Do Not Let Spikes Flip: Margin-Resculpted Learning for Robust Spiking Neural Networks

Desong Zhang, Jia Hu*, Geyong Min*
Department of Computer Science
University of Exeter
{dz288, J.Hu, G.Min}@exeter.ac.uk

Abstract

Spiking Neural Networks (SNNs) transmit information through energy-efficient discrete spike events, unlike the continuous activations used in conventional Artificial Neural Networks (ANNs). Existing SNN robustness methods often inherit ANN-oriented objectives that suppress continuous perturbations or stabilize membrane-potential dynamics. In this paper, we argue that SNN robustness should be analyzed through the stability of spike patterns under perturbations. We formalize perturbation-induced spike flips and introduce the flip rate as a training-agnostic metric for quantifying spike-pattern instability. We further show that spike flips are closely associated with membrane potentials near the firing threshold, and introduce margin occupancy to quantify this threshold-margin vulnerability. Based on these findings, we propose **M**argin-**R**esculpted **L**earning (MRL), which regularizes excessive threshold-margin occupancy with an activity-adaptive bound. MRL reduces vulnerable spike flips while preserving the threshold region needed for surrogate-gradient learning. Experiments across static and event-based datasets, architectures, surrogate functions, and attack settings demonstrate consistent robustness improvements. The code is available at `https://github.com/Apple26419/SNN_MRL`

1 Introduction

As a brain-inspired computing paradigm, Spiking Neural Networks (SNNs) transmit information through binary spike events [38]. By replacing dense continuous activations with sparse event-driven signals, SNNs can reduce floating-point computation and operate efficiently on neuromorphic hardware [46, 42, 59]. These properties make SNNs attractive for resource-constrained and latency-sensitive applications, including edge computing [36, 58, 12] and autonomous driving [60, 47, 41].

However, practical deployment in safety-sensitive scenarios requires not only efficiency but also reliability under adversarial perturbations [15, 3]. Existing robustness studies on SNNs often borrow ideas from conventional Artificial Neural Networks (ANNs), such as adversarial training, Lipschitz regularization, or input purification [37, 16, 7, 35, 57, 43]. Although these methods improve SNN robustness in various settings, many of them are still guided by objectives developed for continuous-valued networks, where robustness is closely related to the stability of internal numerical representations.

This perspective is natural for ANNs, since small input perturbations can be continuously transformed and accumulated through successive layers [4, 39, 50]. In SNNs, however, downstream computation is mediated by spike events. A fluctuation in membrane potential affects later layers only when it crosses the threshold and changes whether a neuron emits a spike [18]. Thus, part of the numerical perturbation can be absorbed by temporal integration and leakage without altering the resulting spike pattern. This does not imply that SNNs are inherently robust, but it suggests that the perturbations most relevant to prediction stability are those that change spike events. We therefore argue that:

*Corresponding author

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

Robustness in SNNs should be analyzed through the stability of spike patterns under perturbations.

This perspective follows from the fact that an SNN prediction is ultimately determined by the resulting spatio-temporal spike pattern [29, 56], although this pattern is generated from continuous membrane dynamics [21, 13]. For analyzing prediction stability, the central question is therefore not whether membrane potentials change, but whether such changes alter the spike events that mediate downstream computation. This distinction exposes a limitation of purely numerical stability objectives: they may suppress membrane-potential variations that do not affect spike events, while leaving unclear which internal states are most responsible for perturbation-induced prediction changes.

Based on this view, we recharacterize adversarial robustness in SNNs in terms of perturbation-induced spike flips. We first formalize spike flips and introduce the flip rate as a quantitative metric for measuring spike pattern instability. We then relate flip rates to membrane-potential distributions around the firing threshold, where small perturbations are more likely to change firing states. To characterize this threshold-adjacent vulnerability, we use margin occupancy as a distribution-level measure of how densely membrane potentials populate the sensitive region near the threshold. This analysis motivates Margin-Resculpted Learning (MRL), a threshold-margin-oriented training strategy that reshapes membrane-potential distributions to reduce vulnerable spike flips while preserving normal spiking dynamics. Specifically, our contributions are threefold:

- **We define spike flip and flip rate.** We formalize perturbation-induced spike flips from the perspective of spike pattern stability and introduce the flip rate as a training-agnostic metric for quantifying spike pattern instability, providing the SNN community with a general tool for diagnosing, comparing, and improving robustness.
- **We reveal the threshold-margin mechanism behind spike flips.** We show that spike flips concentrate around the firing threshold, where small membrane-potential perturbations can change spike emissions, and introduce margin occupancy to quantify this vulnerable region.
- **We propose Margin-Resculpted Learning (MRL).** Guided by this mechanism, MRL reshapes threshold-margin membrane-potential distributions to reduce vulnerable spike flips while preserving normal spiking dynamics. Experiments across datasets, architectures, and perturbation settings show that MRL lowers flip rates and improves adversarial robustness.

