Membrane Sensitivity and Deployment Fragility of Learnable Time Constants in Spiking Neural Networks

Chiu-Chang Cheng[1] **Ya-Ning Chang**[1,*] **Chao-Hung Wang**[1,2,*]
[1]Miin Wu School of Computing, National Cheng Kung University
[2]Academy of Innovative Semiconductor and Sustainable Manufacturing,
National Cheng Kung University
[*]Corresponding authors: yaningchang@gs.ncku.edu.tw, chwang@gs.ncku.edu.tw

Abstract

Learnable neuronal dynamics improve temporal processing in spiking neural networks (SNNs), but can also create a hidden deployment vulnerability: parameters optimized for temporal integration can become fragile to hardware mismatch. We study this failure mode for learnable membrane decay factors β. Across feedforward, recurrent, convolutional, and residual SNNs under hardware-grounded, controlled inference-time mismatch at a demanding 20% coefficient of variation (CV), learnable-β models lose up to 27.3 percentage points (pp) of accuracy on temporally dependent tasks, whereas vision models with weaker temporal dependence lose at most 0.8 pp under per-neuron mismatch. We show that, under a surrogate-linearized first-order analysis, this fragility is governed by the membrane sensitivity $\Psi = \partial V/\partial\beta$, which quantifies how perturbations of the learned decay are amplified into membrane-potential deviations. Ψ admits an online recursion requiring only one additional state per layer, and regularizing its time-averaged energy yields **MARS** (**M**embrane-**A**ware **R**obustness through **S**ensitivity Regularization). We prove a conservative worst-case bound for feedforward networks and give a typical-case diagnostic explaining why Ψ remains informative when worst-case constants are vacuous. On the temporally dependent settings, MARS reduces mismatch-induced losses from up to 27.3 pp to below 0.8 pp under per-neuron, layer-wise, and global mismatch while preserving nominal accuracy, remains effective under additional weight and threshold mismatch or 8/6-bit weight quantization, and incurs no inference overhead. The same sensitivity energy also tracks the robustness conferred by noise augmentation, suggesting a common mechanism underlying robustness to perturbations of learned neuronal dynamics. Code is available at `https://github.com/eason891023/MARS-NeurIPS2026`.

1 Introduction

Learnable membrane time constants improve SNN accuracy by adapting neuronal integration timescales [13, 43, 29], but they also turn the decay factor $\beta = e^{-\Delta t/\tau}$ into a deployment-sensitive dynamical parameter. In analog and mixed-signal neuromorphic systems, device mismatch can alter effective neuron time constants [31, 35, 30], while temperature- and time-dependent drift can further shift deployed dynamics [28]. Measurements on a mixed-signal neuromorphic processor report parameter mismatch of roughly 10–20% [6], and calibration reduces but does not eliminate it [42]; we therefore treat a 20% coefficient of variation (CV) as a demanding stress point rather than a typical calibrated value. We abstract these effects as a controlled perturbation $\tilde{\beta} = \beta + \epsilon$ of the learned decay factor and investigate how such perturbations affect trained spiking neural networks.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Controlled mismatch evaluations reveal a task-dependent robustness–accuracy dilemma in which learned decay factors improve nominal temporal processing while also becoming a dominant source of deployment fragility. On SHD [9] with a feedforward SNN, learning β raises nominal accuracy over a fixed $\beta = 0.5$ (76.9% vs. 75.5%), but under per-neuron β mismatch at $\mathrm{CV} = 0.2$ the learnable-β model loses 27.3 pp, compared with 2.3 pp for fixed β. On SSC [9, 39], the fragility appears mainly under spatially coherent drift (3.4–4.3 pp), whereas DVS-Gesture [1] and a CIFAR-10 static-image control [23] lose at most 0.8 pp under per-neuron mismatch.

Existing approaches do not directly address this failure mode. Noise augmentation can improve robustness by training on sampled perturbations, but it does not reveal how time-constant variation propagates through SNN dynamics [5, 2, 6]. Calibration and hardware-in-the-loop methods require chip-specific effort [30, 35, 8], while quantization-aware training primarily targets weights and activations rather than membrane-time-constant sensitivity [18, 20, 25, 7].

Our core insight is that, under a first-order surrogate-linearized expansion, deployment fragility is governed by the membrane sensitivity Ψ, which acts as the local amplification factor from a decay-factor perturbation to the induced membrane displacement. We show that Ψ admits a closed-form online recursion, computable during the forward pass with one additional state per layer, without storing past trajectories.

The analysis therefore has an algorithmic consequence: the same sensitivity energy that controls first-order fragility can be regularized during training. We instantiate this as **MARS** (**M**embrane-**A**ware **R**obustness through **S**ensitivity Regularization), a training objective that penalizes time-averaged membrane sensitivity. Unlike noise augmentation, MARS does not require sampling a presumed deployment distribution; unlike generic smoothness regularizers [17, 38, 34, 44], it targets the internal Leaky-Integrate and Fire (LIF) sensitivity that accumulates through time.

Empirically, MARS limits the drop on the temporally dependent settings to below 0.8 pp under per-neuron, layer-wise, and global mismatch while preserving nominal accuracy (Figure 1; Table 1), and it remains effective when weight and threshold mismatch or 8/6-bit weights are added (Table 2). The same sensitivity energy also tracks the robustness obtained by noise augmentation, so the analysis supports both routes to robustness (Section 5.5). All experiments are hardware-grounded controlled mismatch evaluations in software simulation, not on-chip measurements: we perturb the trained decay factors β at inference time as a proxy for analog τ-mismatch and drift.

Contributions.

- **Task-dependent fragility from learnable decay factors.** We identify a deployment failure mode in which learning membrane decay factors improves nominal accuracy but can make trained SNNs fragile to inference-time perturbations of β. In hardware-grounded controlled mismatch evaluations, this fragility concentrates in temporally dependent settings and is negligible on a static CIFAR-10 control.
- **First-order mechanism and online sensitivity recursion.** We show that the membrane sensitivity Ψ is the first-order channel through which decay-factor perturbations displace LIF membrane trajectories and surrogate-linearized spike outputs. We derive a closed-form online recursion for Ψ that requires only one additional state per layer, and use it to obtain a conservative worst-case bound for feedforward networks and a typical-case diagnostic.
- **MARS regularization and empirical validation.** We instantiate the analysis as MARS, a training objective that regularizes the time-averaged sensitivity energy $\|\Psi\|^2$. We validate it across five SNN settings spanning feedforward, recurrent, convolutional, and residual architectures, and show that the same sensitivity energy explains the robustness of noise augmentation.

2 Related work

Surrogate-gradient training and learnable temporal dynamics. Surrogate-gradient methods make it possible to train deep SNNs despite the nondifferentiability of spike events [27, 45, 11, 36, 40, 41], and have made it practical to optimize SNNs on temporal and event-based benchmarks [45, 43, 9, 1]. A central direction in this line of work is to move beyond fixed neuronal dynamics: learnable membrane time constants [13, 43], adaptive spiking neurons [4], heterogeneous neuronal

parameters [29], and learnable axonal delays [16, 32] all improve temporal expressiveness by allowing the network to adapt how information is integrated over time. Fractional-order SNNs go further and replace the first-order membrane with long-memory, non-Markovian dynamics [15]; our analysis covers first-order LIF/PLIF dynamics and would require a new derivation for such models. These works typically assume that the learned temporal parameters are faithfully realized at inference; we study how sensitive the trained SNN is to deployment-time perturbations of them.

Deployment perturbations in neuromorphic and analog systems. Analog and mixed-signal neuromorphic systems naturally introduce mismatch in neuron and synapse parameters [31, 19, 30], time- and temperature-dependent drift [28], and residual calibration error [35]. Existing mitigation strategies include hardware calibration [31, 30, 35], hardware-in-the-loop optimization [35, 8], and noise-aware training [6]. HIL training, which can self-correct device mismatch on BrainScaleS-2 [8], remains the strongest option when the target chip is available; MARS targets settings without such access and is complementary to it. We isolate one deployment-relevant axis, perturbation of the learned membrane decay factor, and reduce the model's intrinsic sensitivity to it before deployment.

Robustness and sensitivity regularization. Robustness methods in deep learning often control sensitivity to inputs, weights, or sampled perturbations through Jacobian regularization [17, 38], input-gradient penalties [34], spectral constraints [44, 26], double backpropagation [10], contractive penalties [33], or noise injection [5, 2]. These methods control sensitivity to inputs ($\partial f/\partial x$), to learned weights, or to a chosen perturbation distribution, but none directly analyze how perturbations to an internal dynamical neuron parameter—the membrane decay β—propagate through the LIF recursion and accumulate over time.

3 Problem setup and MARS

3.1 LIF dynamics and deployment perturbation

We consider an SNN with L layers, where each hidden layer $l \in \{1, \dots, L-1\}$ has width n_l and a learnable decay factor $\beta^{(l)} = e^{-\Delta t/\tau^{(l)}} \in (0, 1)$, with $\tau^{(l)}$ the membrane time constant and Δt the simulation step; layer L is a non-spiking readout with weight $W^{(L)}$. Let $V_t^{(l)} \in \mathbb{R}^{n_l}$ denote the membrane potential and $S_t^{(l)} \in \{0, 1\}^{n_l}$ the spike vector. Following the delayed-reset convention of snnTorch [11], the subtract-reset (SR) dynamics are

$$V_t^{(l)} \;=\; \beta^{(l)}\, V_{t-1}^{(l)} \;+\; I_t^{(l)} \;-\; \theta\, S_{t-1}^{(l)}, \tag{1}$$

where $I_t^{(l)} = W^{(l)} S_t^{(l-1)} \in \mathbb{R}^{n_l}$ is the input current driven by the previous layer's spikes, $\theta \in \mathbb{R}_{>0}$ is the firing threshold, and $S_{t-1}^{(l)} = H(V_{t-1}^{(l)} - \theta)$ is the spike emitted at the previous step (H is the Heaviside step function applied componentwise). For hard-reset (HR) neurons, the dynamics zero the potential after a spike:

$$V_t^{(l)} \;=\; \beta^{(l)}\big(\mathbf{1} - S_{t-1}^{(l)}\big) \odot V_{t-1}^{(l)} \;+\; I_t^{(l)}, \tag{2}$$

with $\odot$ the elementwise product and $\mathbf{1}$ the all-ones vector of width n_l. During training, the non-differentiable H is replaced by a surrogate derivative $\sigma'(\cdot)$ [27, 45].

At deployment, each hidden layer experiences a perturbation

$$\tilde{\beta}^{(l)} \;=\; \beta^{(l)} + \varepsilon^{(l)}, \qquad l = 1, \dots, L-1, \tag{3}$$

where $\beta^{(l)}$ and $\varepsilon^{(l)}$ are layer-wise scalar decay factors in the main analysis, and $\varepsilon^{(l)}$ is treated as an unknown deployment perturbation. Concrete sources include analog device mismatch in τ [31, 35] and run-to-run drift [28]. If an implementation uses a no-reset leaky output layer with learnable β, its sensitivity $\Psi^{(L)}$ (no-reset recursion, Appendix B) is added to $\mathcal{R}_{\mathrm{MARS}}$, and Theorem 1 extends with one extra source term (Appendix A).

3.2 Membrane sensitivity and online recursion

We define the per-layer membrane sensitivity at time t as

$$\Psi_t^{(l)} \;\triangleq\; \frac{\partial V_t^{(l)}}{\partial \beta^{(l)}} \;\in\; \mathbb{R}^{n_l}, \tag{4}$$

i.e. the vector of partial derivatives of each neuron's membrane potential with respect to the layer-wise decay factor. At deployment the trained decay is perturbed as $\tilde{\beta}^{(l)} = \beta^{(l)} + \varepsilon^{(l)}$, and the induced membrane displacement is $\delta V_t^{(l)} = \tilde{V}_t^{(l)} - V_t^{(l)}$. For the surrogate-linearized dynamics, i.e. the tangent-linear response along the nominal trajectory with H' replaced by σ', $V_t^{(l)}$ is differentiable in $\beta^{(l)}$. A first-order Taylor expansion of (1) or (2) around the trained $\beta^{(l)}$ gives

$$\delta V_t^{(l)} \triangleq \tilde{V}_t^{(l)} - V_t^{(l)} = \Psi_t^{(l)} \varepsilon^{(l)} + O\big((\varepsilon^{(l)})^2\big), \tag{5}$$

so $\Psi_t^{(l)}$ is exactly the linear amplification factor between the deployment perturbation $\varepsilon^{(l)}$ and the induced membrane displacement $\delta V_t^{(l)}$. The same $\Psi_t^{(l)}$ also governs neuron-specific mismatch, where each neuron realizes its own perturbation and $\varepsilon^{(l)} \in \mathbb{R}^{n_l}$: in a feedforward LIF layer, neuron i's membrane depends only on its own decay, so $\delta V_t^{(l)} = \Psi_t^{(l)} \odot \varepsilon^{(l)} + O(\|\varepsilon^{(l)}\|_\infty^2)$ and $\|\delta V_t^{(l)}\|_2 \le \|\Psi_t^{(l)}\|_2 \|\varepsilon^{(l)}\|_\infty$ to first order. Recurrent layers couple neurons through W_{rec} and are treated empirically (Section 4.3).

The spike displacement requires a separate linearization because the deployed spike is $S_t^{(l)} = H(V_t^{(l)} - \theta)$ and the Heaviside function is discontinuous. In the surrogate-linearized analysis, we use

$$\delta S_t^{(l)} \approx \sigma'(V_t^{(l)} - \theta) \odot \delta V_t^{(l)}.$$

Substituting (5) gives

$$\delta S_t^{(l)} \approx \sigma'(V_t^{(l)} - \theta) \odot \Psi_t^{(l)} \varepsilon^{(l)} + O\big((\varepsilon^{(l)})^2\big).$$

Therefore, if the surrogate derivative satisfies $|\sigma'(x)| \le L_\sigma$ pointwise, the surrogate-linearized spike-output displacement obeys

$$\|\delta S_t^{(l)}\|_2 \le L_\sigma \|\Psi_t^{(l)}\|_2 |\varepsilon^{(l)}| + O\big((\varepsilon^{(l)})^2\big). \tag{6}$$

A key practical point is that $\Psi_t^{(l)}$ is computable *online* during the same forward pass used for training, via a closed-form recursion in $\Psi_{t-1}^{(l)}$ and $V_{t-1}^{(l)}$.

Differentiating (1) with respect to $\beta^{(l)}$ and using $\partial S_{t-1}^{(l)}/\partial \beta^{(l)} \approx \sigma'(V_{t-1}^{(l)} - \theta) \odot \Psi_{t-1}^{(l)}$ gives, for the subtract-reset case,

$$\Psi_t^{(l)} \approx V_{t-1}^{(l)} + \big(\beta^{(l)} - \theta\, \sigma'(V_{t-1}^{(l)} - \theta)\big) \odot \Psi_{t-1}^{(l)}; \tag{7}$$

the hard-reset recursion has the same online structure with reset-dependent coefficients (Appendix B). Both recursions use only quantities already available online ($V_{t-1}^{(l)}$, $\Psi_{t-1}^{(l)}$, $S_{t-1}^{(l)}$), are initialized with $\Psi_0^{(l)} = \mathbf{0}$, and carry only one $O(BN)$ state per layer, without storing past trajectories. When MARS is trained with backpropagation through time, autograd additionally keeps the per-step intermediates of the recursion, which increases peak training memory by a constant factor (1.13–2.30$\times$ in our settings; Appendix P); inference is unaffected. In unified form,

$$\Psi_t^{(l)} = \alpha_{\mathrm{eff}}^{(l)}(t) \odot \Psi_{t-1}^{(l)} + d^{(l)}(t), \tag{8}$$

where $\alpha_{\mathrm{eff}}^{(l)}(t) \in \mathbb{R}^{n_l}$ is the reset-dependent *effective decay* (e.g., $\beta^{(l)} - \theta\sigma'(V_{t-1}^{(l)} - \theta)$ in the SR case) and $d^{(l)}(t) \in \mathbb{R}^{n_l}$ is the input-driven term (e.g., $V_{t-1}^{(l)}$ in the SR case). For recurrent layers, differentiating the recurrent input adds a state-dependent recurrent sensitivity term; the full extension is given in Appendix B. The same recursion form will appear in the theory below.

