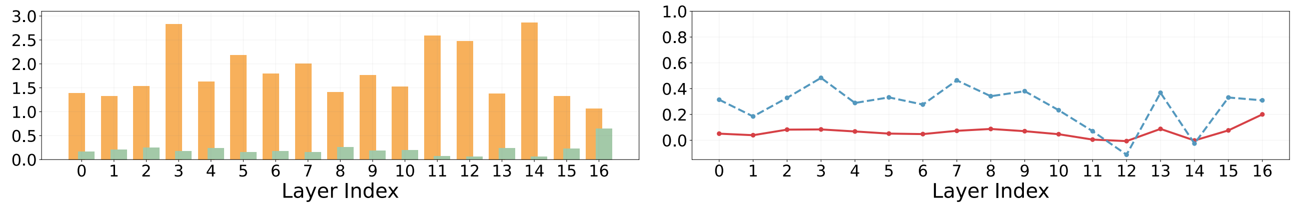
(a) CIFAR100 (soft reset)



(b) DVS-CIFAR10 (soft reset)

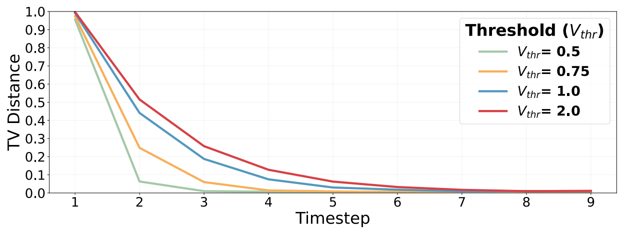


(c) CIFAR100 (hard reset)

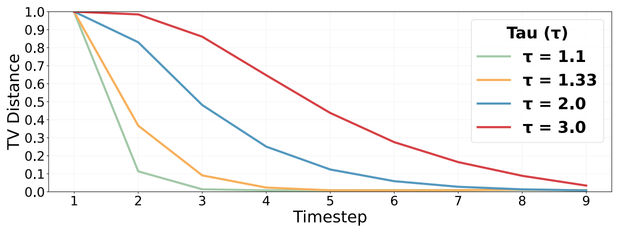


(d) DVS-CIFAR10 (hard reset)

Figure 18. The final values of V_{thr}, τ, and the running mean of the membrane potential at the last timestep ($U[T]$) after training. ResNet-19 is trained on CIFAR100 (timestep = 4) and DVS-CIFAR10 (timestep = 10).



(a) Impact of varying V_{thr} on convergence speed.



(b) Impact of varying τ on convergence speed.

Figure 19. TV distance of membrane potential distribution between timestep 10 and previous timesteps, illustrating the effects of altering the threshold V_{thr} (Left) or the time decay constant τ of the LIF neuron (Right). The input is randomly generated from a Gaussian distribution.