2 Preliminary

Dynamics of leaky integrate-and-fire neuron. In the field of SNNs, the most widely used neuron model is the Leaky Integrate-and-Fire (LIF), which simulates the charging and discharging process of biological neurons to achieve activation [19]. From an implementation perspective, the dynamics of a single LIF neuron are given in Eq. (1), where $u[t]$, $s[t]$, and $c[t]$ denote the membrane potential, the binary spike output, and the input current at time step t, respectively. Here, $\tau > 1$ is the membrane time constant, u_{th} is the firing threshold, and $\mathcal{H}$ is the Heaviside function.

$$\begin{aligned} u[t+1] &= \Big(1 - \frac{1}{\tau}\Big)\big(u[t] - u_{\mathrm{th}} s[t]\big) + c[t], \\ s[t+1] &= \mathcal{H}\big(u[t+1] - u_{\mathrm{th}}\big). \end{aligned} \tag{1}$$

$$\arg\max_{\|\delta\|_p \le \varepsilon} \mathcal{L}\big(f(x+\delta), y\big). \tag{2}$$

Adversarial attacks. Given an input x with label y, adversarial examples are generated by finding a perturbation δ within an ℓ_p-norm ball of radius ε that maximizes the loss $\mathcal{L}\big(f(x+\delta), y\big)$, as formulated in Eq. (2). Details of adversarial attacks are provided in Appendix A.

3 Empirical Findings, Analysis, and Method

This section develops the mechanism that motivates MRL. We first define spike flips and use the flip rate to quantify perturbation-induced spike-pattern instability (Sec. 3.1). We then examine where such flips are more likely to occur and connect them to threshold-adjacent membrane states (Sec. 3.2). Finally, we translate this analysis into MRL, which regularizes vulnerable membrane-potential distributions to reduce spike flips (Sec. 3.3).

3.1 Spike Flips and Performance Degradation

Inference in SNNs is mediated by spatio-temporal spike patterns, from which the final prediction is produced [38]. Therefore, when a perturbation changes the prediction, it must have altered the internal spike pattern at some neuron and time step. This finding motivates us to examine robustness at the level of spike events, rather than only through continuous membrane-potential variations.

We introduce spike flip to describe a local change in spike emission between clean and adversarial inference. We further define the flip rate as a global statistic that measures how much the spike pattern is altered by a perturbation.

Definition 1 (Spike Flip)**.** *Consider an SNN unfolded over T time steps. Following the LIF dynamics in Eq.* (1)*, a clean input and its adversarial counterpart induce two paired membrane-potential trajectories. For neuron i in layer ℓ at time step t, denote the clean and adversarial membrane potentials by $u_{\ell,i}[t]$ and $u^{\mathrm{adv}}_{\ell,i}[t]$, and their spike events by $s_{\ell,i}[t]$ and $s^{\mathrm{adv}}_{\ell,i}[t]$. A spike flip occurs when the clean and adversarial inputs produce different spike events at the same neuron and time step:*

$$f_{\ell,i}[t] = \mathbb{I}\big[s_{\ell,i}[t] \neq s^{\mathrm{adv}}_{\ell,i}[t]\big]\,. \tag{3}$$

Definition 2 (Flip Rate)**.** *Flip rate measures the average fraction of paired clean and adversarial spike events that differ. For a trained SNN, the adversarial flip rate γ^{adv} is defined by averaging spike flips over samples, time steps, neurons, and layers, as Eq. (4), where $\gamma^{\mathrm{adv}}_{\ell}$ denotes the layer-wise flip rate at layer ℓ, L is the number of layers, $\mathcal{I}_\ell$ is the set of neurons in layer ℓ, and N is the number of evaluated samples. The detailed implementation of flip rate statistics is provided in Appendix B.*

$$\gamma^{\mathrm{adv}} = \frac{1}{L}\sum_{\ell=1}^{L}\gamma^{\mathrm{adv}}_{\ell} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{\sum_{m=1}^{N}\sum_{t=1}^{T}\sum_{i\in\mathcal{I}_\ell} f^{(m)}_{\ell,i}[t]}{NT|\mathcal{I}_\ell|}. \tag{4}$$