3.3 MARS objective

Equations (5)–(6) show that $\|\Psi_t^{(l)}\|_2$ controls the first-order amplification from deployment perturbations in β to both membrane displacement and surrogate-linearized spike displacement. We therefore define the MARS regularizer as the time-averaged sensitivity energy summed over learnable-β layers:

$$\mathcal{R}_{\mathrm{MARS}}(\theta_{\mathrm{net}}) = \sum_{l=1}^{L-1} \frac{1}{T} \sum_{t=1}^{T} \big\|\Psi_t^{(l)}\big\|_2^2, \tag{9}$$

where T is the sequence length, the sum runs over the $L-1$ learnable-β LIF/PLIF layers, and θ_{net} collects the trainable network parameters (weights and learnable β). For PLIF-style parameterizations such as $\beta = \mathrm{sigmoid}(w)$, the deployed and perturbed quantity is still β, so the same Ψ applies and gradients reach w through the chain rule. A fixed readout only accumulates downstream perturbations; a learnable leaky readout adds its own term. The total training objective is

$$\mathcal{L}(\theta_{\text{net}}) \;=\; \mathcal{L}_{\text{task}}(\theta_{\text{net}}) \;+\; \lambda \cdot \mathcal{R}_{\text{MARS}}(\theta_{\text{net}}), \tag{10}$$

where $\mathcal{L}_{\text{task}}$ is the standard task loss (cross-entropy on the readout output) and $\lambda > 0$ is a scalar regularization strength. (10) directly targets the trainable source of deployment fragility identified by (5)–(6), rather than matching a particular perturbation distribution.

Pseudocode for the per-layer online MARS recursion is given as Algorithm 1 in Appendix C.

4 Theoretical analysis

4.1 Worst-case perturbation bound

To connect MARS to deployment robustness, we propagate single-layer perturbations through the surrogate-linearized network. The perturbation originating at hidden layer l is amplified by every downstream hidden layer $k > l$ through an unrolled temporal-convolution gain. Appendix A, Eqs. (17)–(20), derives this gain by unrolling the downstream membrane perturbation recurrence and bounding the resulting geometric kernel. We use the compact form

$$\lambda_r^{(k)} \;=\; \frac{L_\sigma \, \|W^{(k)}\|_{\text{op}}}{1 - \bar{\alpha}^{(k)}}, \tag{11}$$

where $\|W^{(k)}\|_{\text{op}}$ denotes the spectral (operator) norm of the feedforward weight matrix at layer k, $\bar{\alpha}^{(k)} \in [0, 1)$ is a scalar upper bound on the (worst-case) effective decay coefficient $\alpha_{\text{eff}}^{(k)}(t)$ from (8), and L_σ is the surrogate-gradient bound from (6). The cumulative gain from layer l to the layer feeding the readout is the propagation factor

$$p^{(l)} \;=\; \prod_{k=l+1}^{L-1} \lambda_r^{(k)}, \tag{12}$$

with the convention $p^{(L-1)} = 1$ (empty product).

Theorem 1 (Surrogate-linearized worst-case MARS bound)**.** *Under the assumptions* (A1) $\|\varepsilon^{(l)}\|_\infty \le \varepsilon_{\max}^{(l)}$ *for the (layer-wise or neuron-specific) decay perturbation of each hidden layer;* (A2) *the spike perturbation is surrogate-linearized and* $|\sigma'(x)| \le L_\sigma$ *for all* x*;* (A3) $|\alpha_{\text{eff}}^{(l)}(t)| \le \bar{\alpha}^{(l)} < 1$ *for every hidden layer* $l \ge 2$ *and timestep, the worst-case squared output perturbation of the surrogate-linearized readout of a feedforward network obeys*

$$\max_{\|\varepsilon^{(l)}\|_\infty \le \varepsilon_{\max}^{(l)}} \left\|\delta V_T^{(L)}\right\|_2^2 \;\le\; C_{\text{arch}} \sum_{l=1}^{L-1} \left(p^{(l)}\, \varepsilon_{\max}^{(l)}\right)^2 \cdot \frac{1}{T} \sum_{t=1}^{T} \left\|\Psi_t^{(l)}\right\|_2^2 \;+\; O(\varepsilon_{\max}^3), \tag{13}$$

where $\delta V_T^{(L)}$ *is the readout-layer membrane perturbation at the final timestep* $t = T$*,* $C_{\text{arch}} \triangleq T^2\,(L-1)\,(L_\sigma\,\|W^{(L)}\|_{\text{op}})^2$ *collects the readout and temporal accumulation factors, and* $\varepsilon_{\max} \triangleq \max_l \varepsilon_{\max}^{(l)}$*; the remainder vanishes for the tangent-linear response (Appendix A). The additional condition* $\lambda_r^{(l)} < 1$ *is not required for the bound; when it holds, the propagation factors are contractive across depth.*

The proof uses first-order expansion of the surrogate-linearized dynamics to obtain local membrane perturbations proportional to $\Psi_t^{(l)}$ (5), unrolls downstream temporal recursions as stable convolutions with gain $\lambda_r^{(k)}$, and then bounds the readout accumulation by Cauchy–Schwarz. The full proof is in Appendix A; it uses the perturbation only through $\|\delta V_t^{(l)}\|_2 \le \|\Psi_t^{(l)}\|_2 \|\varepsilon^{(l)}\|_\infty$, so it covers layer-wise, neuron-specific, and global perturbations alike. The theorem is a sufficient bound rather than a prediction of the accuracy drop. For a fixed trained network, the bound decomposes the effect of β mismatch into three factors: the magnitude of the deployment perturbation, the downstream amplification imposed by the network, and the time-averaged membrane sensitivity energy. MARS directly reduces the last factor through $\mathcal{R}_{\text{MARS}}$ in (9), thereby lowering the first-order vulnerability to perturbations in the learned decay factors.

4.2 Typical-case modeling estimate and relaxed diagnostic

Theorem 1 is intentionally conservative: all trained models we examine have $\lambda_r^{(l)} \gg 1$ on at least one hidden layer, yet the observed degradation is far below what these constants imply. To explain why the same sensitivity energy remains informative, we use a diagnostic estimate, not a formal guarantee, that replaces worst-case quantities with measurable proxies under explicit modeling assumptions (average-direction downstream amplification, weak surrogate/direction coupling, average effective decay, and cross-layer incoherence; Appendix L). With empirical surrogate sparsity $\eta^{(l)} = \frac{1}{T}\sum_t \|\sigma'(V_t^{(l)} - \theta)\|_2^2/(n_l L_\sigma^2)$, effective norm $\|W\|_{\text{eff}}^{(l)} = \|W^{(l)}\|_F/\sqrt{d_{\text{in}}^{(l)}}$ (Lemma 1), and trajectory- and neuron-averaged effective decay $\tilde{\alpha}^{(l)} = \frac{1}{Tn_l}\sum_t \|\alpha_{\text{eff}}^{(l)}(t)\|_1$, the relaxed gain is

$$\tilde{\lambda}_r^{(l)} \;=\; \frac{\sqrt{\eta^{(l)}}\cdot L_\sigma \cdot \|W\|_{\text{eff}}^{(l)}}{1-\tilde{\alpha}^{(l)}}. \tag{14}$$

The resulting estimate of $\mathbb{E}_\varepsilon\|\delta V_T^{(L)}\|_2^2$ again scales with the time-averaged sensitivity energy (Appendix L, Eq. (26)), so $\mathcal{R}_{\text{MARS}}$ is the controllable quantity in both the conservative bound and the diagnostic. The value 1 is the contraction boundary of the geometrically unrolled propagation gain: below it, successive downstream contributions contract, and above it they may grow. It is neither a fitted cutoff nor a cross-dataset classifier of robust versus fragile models; we use $\tilde{\lambda}_r^{(l)} < 1$ only as a within-setting diagnostic, not as a stability condition for the true binary-spike dynamics.

4.3 Recurrent settings

The bounds above analyze feedforward propagation. MARS itself only requires Ψ_t and applies to recurrent layers ($I_t = Wx_t + W_{\text{rec}}S_{t-1}$) by adding the state-dependent term $W_{\text{rec}}\text{diag}(\sigma'(V_{t-1} - \theta))\Psi_{t-1}$ to the recursion (Appendix B, Eq. (25)). Appendix B gives a sufficient contraction condition for such recursions; we do not verify it for our trained models and therefore treat SHD-RSNN and SSC as empirical stress tests rather than theorem-covered cases.

5 Experiments

5.1 Experimental setup

Datasets and architectures. We evaluate five dataset–architecture settings spanning four architecture families: SHD with a 3-layer fully connected SNN (SHD-FC), SHD with a recurrent SNN, SSC with a recurrent delay-learning SNN, DVS-Gesture with a 5-layer convolutional SNN, and CIFAR-10 with a spiking ResNet-18. SHD and SSC are audio spike-train datasets with strong temporal dependence [9]. The SSC setting uses the DelRec architecture [32, 16], which co-optimizes learnable axonal delays alongside learnable β. DVS-Gesture uses a VGG-style convolutional SNN with hard-reset hidden layers [1, 37]; CIFAR-10 uses a spiking ResNet-18 on repeated frames as a static-image control [23, 12, 46]. Full architectural and training details are deferred to Appendix D.

Robustness evaluation. We compare three main training modes—**Clean** (learnable β, no regularization), **MARS**, and **Noise Aug** (Gaussian β-perturbation at $\text{CV}_{\text{train}} = 0.15$)—under three deployment mismatch structures, all with $\tilde{\beta} = \text{clip}(\beta(1+\xi), 0.01, 0.99)$ and $\xi \sim \mathcal{N}(0, \text{CV}^2)$. In **per-neuron** mismatch, our main protocol, the layer-wise decay learned during training is expanded to one value per hidden neuron, and every neuron receives its own ξ, modeling fixed-pattern mismatch between nominally identical analog circuits [30, 6, 28]. In **layer-wise** drift, all neurons of a layer share one ξ, modeling structured residual errors from shared bias settings or imperfect calibration [35, 28]; in **global** drift, all layers share one ξ, modeling chip-wide temperature- or time-dependent drift [28]. We sweep $\text{CV} \in \{0, 0.05, 0.1, 0.15, 0.2, 0.3\}$ with 10 mismatch draws per checkpoint and level; summaries at $\text{CV} = 0.2$ use 200 draws for per-neuron mismatch on SHD and 100 draws for layer-wise and global drift, whose single shared multipliers make the estimate noisier.

Statistics. All settings use 5 independently trained seeds. Mismatch draws are first averaged within each checkpoint, and the training seeds are the unit of inference: we report means with standard deviations or 95% bootstrap confidence intervals across seeds. Methods are compared on paired

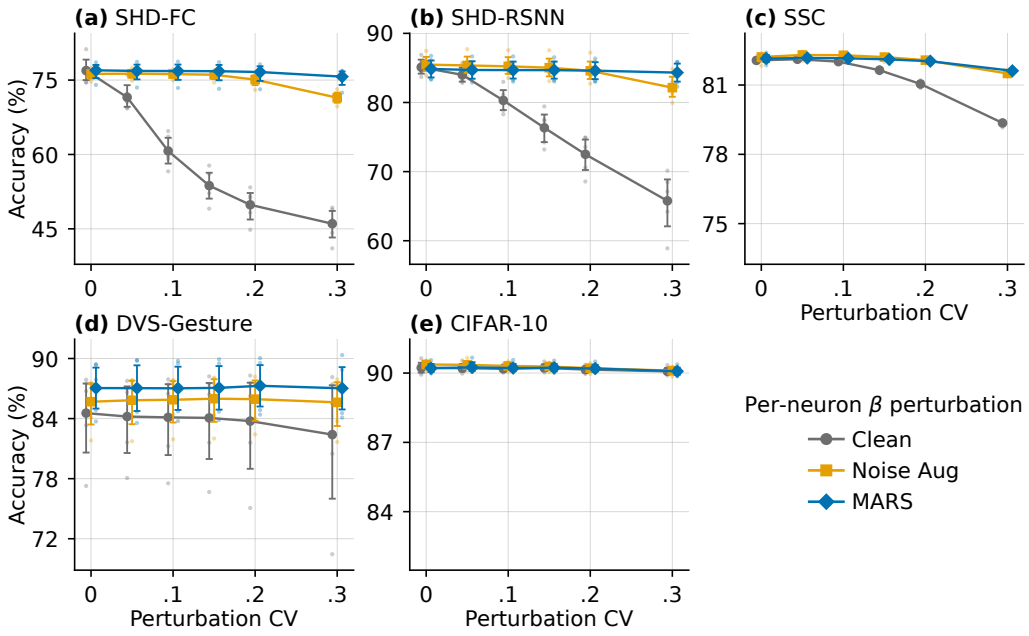


Figure 1: **Accuracy under increasing per-neuron β mismatch across five settings.** Every hidden neuron receives its own perturbed decay $\tilde{\beta}_i = \mathrm{clip}(\beta_i(1+\xi_i), 0.01, 0.99)$ with independent $\xi_i \sim \mathcal{N}(0, \mathrm{CV}^2)$; layer-wise trained decays are expanded to one value per neuron before perturbation. **Top row:** temporally dependent settings. **Bottom row:** DVS-Gesture and CIFAR-10 controls. Each checkpoint is evaluated with 10 mismatch draws per CV level; markers show the mean over 5 training seeds, faint dots the individual seeds, and error bars the 95% bootstrap confidence interval across seeds.

Table 1: **Accuracy drop (pp) at $\mathrm{CV} = 0.2$ under three mismatch structures.** Drop is nominal minus perturbed accuracy, mean±SD over 5 training seeds (C: Clean, NA: Noise Aug, M: MARS). *Per-neuron*: independent perturbation of every neuron's decay (as in Figure 1; 200 draws per checkpoint for SHD, 10 otherwise). *Layer-wise*: one multiplier shared by all neurons of a layer. *Global*: one multiplier shared by all layers. Layer-wise and global use 100 draws per checkpoint.

	Per-neuron			Layer-wise			Global		
Setting	C	NA	M	C	NA	M	C	NA	M
SHD-FC	27.3±3.4	1.4±0.6	0.4±0.3	20.8±2.5	1.5±0.7	0.6±0.5	17.7±2.1	1.5±0.5	0.7±0.4
SHD-RSNN	11.7±2.4	0.9±0.5	0.3±0.4	18.1±3.8	1.3±0.4	0.7±0.5	19.0±2.8	1.5±0.5	0.7±0.5
SSC	1.0±0.1	0.1±0.2	0.1±0.1	3.4±1.1	0.3±0.4	0.4±0.4	4.3±0.6	0.8±0.4	0.7±0.3
DVS-Gesture	0.8±1.3	−0.3±0.4	−0.2±0.4	4.5±1.8	1.6±0.9	2.3±1.4	5.0±2.6	1.8±0.7	4.2±3.5
CIFAR-10	0.1±0.1	0.2±0.1	0.0±0.1	0.1±0.1	0.2±0.1	0.1±0.1	0.1±0.1	0.1±0.1	0.1±0.1

seeds with shared mismatch draws. With five seeds, the smallest attainable exact two-sided sign-flip p-value is 0.0625, so we report paired effect sizes rather than claims of statistical significance.