Under the above definitions, we conduct controlled experiments to examine how the flip rate changes with robustness accuracy under different attack strengths, architectures, surrogate gradient functions, and datasets. The detailed experimental configurations and results are provided in Appendix D. Fig. 1 shows three representative cases, from which we find:

Finding 1. *Across different architectures, surrogate functions, datasets, and attack methods, the flip rate γ^{adv} increases monotonically with the perturbation strength ε and exhibits a consistent negative correlation with robustness.*

Finding 1 indicates that the flip rate tracks the degree of spike-pattern instability associated with adversarial degradation. We next give a simple decision-level argument to explain why such a relation is expected.

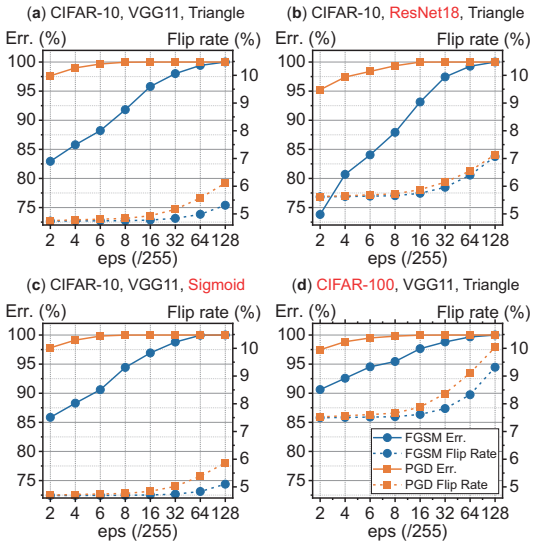

Figure 1: Relationship between model robustness and flip rate. To facilitate comparison, Err. = 1 - Acc. is reported as the classification failure, two y-axes across all subplots are unified. In each subtitle, the elements that differ from (**a**) are highlighted. Full version of results is provided in Appendix D.

Analysis of Finding 1. Following the definitions in Sec. 3.1, the network prediction is determined by the spiking pattern $\{s_{\ell,i}[t]\}$. Therefore, if the predicted label differs between clean and adversarial inference, at least one spike event that is relevant to the final decision must have changed. Let $\mathcal{E}$ denote the event that the predicted label changes between clean and adversarial inference. Under standard adversarial settings, the accuracy degradation $\Delta\mathrm{Acc} = \mathrm{Acc}_{\mathrm{clean}} - \mathrm{Acc}_{\mathrm{adv}}$ is upper bounded by $\mathbb{P}(\mathcal{E})$, since adversarial perturbations are optimized to increase the loss and rarely improve correct predictions [22].

Since the final decision is determined by a finite set of spike events, we can consider a task- and model-dependent set of decision-relevant spike locations $\mathcal{S}$, with $|\mathcal{S}| = K$. A prediction change can occur

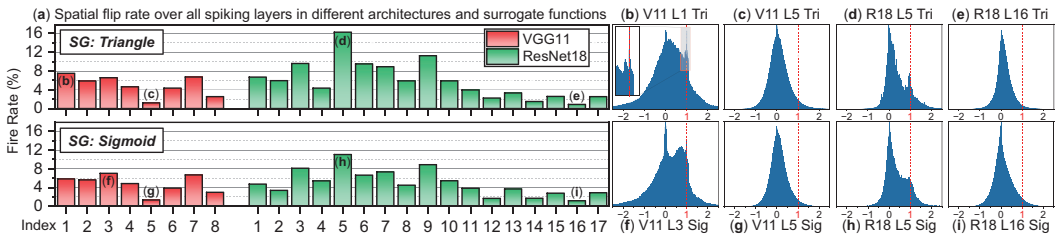

Figure 2: Relationship between flip rate and membrane-potential distributions. 8 representative layers, namely the layers with the highest and lowest flip rates, are selected to be shown, as shown in subplots (**b**)-(**i**),where u_{th} is indicated by a red dashed line.

only if at least one location in this set undergoes a spike flip, which gives $\mathcal{E} \subseteq \bigcup_{(\ell,i,t)\in\mathcal{S}} \{f_{\ell,i}[t] = 1\}$. Applying the union bound yields an upper bound on the probability of prediction change:

$$\Delta\text{Acc} \leq \mathbb{P}(\mathcal{E}) \leq \sum_{(\ell,i,t)\in\mathcal{S}} \mathbb{P}(f_{\ell,i}[t] = 1) = K\,\bar{\gamma}_{\mathcal{S}}, \tag{5}$$

where $\bar{\gamma}_{\mathcal{S}}$ denotes the average flip rate restricted to the decision-critical set $\mathcal{S}$. This restricted flip rate is not directly observable without knowing which spike events are decision-relevant. We therefore use the global flip rate γ^{adv} as a measurable proxy for overall spike-pattern instability. The empirical consistency in Finding 1 suggests that, across the examined settings, the restricted and global flip rates vary in a comparable manner. We express this relation as $\bar{\gamma}_{\mathcal{S}} \leq \alpha\,\gamma^{\text{adv}}$, where α is a task- and model-dependent constant. Combining the above results yields $\Delta\text{Acc} \lesssim \alpha K\,\gamma^{\text{adv}}$, which shows that accuracy degradation can be related to the flip rate up to a task-dependent factor.[2]

3.2 Which Membrane States Are More Flip-vulnerable?

The previous subsection relates the global flip rate to adversarial degradation. However, the flip rate alone does not explain where spike-pattern instability arises. Fig. 1 further suggests that spike flips remain sparse even when robustness drops substantially. This motivates a more fine-grained question: *which internal membrane states are more likely to induce spike flips under perturbations?*

Answering this question requires moving from the decision level to the neuron-state level. A spike flip occurs when the perturbation changes the firing decision at a specific neuron and time step. Since this decision is made by comparing the membrane potential with the firing threshold, flip vulnerability is naturally tied to the distance between the membrane potential and the threshold. For a given induced change in membrane potential, states far from the threshold require a larger change to alter the spike event, whereas threshold-adjacent states can change their firing outcome under smaller perturbations. Therefore, spike flips are expected to concentrate around membrane states located near the firing threshold, rather than being uniformly distributed across neurons and time steps [26, 24].

To examine this threshold-margin effect, we visualize layer-wise membrane-potential distributions together with their corresponding flip rates. Fig. 2 reports representative layers with high and low flip rates, from which we find:

Finding 2. *Flip rates vary substantially across layers, while their layer-wise trends remain consistent across surrogate functions. Layers with higher flip rates exhibit stronger membrane-potential concentration around the firing threshold, whereas layers with lower flip rates show more dispersed distributions away from the threshold. This pattern is consistently observed across different architectures and surrogate functions.*

Finding 2 suggests that spike flips are associated with threshold-adjacent membrane states. To quantify this, we need a distribution-level metric for how much membrane-potential mass lies near the threshold. A simple average margin may miss small but vulnerable concentrations near the boundary, while a hard threshold window cannot reflect the gradual change of vulnerability with distance. We therefore define margin occupancy as a soft measure induced by the surrogate-gradient profile.

Definition 3 (Margin Occupancy)**.** *The margin occupancy (MO) $\mathcal{M}$ of layer ℓ is defined as*

$$\mathcal{M}_\ell = \mathbb{E}[\,\kappa(m_\ell)\,] = \mathbb{E}\left[\,\kappa\left(\frac{|u_\ell - u_{\text{th}}|}{\text{std}(u_\ell - u_{\text{th}})}\right)\right], \tag{6}$$

[2]We emphasize that this bound only characterizes the scaling relationship between accuracy degradation and flip rate.

where the expectation is taken over the sampled inputs, time steps, and neurons in layer ℓ. Here, $m_\ell = [m_{\ell,1}, m_{\ell,2}, \cdots, m_{\ell,|\mathcal{I}_\ell|}]$ denotes the set of normalized margins of membrane potentials in layer ℓ, which measures their scale-invariant distances to the firing threshold. $\kappa(\cdot)$ is the margin vulnerability profile, which assigns a relative flip-vulnerability score to each normalized margin. To make this profile consistent with the training dynamics of SNNs, we instantiate $\kappa(\cdot)$ using the surrogate gradient. Let $\tilde{s}(u)$ denote the primitive surrogate function, and let $g(u) = \frac{\partial \tilde{s}(u)}{\partial u}$ denote its gradient profile (details are provided in Appendix C). $\kappa(\cdot)$ is then defined as

$$\kappa(m_\ell) = \frac{\left| g(\mathrm{std}(u_\ell - u_{\mathrm{th}}) \cdot \mathrm{sign}(u_\ell - u_{\mathrm{th}}) \cdot m_\ell) \right|}{\left| g(0) \right|}. \tag{7}$$