5.2 Main results: robustness under β-variation

Learnable β is fragile on SHD, and MARS removes most of this fragility. Figure 1 summarizes robustness under per-neuron mismatch. At $\mathrm{CV} = 0.2$, Clean learnable-β models lose 27.3 pp on SHD-FC and 11.7 pp on SHD-RSNN, whereas MARS limits the drop to 0.4 and 0.3 pp while matching Clean nominal accuracy within one seed standard deviation (Appendix Table 4). Noise Aug also improves robustness substantially (1.4 and 0.9 pp). Paired over training seeds, the differences between the two robust-training methods are small (≤ 1.03 pp; Appendix F). MARS also does not simply turn learnable-β models into fixed-β models: on SHD-FC, fixed $\beta = 0.5$ reaches 75.5% nominal accuracy and still drops 2.3 pp, whereas MARS keeps the learnable-β accuracy of 77.0%

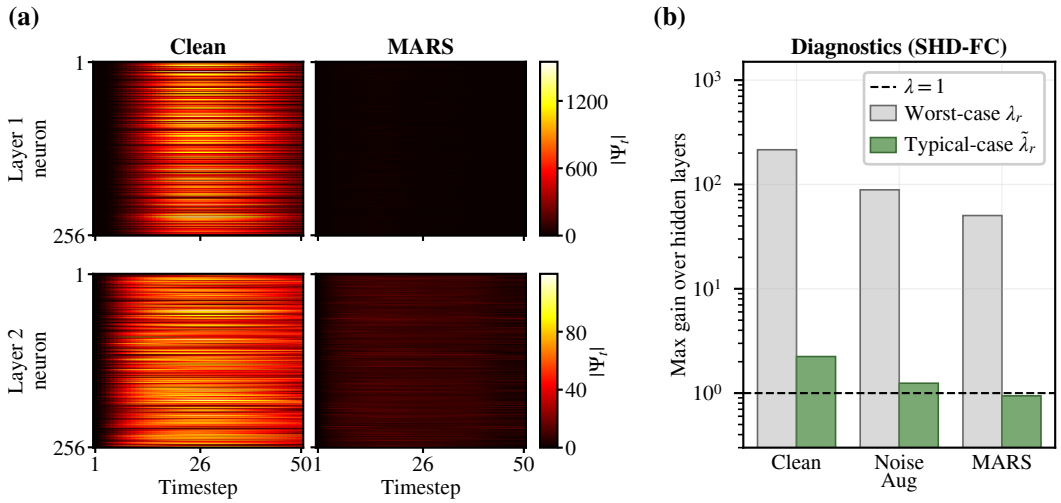


Figure 2: **Sensitivity visualization and diagnostic analysis on SHD-FC. (a)** Per-neuron sensitivity $|\Psi_t^{(l)}|$ (neurons $\times$ timesteps) for clean vs. MARS models. Bright pixels indicate high sensitivity to β-perturbation; the MARS columns are near-dark. MARS suppresses Ψ by $51\times$ in hidden layer 1 and $5.8\times$ in hidden layer 2. **(b)** Worst-case vs. typical-case diagnostics. All training modes have large worst-case propagation factors $\lambda_r \gg 1$, but MARS is the only mode below the typical-case diagnostic threshold $\tilde{\lambda}_r < 1$ in both hidden layers (0.89/0.94); Clean and Noise Aug exceed it.

with a 0.4 pp drop. Figure 2 shows the corresponding suppression of Ψ on SHD-FC and that, although all main modes have vacuous worst-case gains $\lambda_r \gg 1$, MARS is the only mode with $\tilde{\lambda}_r < 1$ in both hidden layers (Appendix M).

Spatially coherent drift exposes further fragility. Independent per-neuron errors can partly average out within a layer, whereas coherent drift shifts all decays of a layer, or of the whole network, together. Table 1 shows that layer-wise and global drift keep Clean SHD drops at 17.7–20.8 pp and also expose SSC, whose Clean drop grows from 1.0 pp under per-neuron mismatch to 3.4 (layer-wise) and 4.3 pp (global). Across the three temporally dependent settings, MARS consistently limits β-mismatch degradation (below 0.8 pp in every mismatch structure); it has lower mean degradation than Noise Aug on the two SHD architectures and comparable performance on SSC, without a universal significance claim.

Low-sensitivity controls. On CIFAR-10, all methods stay within 0.2 pp of nominal accuracy under every mismatch structure, and on DVS-Gesture within 0.8 pp under per-neuron mismatch; MARS does not regress nominal accuracy on either dataset. Under coherent drift, however, DVS-Gesture is not insensitive: the Clean drop reaches 4.5–5.0 pp, and both robust-training methods reduce it (Table 1). MARS targets the accumulation of decay-factor perturbations over time, and its effect is therefore concentrated in the temporally dependent settings.

5.3 Beyond β-only mismatch

Deployed devices perturb several parameters at once and store weights at limited precision. Table 2 combines per-neuron β mismatch with weight and threshold mismatch or with low-bit weights. Weight and threshold mismatch alone costs every method at most 0.34 pp, and adding it to β mismatch leaves the picture unchanged: both robust-training methods keep the drop at or below 1.6 pp, and MARS at or below 0.6 pp. Quantization itself, however, is not addressed by MARS: 6-bit PTQ lowers the nominal accuracy of MARS on SHD-RSNN by 1.7 pp ($84.8\% \rightarrow 83.2\%$). On SSC, additionally perturbing the learned recurrent delays by 5% together with β, weight, and threshold mismatch gives drops of 1.17/0.21/0.23 pp (Clean/Noise Aug/MARS, 5 draws per checkpoint), where MARS and Noise Aug are comparable. Log-normal, heavy-tailed (Student-t_3), and spatially correlated ($\rho \in \{0.5, 0.9\}$) β mismatch at $\mathrm{CV} = 0.2$ yields MARS drops of 0.24–0.82 pp on SHD (10 draws per checkpoint; Appendices H, I, and O).

Table 2: β **mismatch combined with other deployment errors on SHD.** Accuracy drop (pp), mean$\pm$SD over 5 training seeds (C: Clean, NA: Noise Aug, M: MARS). Per-neuron β mismatch at $\mathrm{CV} = 0.2$ is combined with per-weight and per-neuron threshold mismatch ($\mathrm{CV} = 0.05$ each) or with symmetric per-tensor post-training weight quantization (PTQ); PTQ drops are measured from the quantized nominal model. All rows use the same 200 β draws per checkpoint as Table 1.

	SHD-FC			SHD-RSNN		
Deployment errors	C	NA	M	C	NA	M
β only (Table 1)	27.3±3.4	1.4±0.6	0.4±0.3	11.7±2.4	0.9±0.5	0.3±0.4
β + weight + threshold	27.5±3.4	1.6±0.6	0.6±0.3	12.0±2.4	1.1±0.4	0.5±0.4
8-bit PTQ + β	27.5±4.0	1.4±0.6	0.1±0.4	12.1±2.1	1.0±0.5	0.1±0.2
6-bit PTQ + β	27.3±3.7	1.5±1.1	0.3±0.3	11.8±2.3	0.8±0.5	0.2±0.2

Table 3: **Simple baselines under per-neuron β mismatch at $\mathrm{CV} = 0.2$.** Nominal accuracy (%) and accuracy drop (pp), mean$\pm$SD over 5 training seeds; all rows use the same 200 mismatch draws per checkpoint. Weight decay and spectral normalization were trained only for SHD-FC, and recurrent DropConnect applies only to SHD-RSNN. Control hyperparameters are listed in Appendix D.

	SHD-FC		SHD-RSNN	
Method	Nominal	Drop	Nominal	Drop
Clean (learnable β)	76.9±2.7	27.3±3.4	85.1±1.3	11.7±2.4
β-L2	78.1±1.4	27.6±1.6	83.3±2.1	8.9±3.2
Hard β range $[0.1, 0.9]$	77.9±1.2	28.5±3.2	83.8±2.0	10.4±2.1
Weight decay	70.6±2.4	46.5±2.1	–	–
Spectral normalization	73.9±0.6	25.8±2.7	–	–
Recurrent DropConnect	–	–	83.2±1.3	9.7±1.5
Fixed β, homogeneous	75.5±2.4	2.3±0.4	86.2±1.2	1.7±0.3
Fixed β, heterogeneous	76.1±0.8	6.2±3.1	83.9±2.7	3.2±1.2
Noise Aug	76.2±1.1	1.4±0.6	85.5±1.3	0.9±0.5
MARS	77.0±1.7	0.4±0.3	84.8±1.6	0.3±0.4

5.4 Comparison with simple baselines

Table 3 tests whether simpler interventions reproduce the robustness of MARS. Shrinking or bounding the learnable decay (β-L2, a hard range $[0.1, 0.9]$), generic weight or operator regularization (weight decay, spectral normalization), and recurrent DropConnect all leave drops of 8.9–46.5 pp, comparable to or larger than Clean: constraining the magnitude of β or of the weights does not suppress the sensitivity that accumulates through the membrane recursion. Fixing β removes the learned-decay channel altogether and is therefore much more robust (1.7–6.2 pp), although its drops remain 5–17$\times$ larger than those of MARS. MARS thus retains learnable decay factors while removing most of their deployment fragility. We did not evaluate a full output-Jacobian penalty and therefore make no claim relative to every generic sensitivity regularizer.

5.5 Mechanism and temporal expressivity

Sensitivity energy tracks fragility beyond MARS. On the three temporally dependent settings, post-hoc time-averaged sensitivity energy orders the methods as MARS $<$ Noise Aug $<$ Clean in all 15 matched setting–seed triples. Pooled over these settings, Noise Aug reduces it by $4.67\times$ relative to Clean although it neither computes nor optimizes Ψ, and MARS by $78.2\times$. To avoid circularity with the MARS objective, we correlate log sensitivity energy with the per-neuron drop at $\mathrm{CV} = 0.2$ using only Clean and Noise Aug checkpoints, after removing per-setting means: Pearson $r = 0.94$ (Spearman $\rho = 0.95$, $n = 30$; $r = 0.91$ under layer-wise drift). This evidence is correlational, but it indicates that Ψ also explains why an intervention that never observes it becomes robust (Appendix K).

MARS compresses generic history but keeps task-relevant temporal structure. MARS does not shorten the task-context horizon: the shortest input prefix/suffix reaching 95% of full-sequence accuracy is essentially unchanged on all three temporally dependent settings. It also still relies on

temporal order: on SSC, time reversal, temporal shuffling, and rate-preserving repetition reduce MARS accuracy from 82.1% to 9.3%, 8.0%, and 2.8%, and on SHD-RSNN shuffling and rate repetition reduce it from 84.8% to 27.3% and 20.1%. At the same time, MARS reduces generic history: the linear decodability of raw input event counts 20 bins in the past falls from $R^2 = 0.47$ to 0.08 on SHD-FC (Noise Aug lies in between), and the learned decays contract selectively rather than collapsing (on SHD-RSNN, $\beta \in [0.010, 0.096]$ in the first recurrent layer vs. $[0.38, 0.55]$ in the second). We therefore read the effect as selective temporal compression rather than unchanged memory; tasks that require long generic memory may pay an accuracy cost that our benchmarks do not reveal (Appendix J).

6 Discussion

Learnable neuronal dynamics create a robustness problem distinct from weight noise or input perturbation: the decay factors that improve temporal processing become deployment-sensitive parameters. Because this fragility accumulates over time through the membrane sensitivity Ψ, both the fragility and the benefit of suppressing Ψ concentrate in temporally dependent settings, most strongly on SHD and on SSC under coherent drift.

Two routes to robustness. The same analysis explains why both MARS and noise augmentation reduce deployment fragility: both lower the sensitivity energy (Section 5.5). Noise augmentation samples an assumed deployment perturbation model during training and adds negligible training cost; MARS suppresses the derived sensitivity directly, without specifying a noise model, at 1.35–$2.06\times$ training cost per optimization step and no inference cost. Which route to take is a design choice for the target hardware. Whether either is needed also depends on the expected mismatch: at $\mathrm{CV} = 0.05$, Clean SHD-FC models still lose 4.3–5.4 pp, whereas SHD-RSNN loses 1.1–2.2 pp and SSC and DVS-Gesture are essentially unaffected, so well-calibrated deployments with small mismatch may not require additional robustness training. Both routes complement calibration and hardware-in-the-loop training, which compensate residual errors after mapping.

6.1 Limitations

All results are controlled simulations under hardware-grounded mismatch models; we provide no on-chip or hardware-in-the-loop validation, and the simulations omit effects such as readout noise and temperature dependence beyond a global multiplier. The worst-case theorem is a sufficient bound for feedforward networks under surrogate linearization, the typical-case diagnostic rests on modeling assumptions, and the recurrent and delay-learning settings are empirical extensions. MARS increases training cost (1.35–$2.06\times$ per optimization step and 1.13–$2.30\times$ peak memory), does not address weight quantization (6-bit PTQ lowers its SHD-RSNN nominal accuracy by 1.7 pp; Appendix O), and compresses generic temporal memory. Finally, our evaluation covers five small-to-medium settings; the benefit on Transformer-based SNN backbones and large-scale event datasets has not been established.

Broader impact. This work aims to make SNNs with learnable dynamics more reliable on analog and mixed-signal hardware. More robust event-based sensing could also serve monitoring applications, whose deployment should respect the privacy and safety constraints of the target setting.

7 Conclusion

We identify membrane sensitivity Ψ as the first-order channel through which learnable time constants create deployment fragility in SNNs, and derive an online recursion that makes it measurable during training. Regularizing it with MARS reduces drops of up to 27 pp to below 0.8 pp on the temporally dependent settings while preserving nominal accuracy, and the same quantity explains why noise augmentation is also effective. Robust neuromorphic deployment can thus shape the learned dynamics before deployment, not only adapt models to hardware afterwards.

Acknowledgments and Disclosure of Funding

The research was supported by the National Science and Technology Council of Taiwan (Grant No. NSTC 113-2221-E-006-138, NSTC 113-2410-H-006-118, NSTC 114-2221-E-006-205, NSTC 114-2410-H-006-122-MY3, NSTC 115-2221-E-006-237), in part by Higher Education Sprout Project, Ministry of Education to the Headquarters of University Advancement at National Cheng Kung University (NCKU), AI Sustainable Development Goals Future Star Project from Miin Wu School of Computing, NCKU, and Taiwan Semiconductor Research Institute-Joint Development Project (JDP113-Y1-007 and JDP114-Y1-009). The authors declare no competing interests.

References

[1] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7243–7252, 2017.

[2] G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, 8(3):643–674, 1996. doi: 10.1162/neco.1996.8.3.643.

[3] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS)*, pages 929–947, 2024. doi: 10.1145/3620665.3640366.

[4] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[5] C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1): 108–116, 1995. doi: 10.1162/neco.1995.7.1.108.

[6] J. Büchel, D. Zendrikov, S. Solinas, G. Indiveri, and D. R. Muir. Supervised training of spiking neural networks for robust deployment on mixed-signal neuromorphic processors. *Scientific Reports*, 11(1):23376, 2021. doi: 10.1038/s41598-021-02779-x.

[7] A. Castagnetti, A. Pegatoquet, and B. Miramond. Trainable quantization for speedy spiking neural networks. *Frontiers in Neuroscience*, 17:1154241, 2023. doi: 10.3389/fnins.2023.1154241.

[8] B. Cramer, S. Billaudelle, S. Kanya, A. Leibfried, A. Grübl, V. Karasenko, C. Pehle, K. Schreiber, Y. Stradmann, J. Weis, J. Schemmel, and F. Zenke. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4):e2109194119, 2022. doi: 10.1073/pnas.2109194119.

[9] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke. The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33 (7):2744–2757, 2022. doi: 10.1109/TNNLS.2020.3044364.

[10] H. Drucker and Y. LeCun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992. doi: 10.1109/72.165600.

[11] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023. doi: 10.1109/JPROC.2023.3308088.

[12] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian. Deep residual learning in spiking neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 21056–21069, 2021.

[13] 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 (ICCV)*, pages 2661–2671, 2021.

[14] W. Fang, Y. Chen, J. Ding, Z. Yu, T. Masquelier, D. Chen, L. Huang, H. Zhou, G. Li, and Y. Tian. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023. doi: 10.1126/sciadv.adi1480.

[15] C. Ge, Y. Peng, Z. Li, Q. Kang, X. Fu, X. Li, Q. Zhang, J. Ren, and Z.-J. Zha. Fractional-order spiking neural network. In *International Conference on Learning Representations (ICLR)*, 2026.

[16] I. Hammouamri, I. Khalfaoui-Hassani, and T. Masquelier. Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In *International Conference on Learning Representations (ICLR)*, 2024.

[17] J. Hoffman, D. A. Roberts, and S. Yaida. Robust learning with Jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18 (187):1–30, 2018.

[19] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saïghi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5:73, 2011. doi: 10.3389/fnins.2011.00073.

[20] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018. doi: 10.1109/CVPR.2018.00286.

[21] I. Khalfaoui-Hassani, T. Pellegrini, and T. Masquelier. Dilated convolution with learnable spacings. In *International Conference on Learning Representations (ICLR)*, 2023.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[23] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[24] G. Lenz, K. Chaney, S. B. Shrestha, O. Oubari, S. Picaud, and G. Zarrella. Tonic: Event-based datasets and transformations. Zenodo, 2021. doi: 10.5281/zenodo.5079801.