Remark 1. *The surrogate gradient profile provides a smooth, training-consistent proxy for the threshold-sensitive region around the firing boundary [44]. Thus, the margin vulnerability profile is not manually specified by an arbitrary window size, but induced by the surrogate function used during optimization. Further motivation and design details for $\kappa(\cdot)$ are provided in Appendix E.*

MO measures the degree to which a layer places membrane potentials in the threshold-sensitive region. We next examine whether this quantity is aligned with the observed flip rate. Specifically, across multiple architectures and surrogate settings, we compute layer-wise MO on clean inputs and compare it with the corresponding flip rate under FGSM perturbations ($\varepsilon = 8/255$) on CIFAR-10. Their dependence is quantified using Spearman's rank correlation coefficient ρ with significance level p, and the results are summarized in Fig. 3, from which we find:

Finding 3. *Layer-wise MO and flip rate exhibit a strong positive monotonic correlation. According to the statistical interpretation in [8], ρ consistently falls into the strong correlation regime ($\rho > 0.7$), with several settings reaching the extremely strong correlation regime ($\rho > 0.9$). Moreover, all correlations are statistically significant, with $p < 0.05$ in all cases and many satisfying $p < 10^{-5}$.*

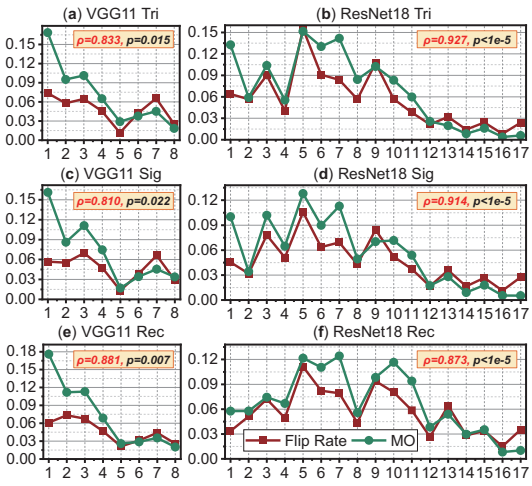

Figure 3: Relationship between flip rates and MOs. Horizontal axes represent the network layer index, while vertical axes jointly denote flip rates and MOs, as they are on the same scale.

Finding 3 answers the question posed in this subsection: **membrane states concentrated near the firing threshold are more flip-vulnerable under perturbations**. Together with Finding 2, this result motivates reducing excessive threshold-margin occupancy in the training objective.

3.3 Regularizing MO by Margin-Resculpted Learning

Findings 1, 2, and 3 suggest a training principle for improving SNN robustness: reduce excessive membrane-potential occupancy around the firing threshold, where spike flips are more likely to occur. At the same time, this threshold region should not be eliminated entirely, since it also provides the functional region through which surrogate gradients support learning [44, 55]. Thus, the goal is not to suppress all threshold-adjacent states, but to control their excessive concentration.

Motivated by this principle, we propose Margin-Resculpted Learning (MRL), which reshapes membrane-potential distributions by regularizing excessive layer-wise MO. Specifically, we incorporate the MO regularizer into the training objective as

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{MRL}} \mathcal{L}_{\mathrm{MRL}}. \tag{8}$$

The two components introduced beyond the standard cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$ are described below.

MRL regularizer $\mathcal{L}_{\mathrm{MRL}}$. For each layer, MRL penalizes only the MO that exceeds an activity-adaptive bound:

$$\mathcal{L}_{\mathrm{MRL}} = \frac{1}{L} \sum_{\ell=1}^{L} \max(0, \mathcal{M}_\ell - \mathcal{B}_\ell). \tag{9}$$