[25] C. Li, L. Ma, and S. Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022. doi: 10.3389/fnins.2022.918793.

[26] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[27] E. O. Neftci, H. Mostafa, and F. Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019. doi: 10.1109/MSP.2019.2931595.

[28] C. Pehle, S. Billaudelle, B. Cramer, J. Kaiser, K. Schreiber, Y. Stradmann, J. Weis, A. Leibfried, E. Müller, and J. Schemmel. The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 16:795876, 2022. doi: 10.3389/fnins.2022.795876.

[29] N. Perez-Nieves, V. C. H. Leung, P. L. Dragotti, and D. F. M. Goodman. Neural heterogeneity promotes robust learning. *Nature Communications*, 12(1):5791, 2021. doi: 10.1038/s41467-021-26022-3.

[30] M. A. Petrovici, B. Vogginger, P. Müller, O. Breitwieser, M. Lundqvist, L. Muller, M. Ehrlich, A. Destexhe, A. Lansner, R. Schüffny, J. Schemmel, and K. Meier. Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLOS ONE*, 9(10):e108590, 2014. doi: 10.1371/journal.pone.0108590.

[31] T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M. A. Petrovici, M. Schmuker, D. Brüderle, J. Schemmel, and K. Meier. Six networks on a universal neuromorphic computing substrate. *Frontiers in Neuroscience*, 7:11, 2013. doi: 10.3389/fnins.2013.00011.

[32] A. Queant, U. Rançon, B. R. Cottereau, and T. Masquelier. DelRec: Learning delays in recurrent spiking neural networks. *arXiv preprint arXiv:2509.24852*, 2025.

[33] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 833–840, 2011.

[34] A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 1660–1669, 2018. doi: 10.1609/aaai.v32i1.11504.

[35] S. Schmitt, J. Klähn, G. Bellec, A. Grübl, M. Güttler, A. Hartel, S. Hartmann, D. Husmann, K. Husmann, S. Jeltsch, V. Karasenko, M. Kleider, C. Koke, A. Kononov, C. Mauch, E. Müller, P. Müller, J. Partzsch, M. A. Petrovici, S. Schiefer, S. Scholze, V. Thanasoulis, B. Vogginger, R. Legenstein, W. Maass, C. Mayr, R. Schüffny, J. Schemmel, and K. Meier. Neuromorphic hardware in the loop: Training a deep spiking network on the BrainScaleS wafer-scale system. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2227–2234. IEEE, 2017. doi: 10.1109/IJCNN.2017.7966125.

[36] S. B. Shrestha and G. Orchard. SLAYER: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[38] J. Sokolić, R. Giryes, G. Sapiro, and M. R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017. doi: 10.1109/TSP.2017.2708039.

[39] P. Warden. Speech Commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[40] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018. doi: 10.3389/fnins.2018.00331.

[41] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019. doi: 10.1609/aaai.v33i01.33011311.

[42] T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel, C. Pehle, S. Billaudelle, G. Kiene, C. Mauch, J. Schemmel, K. Meier, and M. A. Petrovici. Demonstrating advantages of neuromorphic computation: A pilot study. *Frontiers in Neuroscience*, 13:260, 2019. doi: 10.3389/fnins.2019.00260.

[43] B. Yin, F. Corradi, and S. M. Bohté. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10):905–913, 2021. doi: 10.1038/s42256-021-00397-w.

[44] Y. Yoshida and T. Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[45] F. Zenke and T. P. Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4):899–925, 2021. doi: 10.1162/neco_a_01367.

[46] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11062–11070, 2021. doi: 10.1609/aaai.v35i12.17320.

A Proof of Theorem 1

We restate the theorem and give the full proof for the surrogate-linearized dynamics.

Theorem (Restatement of Theorem 1)**.** *Under (A1)–(A3), the worst-case squared surrogate-linearized output perturbation satisfies*

$$\max_{\|\varepsilon\|_\infty \le \varepsilon_{\max}} \|\delta V_T^{(L)}\|_2^2 \;\le\; C_{\mathrm{arch}} \sum_{l=1}^{L-1} \big(p^{(l)}\, \varepsilon_{\max}^{(l)}\big)^2 \cdot \frac{1}{T} \sum_{t=1}^{T} \|\Psi_t^{(l)}\|_2^2 \;+\; O(\varepsilon_{\max}^3),$$

where $C_{\mathrm{arch}} = T^2(L-1)(L_\sigma \|W^{(L)}\|_{\mathrm{op}})^2$ *and* $p^{(l)} = \prod_{k=l+1}^{L-1} \lambda_r^{(k)}$.

Linearized object and remainder. Throughout, δV and δS denote the first-order (tangent-linear) response of the network in which the derivative of H is replaced by σ', evaluated along the nominal trajectory; this is what we call the surrogate-linearized dynamics. For this response, δV is exactly linear in ε, and the bound holds with no remainder. The $O(\varepsilon_{\max}^3)$ term, with $\varepsilon_{\max} \triangleq \max_l \varepsilon_{\max}^{(l)}$, accounts for the second-order Taylor remainder when the bound is applied to a forward model with a smooth surrogate in place of H; the $O(\cdot)$ terms below are understood in the same sense. Assumption (A3) is used only for the downstream layers $k \ge 2$ in Step 2; the decay of a source layer enters only through $\Psi_t^{(l)}$.

Proof. The proof has five steps.

Step 1: single-layer perturbation under surrogate linearization. At hidden layer l the perturbed decay is $\tilde\beta^{(l)} = \beta^{(l)} + \varepsilon^{(l)}$, where $\varepsilon^{(l)}$ is either a layer-wise scalar or a neuron-specific vector in $\mathbb{R}^{n_l}$. For the surrogate-linearized dynamics, where $\delta S_t \approx \sigma'(V_t - \theta) \odot \delta V_t$, first-order Taylor expansion of the membrane recursion yields

$$\delta V_t^{(l)} \;=\; \Psi_t^{(l)} \odot \varepsilon^{(l)} + O\big(\|\varepsilon^{(l)}\|_\infty^2\big), \tag{15}$$

since in a feedforward layer each neuron's membrane depends only on its own decay ($\odot$ reduces to scalar multiplication in the layer-wise case). Here $\Psi_t^{(l)} = \partial V_t^{(l)}/\partial \beta^{(l)}$ is the sensitivity computed by the online recursion (Section 3.2). Maximizing over $\|\varepsilon^{(l)}\|_\infty \le \varepsilon_{\max}^{(l)}$:

$$\max_{\|\varepsilon^{(l)}\|_\infty \le \varepsilon_{\max}^{(l)}} \|\delta V_t^{(l)}\|_2 \;\le\; \|\Psi_t^{(l)}\|_2\, \varepsilon_{\max}^{(l)} + O\big((\varepsilon_{\max}^{(l)})^2\big). \tag{16}$$

Step 2: derivation of the unrolled layer gain. This step justifies the layer gain $\lambda_r^{(k)}$ used in (11). The denominator $1 - \bar\alpha^{(k)}$ comes from explicitly unrolling the downstream membrane recurrence.

The membrane perturbation at layer $k-1$ enters layer k through the surrogate-linearized spike perturbation. By (A2),

$$\|\delta S_t^{(k-1)}\|_2 = \|\sigma'(V_t^{(k-1)} - \theta) \odot \delta V_t^{(k-1)}\|_2 \le L_\sigma \|\delta V_t^{(k-1)}\|_2.$$

Let $e_t^{(k)}$ be the perturbation in layer k caused by a fixed source layer $l < k$. The downstream membrane recursion gives

$$\|e_t^{(k)}\|_2 \le \bar\alpha^{(k)} \|e_{t-1}^{(k)}\|_2 + L_\sigma \|W^{(k)}\|_{\mathrm{op}} \|e_t^{(k-1)}\|_2, \tag{17}$$

where (A3) gives $|\alpha_{\mathrm{eff}}^{(k)}(t)| \le \bar\alpha^{(k)} < 1$. To see the origin of the denominator, write

$$x_t \triangleq \|e_t^{(k)}\|_2, \qquad b_t \triangleq L_\sigma \|W^{(k)}\|_{\mathrm{op}} \|e_t^{(k-1)}\|_2,$$

so that (17) has the scalar form

$$x_t \le \bar\alpha^{(k)} x_{t-1} + b_t, \qquad x_0 = 0.$$

Expanding this recurrence gives

$$\begin{aligned} x_t &\le b_t + \bar\alpha^{(k)} b_{t-1} + \big(\bar\alpha^{(k)}\big)^2 b_{t-2} + \cdots + \big(\bar\alpha^{(k)}\big)^{t-1} b_1 \\ &= \sum_{\tau=1}^{t} \big(\bar\alpha^{(k)}\big)^{t-\tau} b_\tau. \end{aligned} \tag{18}$$

Thus the perturbation sequence in layer k is bounded by the convolution of the upstream perturbation sequence with the geometric kernel

$$h_j = \big(\bar{\alpha}^{(k)}\big)^j, \qquad j = 0, 1, \ldots.$$

Because $0 \le \bar{\alpha}^{(k)} < 1$, this kernel has finite ℓ_1 norm

$$\|h\|_1 = \sum_{j=0}^{\infty} \big(\bar{\alpha}^{(k)}\big)^j = \frac{1}{1 - \bar{\alpha}^{(k)}}. \tag{19}$$

Substituting the definition of b_τ into (18) yields

$$\|e_t^{(k)}\|_2 \le L_\sigma \|W^{(k)}\|_{\mathrm{op}} \sum_{\tau=1}^{t} (\bar{\alpha}^{(k)})^{t-\tau} \|e_\tau^{(k-1)}\|_2.$$

Young's convolution inequality for sequences then gives the time-energy bound

$$\left(\sum_{t=1}^{T} \|e_t^{(k)}\|_2^2 \right)^{1/2} \le \frac{L_\sigma \|W^{(k)}\|_{\mathrm{op}}}{1 - \bar{\alpha}^{(k)}} \left(\sum_{t=1}^{T} \|e_t^{(k-1)}\|_2^2 \right)^{1/2}, \tag{20}$$

which matches the compact layer gain in (11) after identifying $\lambda_r^{(k)} = L_\sigma \|W^{(k)}\|_{\mathrm{op}} / (1 - \bar{\alpha}^{(k)})$. As $\bar{\alpha}^{(k)}$ approaches 1, the geometric kernel decays more slowly, so past perturbations persist longer and the worst-case temporal amplification grows. Iterating (20) from $l+1$ to $L-1$ yields

$$\sum_{t=1}^{T} \|e_t^{(L-1,l)}\|_2^2 \le (p^{(l)})^2 \sum_{t=1}^{T} \|\delta V_t^{(l)}\|_2^2 + O(\varepsilon_{\max}^3), \tag{21}$$

where $e_t^{(L-1,l)}$ denotes the contribution at layer $L-1$ from source layer l, and the cumulative propagation factor is

$$p^{(l)} \triangleq \prod_{k=l+1}^{L-1} \lambda_r^{(k)}, \qquad p^{(L-1)} = 1.$$

Step 3: IF-readout accumulation. Many SNN classifiers use non-spiking readout neurons that accumulate hidden-layer spikes over time. We analyze the common no-decay IF readout, with $\beta^{(L)} = 1$ and no reset; this choice preserves perturbations across timesteps and therefore upper-bounds a leaky readout with an unperturbed decay that is read out at $t = T$. Time-summed and learnable leaky readouts are treated after the proof.

$$\delta V_T^{(L)} \;=\; W^{(L)} \sum_{t=1}^{T} \delta S_t^{(L-1)} \;=\; W^{(L)} \sum_{t=1}^{T} \sigma'(V_t^{(L-1)} - \theta) \odot \delta V_t^{(L-1)}. \tag{22}$$

Taking $\|\cdot\|_2$ and using (A2):

$$\|\delta V_T^{(L)}\|_2 \;\le\; L_\sigma \|W^{(L)}\|_{\mathrm{op}} \sum_{t=1}^{T} \|\delta V_t^{(L-1)}\|_2.$$

Squaring and applying Cauchy–Schwarz over time:

$$\|\delta V_T^{(L)}\|_2^2 \;\le\; (L_\sigma \|W^{(L)}\|_{\mathrm{op}})^2 \cdot T \sum_{t=1}^{T} \|\delta V_t^{(L-1)}\|_2^2. \tag{23}$$

Step 4: cross-layer superposition in time-energy. The total perturbation at layer $L-1$ is the first-order sum of the contributions induced by perturbing each hidden source layer:

$$\delta V_t^{(L-1)} = \sum_{l=1}^{L-1} e_t^{(L-1,l)} + O(\varepsilon_{\max}^2).$$

For each timestep, Cauchy–Schwarz over the $L-1$ source layers gives the generic inequality $\|\sum_{l=1}^{L-1} a_l\|_2^2 \le (L-1)\sum_{l=1}^{L-1}\|a_l\|_2^2$. Applying it to $a_l = e_t^{(L-1,l)}$,

$$\|\delta V_t^{(L-1)}\|_2^2 \le (L-1)\sum_{l=1}^{L-1}\|e_t^{(L-1,l)}\|_2^2 + O(\varepsilon_{\max}^3).$$

Summing over time and using (21) for each source layer yields

$$\sum_{t=1}^{T}\|\delta V_t^{(L-1)}\|_2^2 \le (L-1)\sum_{l=1}^{L-1}(p^{(l)})^2\sum_{t=1}^{T}\|\delta V_t^{(l)}\|_2^2 + O(\varepsilon_{\max}^3).$$

Finally, applying the squared time-summed form of the single-layer perturbation bound (16) gives

$$\sum_{t=1}^{T}\|\delta V_t^{(L-1)}\|_2^2 \le (L-1)\sum_{l=1}^{L-1}(p^{(l)}\varepsilon_{\max}^{(l)})^2\sum_{t=1}^{T}\|\Psi_t^{(l)}\|_2^2 + O(\varepsilon_{\max}^3). \tag{24}$$

Step 5: assembly. Substituting (24) into (23):

$$\begin{aligned}\|\delta V_T^{(L)}\|_2^2 &\le (L_\sigma\|W^{(L)}\|_{\mathrm{op}})^2\cdot T(L-1)\sum_{l=1}^{L-1}(p^{(l)}\varepsilon_{\max}^{(l)})^2\sum_{t=1}^{T}\|\Psi_t^{(l)}\|_2^2 + O(\varepsilon_{\max}^3)\\ &= \underbrace{T^2(L-1)(L_\sigma\|W^{(L)}\|_{\mathrm{op}})^2}_{C_{\mathrm{arch}}}\cdot\sum_{l}(p^{(l)}\varepsilon_{\max}^{(l)})^2\cdot\frac{1}{T}\sum_{t}\|\Psi_t^{(l)}\|_2^2 + O(\varepsilon_{\max}^3),\end{aligned}$$

which matches the stated bound. □

Leaky readout with learnable decay. In the snnTorch SHD, DVS-Gesture, and CIFAR-10 networks, the output layer is a no-reset leaky layer, $V_t^{(L)} = \beta_L V_{t-1}^{(L)} + W^{(L)}S_t^{(L-1)}$ with a learnable and perturbed $\beta_L \in [0,1)$, and decisions use the time-summed membrane $Z \triangleq \sum_{t=1}^{T} V_t^{(L)}$. We treat this layer as a no-reset layer whose time sum is the readout. Unrolling gives $Z = \sum_{s=1}^{T} c_s\, W^{(L)}S_s^{(L-1)}$ with $c_s = \sum_{t=s}^{T}\beta_L^{t-s} \in [1, (1-\beta_L)^{-1}]$. If β_L is unperturbed, Step 3 applies to δZ with the additional factor $\max_s c_s^2 \le (1-\beta_L)^{-2}$ in C_{arch}; this factor is ≈ 1.02 for the learned $\beta_L = 0.01$ on SHD and at most 1.17 (DVS-Gesture) and 1.78 (CIFAR-10) over all trained models. If β_L is perturbed by $\varepsilon^{(L)}$ with $\|\varepsilon^{(L)}\|_\infty \le \varepsilon_{\max}^{(L)}$, then to first order δZ gains the direct term $\sum_{t=1}^{T}\Psi_t^{(L)}\odot\varepsilon^{(L)}$, where $\Psi_t^{(L)} = V_{t-1}^{(L)} + \beta_L\Psi_{t-1}^{(L)}$ is the no-reset sensitivity of Appendix B (the output layer is feedforward, so this also holds for per-neuron $\varepsilon^{(L)}$). Cauchy–Schwarz over time bounds its squared norm by $T^2(\varepsilon_{\max}^{(L)})^2\,\frac{1}{T}\sum_t\|\Psi_t^{(L)}\|_2^2$. Treating it as an L-th source in Step 4, so that Cauchy–Schwarz runs over L rather than $L-1$ terms, gives

$$\begin{aligned}\|\delta Z\|_2^2 \;\le\; L\Bigg[&\frac{T^2(L_\sigma\|W^{(L)}\|_{\mathrm{op}})^2}{(1-\beta_L)^2}\sum_{l=1}^{L-1}\big(p^{(l)}\varepsilon_{\max}^{(l)}\big)^2\,\frac{1}{T}\sum_{t=1}^{T}\|\Psi_t^{(l)}\|_2^2\\ &+\; T^2\big(\varepsilon_{\max}^{(L)}\big)^2\,\frac{1}{T}\sum_{t=1}^{T}\|\Psi_t^{(L)}\|_2^2\Bigg] \;+\; O(\varepsilon_{\max}^3).\end{aligned}$$

The bound thus keeps the form of Theorem 1: the constant gains the factor $(1-\beta_L)^{-2}$, the factor $L-1$ becomes L, and the output layer's own sensitivity energy, which MARS regularizes in these settings, enters as one more source. In the typical-case estimate of Appendix L, the same source term is added and $\tilde{C}_{\mathrm{arch}}$ gains the same factor $(1-\beta_L)^{-2}$. The SSC readout has no learnable decay and is not perturbed, so only the unperturbed case applies there, with its own accumulation weights c_s.

B Full sensitivity derivations

We derive the online recursion for each reset mechanism and state the recurrent extension used to motivate our recurrent stress tests. The feedforward bounds in the main text do not rely on a recurrent-specific stability theorem.

No-reset (NR). With $V_t = \beta V_{t-1} + I_t$, $\partial I_t/\partial\beta = 0$ (the input current depends on the previous layer's β, not the local one), so $\Psi_t^{(\mathrm{NR})} = V_{t-1} + \beta\Psi_{t-1}^{(\mathrm{NR})}$.

Subtract-reset (SR). $V_t = \beta V_{t-1} + I_t - \theta S_{t-1}$ with $S_{t-1} = H(V_{t-1} - \theta)$ replaced by σ' in training:

$$\Psi_t^{(\mathrm{SR})} = V_{t-1} + (\beta - \theta\sigma'(V_{t-1} - \theta)) \odot \Psi_{t-1}^{(\mathrm{SR})},$$

giving effective decay $\alpha_{\mathrm{eff}}^{(\mathrm{SR})}(t) = \beta - \theta\sigma'(V_{t-1} - \theta)$.

Hard-reset (HR). $V_t = \beta(\mathbf{1} - S_{t-1}) \odot V_{t-1} + I_t$, matching (2). For feedforward local β, $\partial I_t/\partial\beta = 0$, so

$$\Psi_t^{(\mathrm{HR})} = (\mathbf{1} - S_{t-1}) \odot V_{t-1} + \beta\big((\mathbf{1} - S_{t-1}) - V_{t-1} \odot \sigma'(V_{t-1} - \theta)\big) \odot \Psi_{t-1}^{(\mathrm{HR})},$$

with an additional $\partial I_t/\partial\beta$ term in recurrent layers.

Verifying assumption (A3) in the experimental settings. In the arctan surrogate used for SHD, DVS-Gesture, and CIFAR-10 (snnTorch `ATan`), $\sigma'(x) = (\nu/2)/(1 + (\pi\nu x/2)^2)$ with slope $\nu = 2$, so $L_\sigma = \nu/2 = 1$. Hence for subtract-reset layers with $\theta = 1$ and $\beta \in (0, 1)$,

$$\alpha_{\mathrm{eff}}^{(\mathrm{SR})}(t) = \beta - \theta\sigma'(V_{t-1} - \theta) \in [\beta - 1, \beta] \subset (-1, 1),$$

which directly verifies (A3). In SSC's triangle surrogate with width parameter $\omega = 1$, $\sigma'(x) \in [0, 1]$, so the same subtract-reset argument gives $\alpha_{\mathrm{eff}}^{(\mathrm{SR})}(t) \in [\beta - 1, \beta] \subset (-1, 1)$. For hard-reset layers, $|\alpha_{\mathrm{eff}}^{(\mathrm{HR})}(t)| = \beta\,|(1 - S_{t-1}) - V_{t-1}\sigma'(V_{t-1} - \theta)| \le c_\sigma\beta$ with $c_\sigma = \sup_v \max\{v\,\sigma'(v-1)\mathbb{1}[v > 1], |1 - v\,\sigma'(v-1)|\mathbb{1}[v \le 1]\} \approx 1.025$ for $\theta = 1$, so (A3) holds whenever $\beta < 1/c_\sigma \approx 0.976$. All trained CIFAR-10 hard-reset layers have $\beta \le 0.82$, giving $\bar\alpha \le 0.84$; consistently, the largest per-neuron effective decay observed on test data (seed 42) is 0.837. All trained DVS-Gesture hard-reset layers have $\beta \le 0.906 < 0.976$, giving $\bar\alpha \le 0.928$; we did not record effective decays on DVS-Gesture test data.

Recurrent extension. For an RSNN layer with recurrent weight W_{rec}, the input contains a feedback term $I_t = Wx_t + W_{\mathrm{rec}}S_{t-1}$. The dependence of S_{t-1} on the membrane state introduces the additional sensitivity contribution $W_{\mathrm{rec}}\sigma'(V_{t-1} - \theta)\Psi_{t-1}$. In the no-reset case this gives

$$\Psi_t^{(l)} \;=\; V_{t-1}^{(l)} \;+\; \big(\beta^{(l)}\,\mathbf{I}_{n_l} \;+\; W_{\mathrm{rec}}^{(l)}\,\mathrm{diag}(\sigma'(V_{t-1}^{(l)} - \theta))\big)\,\Psi_{t-1}^{(l)}, \tag{25}$$

so the scalar decay factor is replaced by a state-dependent matrix. For subtract- or hard-reset recurrent layers, the same recurrent term is added to the corresponding reset-dependent feedforward recursion above.

Proposition 1 (Sufficient recurrent contraction condition)**.** *Consider a recurrent hidden layer whose surrogate-linearized local sensitivity recursion can be written as*

$$\Psi_t^{(l)} \;=\; d_t^{(l)} \;+\; J_t^{(l)}\Psi_{t-1}^{(l)},$$

where $J_t^{(l)} \in \mathbb{R}^{n_l \times n_l}$ is the state-dependent recurrent Jacobian. If there exists an induced matrix norm $\|\cdot\|_\star$ and a constant $\bar\gamma^{(l)} < 1$ such that $\|J_t^{(l)}\|_\star \le \bar\gamma^{(l)}$ for all timesteps, then

$$\|\Psi_t^{(l)}\|_\star \;\le\; \sum_{\tau=1}^{t} \big(\bar\gamma^{(l)}\big)^{t-\tau}\|d_\tau^{(l)}\|_\star,$$

If $\|\cdot\|_\star = \|\cdot\|_2$ (spectral norm), the downstream unrolling in Step 2 of Appendix A holds with $1 - \bar\alpha^{(l)}$ replaced by $1 - \bar\gamma^{(l)}$. For layer-wise or global (scalar) ε, Step 1 also holds with this Ψ, so Theorem 1 extends to recurrent layers that satisfy the condition. For neuron-specific ε, $\delta V \ne \Psi \odot \varepsilon$ in recurrent layers, because W_{rec} couples neurons, and the extension does not apply. For the no-reset recurrent case, $J_t^{(l)} = \beta^{(l)}I + W_{\mathrm{rec}}^{(l)}\mathrm{diag}(\sigma'(V_{t-1}^{(l)} - \theta))$; for subtract- and hard-reset recurrent layers, $J_t^{(l)}$ is the corresponding reset-dependent diagonal term from the feedforward recursion plus the same recurrent contribution.

Proof. Apply the induced norm and submultiplicativity:

$$\|\Psi_t^{(l)}\|_\star \le \|d_t^{(l)}\|_\star + \|J_t^{(l)}\|_\star\|\Psi_{t-1}^{(l)}\|_\star \le \|d_t^{(l)}\|_\star + \bar\gamma^{(l)}\|\Psi_{t-1}^{(l)}\|_\star.$$

Unrolling this scalar recurrence gives the stated geometric series bound, exactly as in Appendix A. □

C Online MARS algorithm

Algorithm 1 Online MARS recursion for one layer. The algorithm adds a single sensitivity state Ψ per layer and accumulates $\|\Psi_t\|_2^2$ during the same forward pass used to compute the task loss. No second-order optimization or extra temporal buffer is required.

Require: decay β, threshold θ, inputs $\{I_t\}_{t=1}^{T}$, reset type $\in \{\text{NR}, \text{SR}, \text{HR}\}$

$V_0 \leftarrow \mathbf{0}$, $\Psi_0 \leftarrow \mathbf{0}$, $\mathcal{R}_{\text{MARS}}^{(l)} \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Compute V_t and S_{t-1} via the LIF forward dynamics (no reset, (1), or (2))
  if no reset **then**
    $\Psi_t \leftarrow V_{t-1} + \beta\, \Psi_{t-1}$
  else if subtract reset **then**
    $\Psi_t \leftarrow V_{t-1} + (\beta - \theta\, \sigma'(V_{t-1} - \theta)) \odot \Psi_{t-1}$
  else {hard reset}
    $\Psi_t \leftarrow (\mathbf{1} - S_{t-1}) \odot V_{t-1} + \beta\big((\mathbf{1} - S_{t-1}) - V_{t-1} \odot \sigma'(V_{t-1} - \theta)\big) \odot \Psi_{t-1}$
  end if
  $\mathcal{R}_{\text{MARS}}^{(l)} \leftarrow \mathcal{R}_{\text{MARS}}^{(l)} + \|\Psi_t\|_2^2$
end for
return $(1/T)\, \mathcal{R}_{\text{MARS}}^{(l)}$

D Detailed experimental setup

Architectures. SHD-FC is a 3-layer fully connected SNN with two 256-unit LIF hidden layers and a no-reset leaky output layer. The recurrent SHD setting uses two 256-unit recurrent LIF hidden layers and the same no-reset leaky output. The SSC setting uses the DelRec architecture [32]: 3 hidden recurrent-delay PLIF layers $\times$ 256 units, with both learnable β and learnable axonal delays trained via differentiable delay interpolation [16], followed by a non-spiking linear readout. DVS-Gesture uses a VGG-style convolutional SNN (channels 64–128–256–256–512, 3×3 kernels, 1024-unit FC head) with hard-reset hidden LIF layers and a no-reset leaky output. CIFAR-10 uses a spiking ResNet-18 (512-d final features) with hard-reset hidden LIF layers and a no-reset leaky output on repeated frames. In the snnTorch SHD/DVS-Gesture/CIFAR-10 implementations, learnable β is enabled for every LIF layer, including the no-reset leaky output layer used as the readout and included in β-perturbation. In SSC, learnable β is used in the DelRec recurrent-delay PLIF layers, while the final readout is a non-spiking linear layer. We use standard-width architectures rather than larger backbones [43, 29], since this gives a controlled testbed for the robustness–accuracy trade-off; scaling MARS to higher-capacity backbones is a natural extension.

Training hyperparameters. For SHD, DVS-Gesture, and CIFAR-10, we train for 50 epochs with Adam [22], using learning rate 2×10^{-3} on SHD and 10^{-3} on DVS-Gesture/CIFAR-10, arctan surrogate slope $\nu = 2.0$, gradient clipping at norm 1, and cosine annealing on DVS-Gesture/CIFAR-10. For SSC, we follow the DelRec protocol with 100 epochs, a triangle surrogate, Adam for weights (10^{-3}) and delay-position parameters (5×10^{-2}), and one-cycle schedules. All learnable β are initialized at 0.5; for SSC this corresponds to initializing the PLIF time constant at $\tau = 2$. MARS uses $\lambda = 5\times10^{-4}$ in the main experiments; the full sweep is in Appendix G. The implementation normalizes (9) differently from the displayed sum: for each regularized layer it averages $(\Psi_{t,i}^{(l)})^2$ over neurons i and batch samples (rather than summing over neurons), averages over timesteps (the $t = 0$ term, where $\Psi = \mathbf{0}$, is excluded for reset layers and contributes zero for the no-reset output layer), and then averages, rather than sums, over the regularized layers, which include a learnable leaky output layer where present; λ refers to this normalized form.

Baseline controls. All controls in Table 3 use the SHD architectures, data pipeline, 50 training epochs, learning rate 2×10^{-3}, and the same five seeds as the main runs. β-L2 adds λ_β times the mean squared learnable β to the loss, with $\lambda_\beta = 10^{-3}$ (SHD-FC) and 10^{-1} (SHD-RSNN) selected on a validation pilot. The hard range projects every learnable β onto $[0.1, 0.9]$ after each optimizer step. Fixed homogeneous β keeps every decay at its initial value 0.5; fixed heterogeneous β draws a

per-neuron decay $0.5(1 + 0.2z)$, $z \sim \mathcal{N}(0, 1)$, clipped to $[0.1, 0.9]$, once per seed and keeps it fixed. Recurrent DropConnect masks recurrent weights with probability 0.05 during training (selected on a validation pilot). Weight decay uses Adam with coefficient 10^{-3}, and spectral normalization is applied to every linear layer of SHD-FC.

Mismatch evaluation. Learnable decays are scalars per layer in all trained checkpoints. For per-neuron mismatch, each scalar is expanded to one value per neuron (one per channel for convolutional layers) before perturbation; layer-wise drift draws one multiplier per layer, and global drift one multiplier for all layers. For SSC, the PLIF decay $\beta = 1 - \mathrm{sigmoid}(w)$ is perturbed in β and mapped back to w. Draws are deterministic functions of the setting, training seed, mismatch structure, and draw index, so all methods of a seed see identical draws, and the CV sweep rescales the same draws. Readout layers with a learnable decay are perturbed as well; learned SSC delays are perturbed only where stated.

E Full main-result numbers

Table 4: Nominal accuracy (mean±SD over 5 training seeds) and accuracy drop under per-neuron β mismatch at $\mathrm{CV} = 0.2$ (mean and 95% percentile bootstrap confidence interval across seeds; $10{,}000$ resamples). Mismatch draws are averaged within each checkpoint first: 200 draws per checkpoint for SHD and 10 otherwise.

Setting	Metric	Clean	Noise Aug	MARS
SHD-FC	Nominal acc. (%)	76.9±2.7	76.2±1.1	77.0±1.7
	Drop @ CV= 0.2 (pp)	27.3 [24.5, 30.0]	1.4 [0.9, 1.8]	0.4 [0.1, 0.6]
SHD-RSNN	Nominal acc. (%)	85.1±1.3	85.5±1.3	84.8±1.6
	Drop @ CV= 0.2 (pp)	11.7 [10.0, 13.7]	0.9 [0.5, 1.2]	0.3 [0.0, 0.6]
SSC	Nominal acc. (%)	82.1±0.1	82.2±0.2	82.1±0.2
	Drop @ CV= 0.2 (pp)	1.0 [1.0, 1.1]	0.1 [0.0, 0.2]	0.1 [0.0, 0.2]
DVS-Gesture	Nominal acc. (%)	84.5±4.4	85.7±2.6	87.0±2.6
	Drop @ CV= 0.2 (pp)	0.8 [−0.2, 1.8]	−0.3 [−0.5, 0.0]	−0.2 [−0.6, 0.1]
CIFAR-10	Nominal acc. (%)	90.2±0.3	90.4±0.2	90.2±0.2
	Drop @ CV= 0.2 (pp)	0.1 [0.0, 0.2]	0.2 [0.1, 0.2]	0.0 [0.0, 0.1]

F Per-seed results and paired differences

Table 5 lists the per-neuron drop at $\mathrm{CV} = 0.2$ of every training seed, and Table 6 reports paired differences between methods for all three mismatch structures. Each difference is computed per training seed with shared mismatch draws and then summarized by its mean, a 95% percentile bootstrap confidence interval over seeds ($10{,}000$ resamples), the number of seeds in which the first method has the lower drop, and the exact two-sided sign-flip p-value. With five seeds the smallest attainable p-value is 0.0625, and a bootstrap over five values has only 126 distinct resamples, so percentile intervals are coarse and tend to undercover; we therefore read these entries as effect sizes rather than significance tests.

G Hyperparameter sensitivity

Table 7 reports the SHD-FC sweep $\lambda \in \{10^{-4}, 5{\times}10^{-4}, 10^{-3}, 5{\times}10^{-3}, 10^{-2}\}$ with five seeds per value. The sweep uses a standalone configuration: layer-wise evaluation (10 draws), seeds 42/52/62/72/82, no input augmentation (hence nominal accuracies of 71–74%), no stored checkpoints, and no Clean arm. Within the sweep, the drop at $\mathrm{CV} = 0.2$ decreases monotonically with λ, from 4.2 pp at 10^{-4} to 0.2 pp at 10^{-2}, whereas nominal accuracy varies non-monotonically; $\lambda = 5{\times}10^{-4}$, used in the main experiments, has the highest nominal accuracy. The robustness–accuracy operating point therefore depends on λ, but the qualitative effect does not require a finely tuned value.

Table 5: Per-seed accuracy drop (pp) under per-neuron β mismatch at CV $= 0.2$ (same draws as Table 4).

Setting	Method	seed 42	seed 52	seed 72	seed 92	seed 102
SHD-FC	Clean	28.3	31.6	26.8	27.9	22.2
	Noise Aug	1.2	1.7	0.5	1.6	2.0
	MARS	0.8	0.1	0.3	0.6	0.0
SHD-RSNN	Clean	11.0	11.0	15.5	12.2	9.0
	Noise Aug	1.2	1.0	0.4	1.4	0.4
	MARS	0.2	0.4	−0.2	0.4	0.8
SSC	Clean	1.1	1.1	1.0	0.9	1.0
	Noise Aug	0.2	0.2	−0.1	0.1	0.3
	MARS	0.1	0.1	0.3	−0.1	0.2
DVS-Gesture	Clean	2.2	1.2	−0.4	1.7	−0.8
	Noise Aug	0.1	−0.1	0.0	−0.7	−0.6
	MARS	−0.2	0.3	0.0	−0.7	−0.6
CIFAR-10	Clean	0.0	0.0	0.0	0.1	0.3
	Noise Aug	0.1	0.1	0.1	0.3	0.2
	MARS	0.0	0.1	0.0	0.1	0.0

Table 6: Paired differences in accuracy drop (pp) at CV $= 0.2$: mean [95% percentile bootstrap CI, $10{,}000$ resamples], number of seeds with a lower drop for MARS, and exact two-sided sign-flip p. Per-neuron uses the draws of Table 4; layer-wise and global use 100 draws per checkpoint.

		MARS − Noise Aug			MARS − Clean		
Setting	Structure	Mean [CI]	Lower	p	Mean [CI]	Lower	p
SHD-FC	Per-neuron	−1.0 [−1.7, −0.4]	5/5	0.0625	−27.0 [−29.7, −24.2]	5/5	0.0625
	Layer-wise	−0.8 [−1.5, −0.1]	5/5	0.0625	−20.1 [−22.0, −18.5]	5/5	0.0625
	Global	−0.8 [−1.2, −0.3]	5/5	0.0625	−17.0 [−18.5, −15.4]	5/5	0.0625
SHD-RSNN	Per-neuron	−0.6 [−0.9, −0.1]	4/5	0.125	−11.4 [−13.8, −9.4]	5/5	0.0625
	Layer-wise	−0.7 [−1.0, 0.0]	4/5	0.125	−17.4 [−20.6, −14.2]	5/5	0.0625
	Global	−0.9 [−1.6, 0.0]	4/5	0.125	−18.3 [−20.5, −15.8]	5/5	0.0625
SSC	Per-neuron	0.0 [−0.2, 0.2]	4/5	0.875	−0.9 [−1.0, −0.8]	5/5	0.0625
	Layer-wise	0.1 [−0.1, 0.2]	1/5	0.562	−3.0 [−4.0, −2.1]	5/5	0.0625
	Global	−0.1 [−0.5, 0.1]	2/5	0.750	−3.5 [−3.9, −3.0]	5/5	0.0625
DVS-Gesture	Per-neuron	0.0 [−0.1, 0.3]	4/5	1.000	−1.0 [−2.1, 0.0]	3/5	0.250
	Layer-wise	0.7 [−0.6, 2.3]	3/5	0.562	−2.2 [−4.5, −0.5]	4/5	0.125
	Global	2.4 [−0.4, 5.5]	2/5	0.312	−0.8 [−4.3, 2.2]	2/5	0.750
CIFAR-10	Per-neuron	−0.1 [−0.2, −0.1]	5/5	0.0625	−0.1 [−0.2, 0.0]	4/5	0.250
	Layer-wise	−0.1 [−0.1, 0.0]	5/5	0.0625	0.0 [−0.1, 0.0]	4/5	0.375
	Global	−0.1 [−0.1, −0.1]	5/5	0.0625	0.0 [−0.1, 0.0]	3/5	0.438

H Additional mismatch distributions

Table 8 replaces the i.i.d. Gaussian factor of the per-neuron protocol by a mean-preserving log-normal factor, a Student-t_3 factor standardized to unit variance, and equicorrelated Gaussian factors ($\rho \in \{0.5, 0.9\}$, shared component per layer), all at CV $= 0.2$ with the same $[0.01, 0.99]$ clip. These results use a separate evaluator (10 draws per checkpoint, independent of the draws in Table 4); only the two robust-training methods were evaluated. Both keep the accuracy change below 2 pp under every distribution, with MARS between 0.24 and 0.82 pp, so the robustness is not specific to independent Gaussian mismatch.

Table 7: MARS λ sweep on SHD-FC (layer-wise β evaluator, CV $= 0.2$, 10 MC draws; seeds 42/52/62/72/82; training configuration without input augmentation). Mean±SD over 5 seeds; drop = nominal − deployed accuracy (pp). Source: `results/lambda_sweep/shd_fc_lambda_sweep.json` in the code release.

λ	Nominal (%)	Acc. @ CV 0.2 (%)	Drop (pp)	Per-seed drops (pp)
1×10^{-4}	71.8±3.3	67.6±3.1	4.20±1.94	3.36, 6.66, 3.01, 2.13, 5.82
5×10^{-4}	74.4±2.3	71.9±2.7	2.49±0.80	2.29, 1.61, 2.08, 3.71, 2.75
1×10^{-3}	73.8±1.7	71.8±1.0	1.99±1.48	1.96, 1.33, 1.90, 4.39, 0.38
5×10^{-3}	71.1±1.6	70.8±1.9	0.38±0.75	0.44, −0.90, 1.04, 0.79, 0.55
1×10^{-2}	72.3±2.6	72.1±2.7	0.23±0.19	0.17, 0.16, 0.53, 0.01, 0.27

Table 8: Non-Gaussian and correlated β mismatch (per-neuron, CV $= 0.2$; log-normal factors are mean-preserving before clipping, t_3 is standardized to unit variance, ρ = equicorrelation across the neurons of a layer). Accuracy change from nominal (pp, mean±SD over 5 seeds; 10 draws per checkpoint); no Clean arm was run.

Setting	Method	Gaussian	Log-normal	Student-t_3	Corr. ρ=0.5	Corr. ρ=0.9
SHD-FC	Noise Aug	−1.41±0.78	−1.88±0.79	−0.83±0.46	−1.59±0.41	−1.42±0.47
	MARS	−0.24±0.52	−0.81±0.32	−0.40±0.25	−0.49±0.45	−0.69±0.61
SHD-RSNN	Noise Aug	−0.74±0.47	−1.05±0.84	−1.22±0.96	−1.23±0.99	−1.12±0.39
	MARS	−0.34±0.34	−0.33±0.59	−0.62±0.89	−0.30±0.29	−0.82±0.60

I Joint mismatch

Table 9 expands Table 2. Auxiliary mismatch perturbs every weight and the threshold of every hidden neuron with CV $= 0.05$; the joint condition adds per-neuron β mismatch at CV $= 0.2$ with the same β draws as Table 4. Auxiliary mismatch alone costs every method at most 0.34 pp, and it adds at most 0.3 pp to the β-only drop. For SSC, a separate evaluation combines per-neuron β mismatch (CV $= 0.2$), per-weight mismatch and per-layer threshold mismatch (CV $= 0.05$; SpikingJelly's reset kernel requires a scalar threshold), and per-neuron mismatch of the learned recurrent delays (CV $= 0.05$), with 5 draws per checkpoint. The resulting drops are 1.17 ± 0.25 (Clean), 0.21 ± 0.16 (Noise Aug), and 0.23 ± 0.16 pp (MARS).

Table 9: Accuracy drop (pp, mean±SD over 5 seeds) under auxiliary weight/threshold mismatch alone (CV $= 0.05$), per-neuron β mismatch alone (CV $= 0.2$), and both together. 200 draws per checkpoint; the β draws are shared with Table 4.

Setting	Method	Weight + threshold	β only	Joint
SHD-FC	Clean	0.13±0.14	27.35±3.42	27.51±3.42
	Noise Aug	0.13±0.31	1.39±0.60	1.58±0.61
	MARS	0.30±0.31	0.37±0.34	0.57±0.32
SHD-RSNN	Clean	0.16±0.16	11.75±2.41	12.03±2.41
	Noise Aug	0.34±0.38	0.88±0.46	1.13±0.43
	MARS	0.28±0.25	0.32±0.36	0.47±0.42

J Temporal expressivity and learned decays

Metrics. The *task-context horizon* is the smallest retained prefix (or suffix) of the input, in time bins, at which accuracy reaches 95% of full-input accuracy (SHD: $T = 50$ bins; SSC: $T = 250$ bins). It is a task-operational metric of how much ordered input the classifier needs, not an intrinsic time constant of the network. R_k^2 is the linear decodability of the raw input event count k bins in the past from frozen hidden states, a measure of generic trace memory. Destructive controls evaluate the unchanged models on time-reversed inputs, fully time-shuffled inputs (10 permutations), and *rate-repeat* inputs, which replace every frame by the temporal mean and thus keep the total input while removing temporal structure. Reset@k resets all neuron states every k bins.

Results. Tables 10–12 give the full results. The task-context horizon is the same for all methods (one SHD-FC MARS seed needs a 50-bin prefix, hence 42.0 ± 4.5), and all tasks depend strongly on temporal order: shuffling and rate-repeat reduce SSC accuracy from 82% to 8% and 3% for every method. MARS lowers long-lag generic memory (R^2_{20}) while increasing short-lag decodability (R^2_1). Its effect on order sensitivity depends on the architecture: on SHD-FC it is more tolerant of reversal, shuffling, and frequent state resets than Clean, consistent with a shorter integration window in layer 1 ($\beta \approx 0.19$); on SHD-RSNN it is more tolerant of reversal and less tolerant of full shuffling and rate-repeat than the other methods. On SSC, timing jitter of ± 1–10 bins and local shuffling within windows of 2–25 bins change the accuracies of the three methods by amounts that differ by at most 0.8 pp. We read these results as selective temporal compression rather than preserved generic memory.

Learned decays and activity. Robust training contracts the learned β, most strongly for MARS in the first hidden layer (Table 12): SHD-FC 0.19 (Clean 0.84) and SHD-RSNN 0.010–0.096 across seeds, i.e., at or near the 0.01 clip floor, so this recurrent layer integrates over very few steps and temporal context is carried mainly by recurrence and the second layer, which keeps $\beta \approx 0.45$. Learned β is a per-layer scalar in these models, so the spread is across seeds. Mean hidden-layer spike probabilities stay within 0.03–0.14 for all methods; we report activity descriptively and make no energy claim.

Table 10: Task-context horizon (bins) and accuracy (%) under destructive temporal controls. Mean±SD over 5 seeds (SSC horizons have SD 0). SHD: $T = 50$ bins; SSC: $T = 250$ bins.

Setting	Method	Prefix	Suffix	Original	Reversed	Shuffled	Rate-repeat
SHD-FC	Clean	40.0±0.0	50.0±0.0	76.9±2.7	34.9±2.0	40.0±1.2	34.2±1.4
	Noise Aug	40.0±0.0	50.0±0.0	76.2±1.1	62.6±2.5	44.5±1.9	32.0±1.9
	MARS	42.0±4.5	50.0±0.0	77.0±1.7	73.5±2.2	51.4±3.2	20.4±2.3
SHD-RSNN	Clean	40.0±0.0	50.0±0.0	85.1±1.3	37.1±2.3	32.4±3.9	25.9±2.9
	Noise Aug	40.0±0.0	50.0±0.0	85.5±1.3	38.0±4.0	33.1±1.9	25.3±2.2
	MARS	40.0±0.0	50.0±0.0	84.8±1.6	44.8±4.2	27.3±2.2	20.1±3.3
SSC	Clean	150	250	82.1±0.1	9.2±0.2	7.8±0.6	2.7±0.4
	Noise Aug	150	250	82.2±0.2	9.2±0.3	7.9±0.5	2.9±0.9
	MARS	150	250	82.1±0.2	9.3±0.0	8.0±0.6	2.8±0.3

Table 11: Generic trace memory and state-reset tolerance on SHD. R^2_k: linear decodability of the raw event count k bins in the past from frozen hidden states. Reset@k: accuracy (%) when all states are reset every k bins (Reset@50 = original). Mean±SD over 5 seeds.

Setting	Method	R^2_1	R^2_5	R^2_{10}	R^2_{20}	Reset@1	Reset@10	Reset@25
SHD-FC	Clean	0.80±0.01	0.82±0.01	0.75±0.00	0.47±0.01	32.2±2.8	37.3±2.7	60.0±2.6
	Noise Aug	0.92±0.01	0.87±0.00	0.67±0.02	0.16±0.03	54.9±2.3	69.0±0.5	75.5±1.2
	MARS	0.96±0.00	0.79±0.03	0.40±0.05	0.08±0.01	68.5±0.9	75.5±1.9	77.0±2.0
SHD-RSNN	Clean	0.93±0.01	0.90±0.00	0.83±0.01	0.59±0.04	9.2±2.9	33.3±2.6	74.3±1.7
	Noise Aug	0.96±0.00	0.92±0.00	0.76±0.02	0.25±0.03	9.8±4.1	36.8±3.6	69.2±4.0
	MARS	0.96±0.00	0.89±0.01	0.69±0.03	0.20±0.08	9.6±1.2	34.7±2.8	69.9±3.3

K Mechanism association

Post-hoc time-averaged sensitivity energy $\frac{1}{T}\sum_t \|\Psi_t\|_2^2$ was computed for every Clean, Noise Aug, and MARS checkpoint of the three temporally dependent settings. Pooled over these settings (geometric mean of matched ratios), Noise Aug reduces it by $4.67\times$ relative to Clean, MARS by $78.2\times$, and MARS by $16.7\times$ relative to Noise Aug; the ordering MARS $<$ Noise Aug $<$ Clean holds in all 15 matched setting–seed triples. These ratios refer to sensitivity energy, not to the amplitude of Ψ. Table 13 correlates $\log_{10}$ energy with the drop at $\mathrm{CV} = 0.2$ after removing per-setting means, so that raw energies are never compared across architectures. Restricting the analysis to Clean and Noise Aug avoids circularity with the MARS objective. The association is correlational and is partly

Table 12: Learned (per-layer scalar) β of the hidden layers, mean over 5 seeds [min, max], and mean spike probability per hidden layer (mean±SD over 5 seeds, 4 test batches). The readout β is 0.01 for every method and seed.

Setting	Method	β layer 1	β layer 2	Spike prob. layer 1	Spike prob. layer 2
SHD-FC	Clean	0.840 [0.820, 0.849]	0.798 [0.788, 0.804]	0.080±0.002	0.031±0.001
	Noise Aug	0.634 [0.601, 0.655]	0.656 [0.649, 0.668]	0.085±0.003	0.033±0.001
	MARS	0.190 [0.164, 0.228]	0.575 [0.535, 0.635]	0.095±0.007	0.037±0.001
SHD-RSNN	Clean	0.555 [0.507, 0.630]	0.806 [0.785, 0.835]	0.131±0.017	0.037±0.004
	Noise Aug	0.500 [0.487, 0.521]	0.590 [0.567, 0.616]	0.136±0.011	0.031±0.003
	MARS	0.032 [0.010, 0.096]	0.446 [0.377, 0.553]	0.134±0.010	0.040±0.003

driven by the separation between Clean and Noise Aug checkpoints within each setting. Figure 3 shows the raw relation for each setting.

Table 13: Pearson r and Spearman ρ between per-setting-centered $\log_{10}$ sensitivity energy and drop at CV $= 0.2$ (SHD-FC, SHD-RSNN, SSC). All $p < 10^{-8}$.

Drop estimate	Checkpoints	n	Pearson r	Spearman ρ
Per-neuron	Clean + Noise Aug	30	0.94	0.95
Per-neuron	all three methods	45	0.77	0.76
Layer-wise (100 draws)	Clean + Noise Aug	30	0.91	0.93
Layer-wise (100 draws)	all three methods	45	0.77	0.77

L Typical-case diagnostic: modeling assumptions and detailed sketch

This appendix records the modeling assumptions behind the typical-case diagnostic in Section 4.2. It is a detailed sketch of a modeling-assumption-based estimate, not a formal proof. The assumptions below are empirical approximations, not properties of arbitrary SNNs.

With the propagation factor $\tilde{p}^{(l)} \triangleq \prod_{k=l+1}^{L-1} \tilde{\lambda}_r^{(k)}$, defined analogously to (12), the assumptions below lead to the heuristic estimate

$$\mathbb{E}_\varepsilon\big[\|\delta V_T^{(L)}\|_2^2\big] \;\lesssim\; \tilde{C}_{\mathrm{arch}} \sum_{l=1}^{L-1} (\varepsilon_{\max}^{(l)})^2 \left(\tilde{p}^{(l)}\right)^2 \cdot \frac{1}{T} \sum_{t=1}^{T} \big\|\Psi_t^{(l)}\big\|_2^2 \;+\; O(\varepsilon_{\max}^3), \tag{26}$$

with $\tilde{C}_{\mathrm{arch}} \triangleq T^2 (L_\sigma \|W\|_{\mathrm{eff}}^{(L)})^2$, where the T^2 factor follows from Cauchy–Schwarz over time and requires no incoherence assumption (Ingredient 1). The missing $(L-1)$ factor relative to Theorem 1 is absorbed into the cross-layer incoherence assumption (B5); without it, the conservative factor reappears. The expression is a modeling-assumption-based estimate rather than a formal theorem.

Typical-case assumptions. **(B1) Average-direction amplification model.** To model typical rather than worst-case downstream amplification, we compare perturbations against an isotropic prior over unit directions and use Lemma 1. This is a prior on directions, not a claim that the trained network's true perturbation directions are literally uniform on the sphere. **(B2) Temporal accumulation (no incoherence assumption).** We do not assume incoherence across timesteps: both the readout time sum (Ingredient 1) and the within-layer temporal convolution (Ingredient 5) are bounded by Cauchy–Schwarz/Young-type inequalities. **(B3) Weak surrogate-mask/direction coupling.** The overlap between $|\delta V_t|$ and the surrogate-active coordinates $|\hat{\sigma}_t|$, where $\hat{\sigma}_t^{(l)} \triangleq \sigma'(V_t^{(l)} - \theta)$, is not systematically larger than what is already captured by the empirical sparsity factor η. **(B4) Average multiplicative decay proxy.** The effective-decay products are modeled by an exponential proxy with rate $\tilde{\alpha}$: for typical vectors x,

$$\Big\|\Big(\prod_{s=\tau+1}^{t} \alpha_{\mathrm{eff}}(s)\Big) \odot x\Big\|_2 \;\lesssim\; \tilde{\alpha}^{\,t-\tau}\, \|x\|_2.$$

For a neuron-uniform coefficient that is independent across s and of the propagated perturbation, with $\mathbb{E}|\alpha_{\text{eff}}(s)| = \tilde{\alpha}$, this holds with equality in expectation. **(B5) Cross-layer incoherence at the readout.** After downstream propagation to the layer feeding the readout, perturbations originating from different source layers are treated as approximately orthogonal in expectation. This is the extra assumption responsible for dropping the worst-case $(L-1)$ factor from the readout constant.

Under these modeling assumptions, we approximate the conservative ingredients in Theorem 1 by the following typical-case counterparts.

Ingredient 1: temporal accumulation by Cauchy–Schwarz. For the readout sum, Cauchy–Schwarz gives $\|\sum_{t=1}^{T} \hat{\sigma}_t^{(L-1)} \odot \delta V_t^{(L-1)}\|_2^2 \le T \sum_{t=1}^{T} \|\hat{\sigma}_t^{(L-1)} \odot \delta V_t^{(L-1)}\|_2^2$ without any assumption on the perturbation directions. We keep this factor rather than assuming incoherent accumulation, because the perturbation directions are temporally coherent in practice (Appendix M), so this factor is not substantially loose.

Ingredient 2: average-direction weight amplification. Under (B1), for $W \in \mathbb{R}^{m\times n}$ and a random unit-direction vector $u \in \mathbb{S}^{n-1}$ from the isotropic prior, $\mathbb{E}[\|Wu\|^2] = \operatorname{tr}(W^\top W)/n = \|W\|_F^2/n = \|W\|_{\text{eff}}^2$ (Lemma 1). This motivates replacing the worst-case operator norm $\|W\|_{\text{op}}^2$ by the prior-averaged amplification $\|W\|_{\text{eff}}^2$ in both readout and inter-layer steps.

Ingredient 3: weak surrogate-mask coupling summarized by η. Under (B3), the expected overlap between the perturbation direction and the surrogate-active coordinates is summarized by the empirical surrogate sparsity:

$$\mathbb{E}\big[\|\hat{\sigma}_t^{(l)} \odot \delta V_t^{(l)}\|_2^2 \,\big|\, \|\delta V_t^{(l)}\|_2\big] \approx \eta^{(l)} L_\sigma^2 \, \|\delta V_t^{(l)}\|_2^2.$$

This contributes the $\sqrt{\eta}$ factor in $\tilde{\lambda}_r$. The assumption is not literal independence; it is a weak-coupling proxy that discards only large positive alignment beyond what η already captures.

Ingredient 4: average-case decay proxy. Under (B4), products of effective decay coefficients are modeled by the trajectory- and neuron-averaged quantity $\tilde{\alpha}^{(l)} = \frac{1}{Tn_l} \sum_t \|\alpha_{\text{eff}}^{(l)}(t)\|_1$ rather than a worst-case bound $\bar{\alpha}^{(l)} \ge \max_t \|\alpha_{\text{eff}}^{(l)}(t)\|_\infty$. This is a plug-in exponential proxy, not a consequence of Jensen's inequality.

Ingredient 5: assembly. Let $e_t^{(k,l)}$ denote the perturbation at hidden layer k and timestep t that originates from source layer l. Unrolling the single-layer surrogate-linearized recurrence of Appendix A gives $e_t^{(k,l)} = \sum_{\tau=1}^{t} \big(\prod_{s=\tau+1}^{t} \alpha_{\text{eff}}^{(k)}(s)\big) \odot u_\tau$ with $u_\tau = W^{(k)}\big(\hat{\sigma}_\tau^{(k-1)} \odot e_\tau^{(k-1,l)}\big)$. The mask gating the input to layer k is that of layer $k-1$, so Ingredients 2 and 3 give $\mathbb{E}\|u_\tau\|_2^2 \lesssim \eta^{(k-1)} L_\sigma^2 (\|W\|_{\text{eff}}^{(k)})^2 \, \mathbb{E}\|e_\tau^{(k-1,l)}\|_2^2$. By the triangle inequality, (B4), and the weighted Cauchy–Schwarz inequality $\big(\sum_\tau (\tilde{\alpha}^{(k)})^{t-\tau} x_\tau\big)^2 \le \big(\sum_j (\tilde{\alpha}^{(k)})^j\big) \sum_\tau (\tilde{\alpha}^{(k)})^{t-\tau} x_\tau^2$, which needs no temporal incoherence,

$$\mathbb{E}\big[\|e_t^{(k,l)}\|_2^2\big] \;\lesssim\; \frac{\eta^{(k-1)} L_\sigma^2 \big(\|W\|_{\text{eff}}^{(k)}\big)^2}{1-\tilde{\alpha}^{(k)}} \sum_{\tau=1}^{t} \big(\tilde{\alpha}^{(k)}\big)^{t-\tau} \mathbb{E}\big[\|e_\tau^{(k-1,l)}\|_2^2\big].$$

Summing over t and using $\sum_{j\ge 0} (\tilde{\alpha}^{(k)})^j \le (1-\tilde{\alpha}^{(k)})^{-1}$ once more gives

$$\sum_{t=1}^{T} \mathbb{E}\big[\|e_t^{(k,l)}\|_2^2\big] \;\lesssim\; \frac{\eta^{(k-1)} L_\sigma^2 \big(\|W\|_{\text{eff}}^{(k)}\big)^2}{\big(1-\tilde{\alpha}^{(k)}\big)^2} \sum_{t=1}^{T} \mathbb{E}\big[\|e_t^{(k-1,l)}\|_2^2\big].$$

Finally, Ingredient 1, with the mask of layer $L-1$ and Ingredients 2–3 applied to the readout, gives

$$\mathbb{E}\big[\|\delta V_T^{(L)}\|_2^2\big] \;\lesssim\; T\,\eta^{(L-1)} \big(L_\sigma \|W\|_{\text{eff}}^{(L)}\big)^2 \sum_{l=1}^{L-1} \sum_{t=1}^{T} \mathbb{E}\big[\|e_t^{(L-1,l)}\|_2^2\big],$$

where (B5) treats the layer-originated terms as additive in expectation rather than introducing the worst-case extra $(L-1)$ factor from Theorem 1. Along the chain from source layer l, the factors

Table 14: Worst-case λ_r vs. typical-case $\tilde{\lambda}_r$ diagnostics for SHD-FC. $\tilde{\lambda}_r < 1$ is the relaxed diagnostic threshold from Section 4.2. MARS is the only method below this threshold in both hidden layers; Clean and Noise Aug exceed it. Values use the snnTorch arctan derivative ($L_\sigma = 1$). The lif1 values (input weights) do not enter $\tilde{p}^{(l)}$ in (26) and are reported as per-layer diagnostics; the value that enters is that of lif2.

Method	Layer	$\|W\|_{\text{op}}$	$\|W\|_{\text{eff}}$	κ	η	$\bar{\alpha}$	$\tilde{\alpha}$	λ_r	$\tilde{\lambda}_r$
Clean	lif1	33.2	4.14	64	0.008	0.846	0.832	215.2	2.24 ×
	lif2	26.6	3.92	46	0.012	0.775	0.768	118.3	1.82 ×
MARS	lif1	41.9	4.62	82	0.026	0.168	0.163	50.4	**0.89** ✓
	lif2	19.6	3.84	26	0.014	0.526	0.518	41.4	**0.94** ✓
Noise Aug	lif1	32.8	3.76	76	0.017	0.631	0.607	88.8	1.24 ×
	lif2	19.8	3.41	34	0.016	0.637	0.627	54.6	1.16 ×

$\eta^{(l)}, \eta^{(l+1)}, \ldots, \eta^{(L-1)}$ appear once each. Regrouping them so that $\eta^{(k)}$ is attached to the step into layer k gives the per-layer gains

$$\tilde{\lambda}_r^{(k)} = \frac{\sqrt{\eta^{(k)}}\, L_\sigma \, \|W\|_{\text{eff}}^{(k)}}{1 - \tilde{\alpha}^{(k)}},$$

and leaves the source factor $\eta^{(l)} \le 1$, which we drop. Hence

$$\begin{aligned}
\mathbb{E}\big[\|\delta V_T^{(L)}\|_2^2\big] &\lesssim T\big(L_\sigma \|W\|_{\text{eff}}^{(L)}\big)^2 \sum_{l=1}^{L-1} \eta^{(l)} \big(\tilde{p}^{(l)} \varepsilon_{\max}^{(l)}\big)^2 \sum_{t=1}^{T} \|\Psi_t^{(l)}\|_2^2 \\
&\le T\big(L_\sigma \|W\|_{\text{eff}}^{(L)}\big)^2 \sum_{l=1}^{L-1} \big(\tilde{p}^{(l)} \varepsilon_{\max}^{(l)}\big)^2 \sum_{t=1}^{T} \|\Psi_t^{(l)}\|_2^2,
\end{aligned}$$

using $\|\delta V_t^{(l)}\|_2 \le \|\Psi_t^{(l)}\|_2 \varepsilon_{\max}^{(l)}$ at the source. Rewriting the time sum as $\frac{1}{T}\sum_t \|\Psi_t^{(l)}\|_2^2$ yields (26), which is therefore conservative relative to the chain product by the factor $1/\eta^{(l)}$. Since $\tilde{p}^{(l)}$ involves only $\tilde{\lambda}_r^{(k)}$ with $k \ge l+1 \ge 2$, the first-layer value $\tilde{\lambda}_r^{(1)}$, which acts on the unperturbed input weights, does not enter (26); for the two-hidden-layer SHD-FC network only $\tilde{\lambda}_r^{(2)}$ does, and $\tilde{\lambda}_r^{(1)}$ is reported in Table 14 as a per-layer diagnostic. Because (B1) and (B3)–(B5) are modeling assumptions, the diagnostic should be read as an explanatory estimate rather than a worst-case guarantee.

Lemma 1 (Typical-case amplification)**.** *For $W \in \mathbb{R}^{m\times n}$ and a uniform random unit vector $u \in \mathbb{S}^{n-1}$ over the input dimension, $\mathbb{E}[\|Wu\|_2^2] = \|W\|_F^2/n \triangleq \|W\|_{\text{eff}}^2$.*

Proof. $\mathbb{E}[\|Wu\|^2] = \mathbb{E}[u^\top W^\top W u] = \operatorname{tr}(W^\top W\, \mathbb{E}[uu^\top]) = \operatorname{tr}(W^\top W)/n = \|W\|_F^2/n$, since $\mathbb{E}[uu^\top] = I_n/n$. □

M Empirical checks for typical-case diagnostic ingredients

Table 14 reports the relaxed diagnostic values used in Section 5.2. Here we additionally report lightweight checks corresponding to the modeling ingredients behind the typical-case diagnostic on SHD-FC. These checks support the heuristic interpretation but do not make it a worst-case guarantee.

Columns of Table 14: $\|W\|_{\text{op}}$ and $\|W\|_{\text{eff}} = \|W\|_F/\sqrt{d_{\text{in}}}$, where d_{in} is the input dimension of the corresponding weight matrix; $\kappa \triangleq (\|W\|_{\text{op}}/\|W\|_{\text{eff}})^2$ is the effective condition number; η is the empirical surrogate sparsity; $\bar{\alpha}$ is the 99th percentile over timesteps of the neuron-averaged $|\alpha_{\text{eff}}(t)|$ (a summary statistic, not the uniform bound required by (A3)) and $\tilde{\alpha}$ its time average. Using the per-neuron supremum instead only increases λ_r (e.g., to 249.9 for MARS lif1), so the conclusion $\lambda_r \gg 1$ is unchanged.

(B1) average-direction amplification. Table 14 shows that the squared operator/effective-norm ratio κ is large (26–82), so worst-case singular directions are much more pessimistic than average directions.

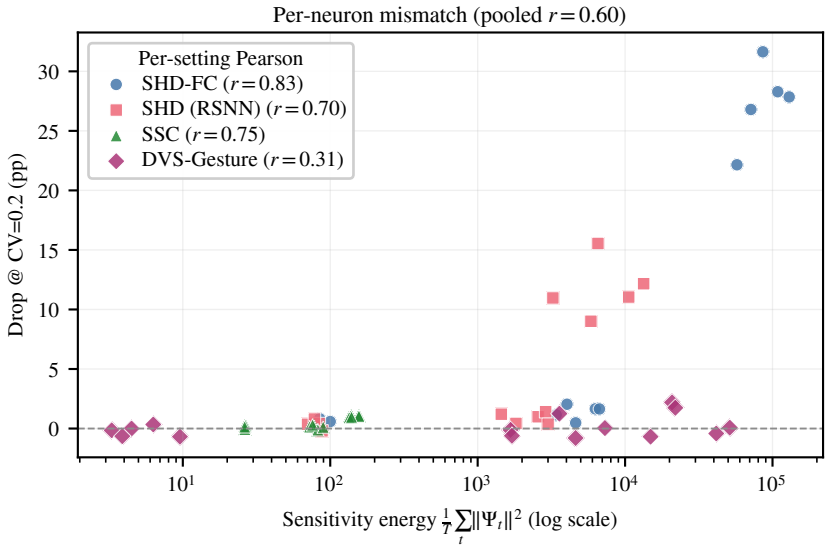


Figure 3: **Post-hoc sensitivity energy and deployment drop.** Each point is a Clean, Noise Aug, or MARS checkpoint evaluated under per-neuron β mismatch at CV $= 0.2$ (200 draws per checkpoint for SHD, 10 otherwise). Within each temporally dependent setting, higher sensitivity energy corresponds to a larger drop (per-setting Pearson $r = 0.70$–0.83); DVS-Gesture checkpoints remain in the low-drop regime ($r = 0.31$).

This motivates the effective-norm diagnostic; it is not by itself a proof that the trained perturbation directions are isotropic. **(B2) temporal accumulation.** We bound the time sum by Cauchy–Schwarz rather than assuming incoherence because perturbation directions are temporally coherent: on SHD-FC (5 seeds, 30 test batches, per-neuron β mismatch at CV $= 0.1$), the mean pairwise cosine between normalized $\delta V_t^{(l)}$ at different timesteps of the same input is 0.50–0.82 (layer 1) and 0.71–0.83 (layer 2) across training modes, and the per-neuron autocorrelation $\mathrm{corr}(\delta V_t, \delta V_{t+k})$ falls below 0.1 only after at least 6 steps. **(B3) weak surrogate-mask coupling.** Table 14 reports $\eta \in [0.008, 0.026]$, confirming sparse surrogate support. We also compute $c_t = \langle|\hat{\sigma}_t|, |\Psi_t|\rangle / (\|\hat{\sigma}_t\|_2 \|\Psi_t\|_2)$ on SHD-FC across 10 test batches. The mean cosine lies in $[0.03, 0.15]$ across modes and layers, while a coordinate-permuted baseline lies in $[0.13, 0.26]$. Thus we do not observe strong positive alignment between surrogate activity and perturbation magnitude, but we still treat (B3) only as a weak-coupling proxy rather than literal independence. **(B4) average multiplicative decay proxy.** Table 14 reports $\tilde{\alpha} \leq \bar{\alpha} < 1$ in all layers. We use $\tilde{\alpha}$ only as the plug-in rate in the exponential product proxy of (B4); it is not derived from Jensen's inequality or membrane-potential normality. **(B5) cross-layer incoherence.** We do not separately validate (B5) in this appendix. Its role is only to explain why the heuristic diagnostic omits the extra $(L-1)$ factor from the worst-case readout bound; if one drops (B5), that factor should be restored. These checks support the typical-case diagnostic as a reasonable modeling approximation rather than a worst-case guarantee.

N External assets and licensing

We rely entirely on publicly released benchmark datasets and open-source software. We do not redistribute any third-party dataset; users reproducing the experiments should obtain the data from the original sources and comply with the corresponding terms of use.

Datasets. SHD and SSC are obtained from the Heidelberg spiking datasets release [9] (Creative Commons Attribution 4.0, CC BY 4.0; `https://zenkelab.org/datasets`). DVS-Gesture [1] is released by IBM under CC BY 4.0 according to the README distributed with the dataset; we load it through tonic [24], using the pre-processed copy by the tonic authors that is also released under CC BY 4.0 (Zenodo record 8060604). CIFAR-10 [23] is distributed without an explicit license; we use it under the terms of its official page (`https://www.cs.toronto.edu/~kriz/cifar.html`), which asks users to cite the accompanying technical report.

Software. Versions are those used with the released code (the checkpoints were trained with an earlier PyTorch release). We use PyTorch 2.11.0 [3] and torchvision 0.26.0 (both BSD-3-Clause), snnTorch 0.9.4 [11] (MIT), tonic 1.6.0 [24] (GPL-3.0), SpikingJelly 0.0.0.0.14 [14] (this release ships under the Open-Intelligence Open Source License v1.0; the current repository uses Apache-2.0), and DCLS 0.1.1 [21] (MIT). The SSC experiments build on the DelRec code [32] (MIT, `https://github.com/alexmaxad/DelRec`), which our release vendors with modifications under its original license and copyright notice; the modified files are listed in the repository.

O Low-precision weights

MARS regularizes sensitivity to the decay factor only; it is not a full-parameter robustness method. Table 15(a) applies symmetric per-tensor post-training quantization (PTQ) to the weights of the main checkpoints and then per-neuron β mismatch at $\mathrm{CV} = 0.2$ (200 draws per checkpoint, shared with Table 4); drops are measured from the quantized nominal model. At 8 and 6 bits, the β-mismatch drops of both robust-training methods remain at most 1.5 pp, but quantization lowers nominal accuracy, most visibly for MARS on SHD-RSNN at 6 bits ($84.8 \rightarrow 83.2\%$). Table 15(b) shows that at 4 bits, without β mismatch, MARS on SHD-RSNN collapses to 16.7%, whereas Clean and Noise Aug retain 73–77%; on SHD-FC all methods lose 14–18 pp. In the seed-42 SHD-FC checkpoints, the operator norm of the first weight matrix is 26% larger for MARS than for Clean (41.9 vs. 33.2; Table 14). This co-occurs with the larger low-bit sensitivity, but we have not established a causal link: the measurement covers one seed and no norm-controlled intervention was run. Weight quantization should therefore be handled by dedicated methods alongside MARS.

Table 15: Weight quantization of the main checkpoints. **(a)** PTQ followed by per-neuron β mismatch at $\mathrm{CV} = 0.2$; nominal is the quantized model without β mismatch (32 = full precision). **(b)** Nominal accuracy after 4-/3-bit PTQ without β mismatch. Mean±SD over 5 seeds.

(a) PTQ + per-neuron β mismatch

Setting	Bits	Method	Nominal (%)	Drop @ CV= 0.2 (pp)
SHD-FC	32	Clean	76.9±2.7	27.35±3.42
		Noise Aug	76.2±1.1	1.39±0.60
		MARS	77.0±1.7	0.37±0.34
	8	Clean	77.1±2.7	27.45±4.01
		Noise Aug	76.1±1.0	1.45±0.59
		MARS	76.7±1.9	0.05±0.36
	6	Clean	76.4±3.0	27.28±3.74
		Noise Aug	75.3±1.1	1.49±1.07
		MARS	76.4±1.9	0.29±0.29
SHD-RSNN	32	Clean	85.1±1.3	11.75±2.41
		Noise Aug	85.5±1.3	0.88±0.46
		MARS	84.8±1.6	0.32±0.36
	8	Clean	85.3±1.4	12.09±2.10
		Noise Aug	85.6±1.4	1.00±0.46
		MARS	84.5±1.6	0.10±0.21
	6	Clean	84.8±1.3	11.84±2.27
		Noise Aug	84.8±1.3	0.76±0.48
		MARS	83.2±1.4	0.21±0.19

(b) Low-bit PTQ of the main checkpoints, no β mismatch

Setting	Method	FP32 (%)	PTQ 4-bit	PTQ 3-bit	Drop @ 4-bit
SHD-FC	Clean	76.9±2.7	63.3±4.0	18.8±5.9	13.60±4.58
	Noise Aug	76.2±1.1	57.8±5.2	12.6±8.1	18.44±5.42
	MARS	77.0±1.7	61.0±3.9	12.6±4.9	16.02±2.97
SHD-RSNN	Clean	85.1±1.3	73.4±3.8	41.2±11.9	11.72±3.62
	Noise Aug	85.5±1.3	77.2±5.1	28.7±8.8	8.30±4.13
	MARS	84.8±1.6	16.7±2.3	6.9±1.6	68.14±3.27

P Computational overhead

Hardware and total compute. All experiments were run on a single NVIDIA RTX 5090 GPU (32 GB). Estimated from the run timestamps of the sequentially executed training runs (including end-of-training mismatch evaluation), the 75 main training runs (5 settings $\times$ 3 training modes $\times$ 5 seeds) took about 204 GPU-hours ($\approx$ 8.5 GPU-days): about 14–37 min per run for SHD-FC, SHD-RSNN, DVS-Gesture, and CIFAR-10, and about 12 h per run for SSC, which accounts for 180 of the 204 hours. These are wall-clock times; shorter jobs occasionally shared the GPU during the SSC batches, so the SSC figure is an upper estimate. The λ sweep (Appendix G) took a further 7 h; the remaining controls, post-hoc evaluations, and preliminary or discarded runs were not timed per run, and we estimate their total compute to be of the same order as the main runs.

Protocol. Costs are measured on one NVIDIA RTX 5090 for one complete model-side optimization step (forward pass, MARS regularizer or noise augmentation, loss, backward pass, gradient clipping, Adam update) on a real batch from each setting's training loader, with 3 warm-up steps followed by 10 timed steps. Data loading is excluded. Memory is the *total* peak allocated GPU memory during the step (not the increment over the pre-step allocation). There is one measurement run per setting (no seed replication); the $\pm$ values in Table 16 are the SD across the 10 timed steps. Separately, the end-to-end SHD-FC training epoch, including the standard multi-worker loader and input augmentation, grows from 58.3 s (Clean) to 80.1 s (MARS), i.e., 1.37$\times$. This is a different quantity from the model-side step ratios (1.35–2.06$\times$). MARS adds no inference cost: the regularizer is used only during training, and the deployed network is architecturally identical to a Clean one.

Complexity. For a feedforward LIF layer with n_l neurons, the online sensitivity recursion carries one state $\Psi_t^{(l)}$ per neuron and sample, i.e., $\Theta(B\sum_l n_l)$ additional state and $\Theta(TB\sum_l n_l)$ additional element-wise forward work for batch size B and T steps. Recurrent layers add a matrix–vector term per step proportional to the number of recurrent connections, and the delay-aware SSC implementation additionally stores and updates a delay buffer of past sensitivities, so the feedforward expression is not exact there. The online state count is not the training-memory complexity: our implementation keeps the recursion in the autograd graph (`detach_state=False`), so backpropagation through time stores time-step-dependent intermediates of the recursion, and the extra training memory grows with T (Table 16).

Table 16: Absolute model-side step time (ms, mean$\pm$SD over 10 timed steps) and total peak allocated GPU memory (GiB) on one RTX 5090, with ratios to Clean. Data loading excluded; single measurement run per setting. DVS-Gesture uses a real batch from its training loader.

		Step time (ms)			Peak memory (GiB)			MARS / Clean	
Setting	Batch/T	Clean	Noise Aug	MARS	Clean	Noise Aug	MARS	Time	Mem.
SHD-FC	64/50	28.95$\pm$0.21	28.79$\pm$0.10	59.74$\pm$0.46	0.049	0.064	0.085	2.06$\times$	1.74$\times$
SHD-RSNN	64/50	35.44$\pm$0.42	35.47$\pm$0.18	72.92$\pm$0.39	0.068	0.075	0.107	2.06$\times$	1.58$\times$
SSC (DelRec)	128/250	383.87$\pm$2.71	390.82$\pm$1.56	519.31$\pm$2.35	5.247	5.542	5.905	1.35$\times$	1.13$\times$
DVS-Gesture	8/16	82.74$\pm$1.22	80.77$\pm$0.44	142.56$\pm$0.81	5.561	6.162	12.765	1.72$\times$	2.30$\times$
CIFAR-10	64/4	42.53$\pm$1.52	42.16$\pm$4.49	75.07$\pm$3.91	3.741	4.711	7.272	1.77$\times$	1.94$\times$

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Introduction state the paper's simulation-scoped claims: learnable membrane time constants create task-dependent deployment fragility, membrane sensitivity Ψ explains first-order fragility and admits an online recursion, MARS reduces degradation under per-neuron, layer-wise, and global β mismatch on the temporally dependent settings, and the same sensitivity energy accounts for the robustness of noise augmentation. All experiments are described as hardware-grounded controlled mismatch evaluations in simulation, and Section 6.1 states the absence of on-chip validation.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 6.1 discusses the simulation-only scope (no on-chip or hardware-in-the-loop validation), the sufficient and surrogate-linearized nature of the worst-case theorem, the modeling assumptions behind the typical-case diagnostic, the empirical status of the recurrent and delay-learning settings, the training-time and memory overhead of MARS (Appendix P), the fact that MARS does not address weight quantization (Appendix O), the compression of generic temporal memory (Appendix J), and the untested Transformer-based SNN backbones and large-scale event datasets.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Theorem 1 states assumptions (A1)–(A3), and Appendix A gives the full proof, including its extension to the learnable leaky readout used in the snnTorch settings (Appendix A); Appendix B gives the sensitivity derivations and recurrent contraction proposition. The typical-case expression in Section 4.2 is explicitly presented as a modeling-assumption-based diagnostic, with assumptions and derivation sketch in Appendix L, not as a formal theorem.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 5.1 specifies datasets, architectures, training modes, the per-neuron, layer-wise, and global β-mismatch protocols, CV levels, and the number of mismatch draws per protocol. Appendix D gives architecture, training, baseline-control, and evaluation details, Appendix G reports the MARS regularization sweep, and Appendix E and Appendix F report the full and per-seed numerical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Code, trained-model evaluation scripts, and a reproducibility manifest mapping every reported number to its script and result file are released at `https://github.com/eason891023/MARS-NeurIPS2026` (release tag `v1.0-neurips2026`). The experiments use public benchmark datasets, which are not redistributed (Appendix N).

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5.1 reports the dataset–architecture settings, training modes, mismatch protocols, CV levels, and number of mismatch draws. Appendix D reports optimizer, epoch count, surrogate choice, learning rates, initialization, the MARS regularization strength, and the hyperparameters of all baseline controls, while Appendix G reports the regularization-strength sweep.

Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: All settings use 5 independently trained seeds, which are the unit of inference; mismatch draws are first averaged within each checkpoint. Figure 1 shows 95% percentile bootstrap confidence intervals across seeds, Tables 1, 2, and 3 report mean $\pm$ SD across seeds, and Appendix F reports per-seed values and paired differences with bootstrap intervals and exact sign-flip p-values. Section 5.1 states that with five seeds the smallest attainable two-sided p-value is 0.0625, so we report paired effect sizes rather than claims of statistical significance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: All experiments were run in PyTorch on a single NVIDIA RTX 5090 GPU with 32 GB memory. Appendix P reports, for every setting, the training-step time and peak GPU memory of Clean, Noise Aug, and MARS, the per-run and total training time of the main runs (about 8.5 GPU-days, estimated from run timestamps), and an estimate of the additional compute used by other reported and preliminary experiments; MARS adds no inference cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work consists of theoretical analysis and benchmark simulation experiments on public datasets, with no human-subject data collection or high-risk model release. The authors are not aware of ethical issues beyond the general dual-use considerations of more reliable event-based inference, which are discussed in Section 6.1.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Section 6.1 discusses the positive impact of more reliable low-power event-based inference and the possible negative impact of improving sensing systems used in monitoring contexts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release high-risk generative models, scraped datasets, or similarly sensitive assets requiring special release safeguards.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: Appendix N cites every dataset and software package, gives the versions used, and states each license or terms of use: SHD/SSC and DVS-Gesture (CC BY 4.0), CIFAR-10 (no explicit license; terms of its official page), PyTorch and torchvision (BSD-3-Clause), snnTorch (MIT), tonic (GPL-3.0), SpikingJelly (Open-Intelligence Open Source License v1.0 for the release used), DCLS (MIT), and the DelRec code (MIT), which our release vendors with modifications under its original license. No third-party dataset is redistributed.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: The only new asset is the code release at `https://github.com/eason891023/MARS-NeurIPS2026` (tag `v1.0-neurips2026`), documented by its README and MANIFEST (training and evaluation commands, result-file provenance) and released under the MIT license; no new dataset or model checkpoints are released.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subjects research.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve human-subjects research requiring IRB review.

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not part of the proposed method, theoretical analysis, training pipeline, or experimental evaluation.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.