This design is important because membrane potentials near the threshold are not purely harmful. They also form the effective working region of surrogate-gradient learning [33, 27, 56]. Directly driving MO toward zero may therefore weaken gradient propagation and harm normal spiking dynamics. We instead allow each layer to retain a limited amount of threshold-margin occupancy and penalize only the excess beyond this admissible level. Specifically, we set

$$\mathcal{B}_\ell = \big(r_\ell(1-r_\ell)\big)^2, \tag{10}$$

where r_ℓ is the average firing rate of layer ℓ. We design $\mathcal{B}_\ell$ as an activity-adaptive budget for admissible threshold-margin occupancy. Since spike emission is binary, a randomly selected spike event from layer ℓ can be viewed as a Bernoulli variable with mean r_ℓ, whose uncertainty is measured by $r_\ell(1-r_\ell)$ [28]. This uncertainty vanishes when the layer is nearly silent or nearly saturated, and is maximized when firing decisions are balanced, providing a natural signal for when threshold-adjacent membrane states are functionally justified for surrogate-gradient learning. Rather than using this first-order uncertainty directly, we adopt the squared form $\big(r_\ell(1-r_\ell)\big)^2$ as a conservative budget: it remains nonzero when spike decisions are genuinely uncertain, but becomes substantially tighter in sparse or near-deterministic regimes. In particular, when $r_\ell \ll 1$, which is common in SNNs, the bound scales as $\mathcal{B}_\ell = r_\ell^2 + \mathcal{O}(r_\ell^3)$, compared with the linear scaling $r_\ell(1-r_\ell) = r_\ell + \mathcal{O}(r_\ell^2)$. Thus, $\mathcal{B}_\ell$ lets each layer retain a limited threshold-margin region for learning while penalizing excessive accumulation that may increase spike-flip vulnerability.

Remark 2. *The squared bound is not introduced as a unique theoretical optimum, but as a conservative, hyperparameter-free activity-adaptive budget with desirable boundary behavior: $\mathcal{B}_\ell = 0$ when $r_\ell \in \{0, 1\}$ and it is maximized at $r_\ell = 1/2$. Appendix F provides the full motivation, examples, and mathematical properties for this activity-adaptive budget design.*

Loss balancing via λ_{MRL}. Although $\mathcal{L}_{\mathrm{MRL}}$ penalizes excessive MO, its raw scale can vary across architectures, layers, and training stages because membrane-potential statistics evolve during training. A fixed coefficient may therefore make the regularizer either too weak to affect optimization or too strong when the MO penalty becomes active. To keep the auxiliary constraint effective without requiring dataset- or architecture-specific tuning, we balance the two objectives at the optimization scale by matching their gradient magnitudes as Eq. (11), where θ denotes trainable parameters, and σ is a small constant for numerical stabilization. Eq. (11) adaptively rescales $\mathcal{L}_{\mathrm{MRL}}$ to a comparable optimization scale with $\mathcal{L}_{\mathrm{CE}}$, keeping MRL active throughout training. The stop-gradient operation prevents gradients from propagating through the balancing coefficient itself.

$$\lambda_{\mathrm{MRL}} \;=\; \frac{\mathrm{stopgrad}(\|\nabla_\theta \mathcal{L}_{\mathrm{CE}}\|)}{\mathrm{stopgrad}(\|\nabla_\theta \mathcal{L}_{\mathrm{MRL}}\|) + \sigma}. \tag{11}$$

4 Experiment

4.1 Robustness Evaluation and Comparison

Experiments are conducted on CIFAR-10, CIFAR-100, ImageNet, DVS-CIFAR10, and DVS-Gesture. We evaluate robustness under FGSM, PGD, and APGD attacks. Dataset descriptions, attack configurations, and hyperparameter settings are provided in Appendices G, A, and H, respectively.

White-box performance comparison. We evaluate MRL under white-box attacks across different network architectures (VGG11 and ResNet18) and surrogate gradient settings (Rectangle, Sigmoid, and Triangle), and compare it against SOTA defenses. The results are summarized in Table 1. Under the same VGG11 architecture, our method consistently outperforms existing approaches. For example, on CIFAR-100 under PGD attack, MRL achieves a peak accuracy of 14.25%, substantially higher than the 2.04% achieved by FEEL. Moreover, MRL generalizes well across architectures, delivering significant robustness gains on ResNet18. Importantly, our method remains compatible with different surrogate gradients, as MO is fundamentally derived from the gradient profiles of surrogate functions.

Large-scale white-box performance with deep networks. Since MRL is inherently designed in a layer-wise manner, we further evaluate its compatibility with deeper networks and its scalability to the large-scale ImageNet dataset.[3] As shown in Table 2, MRL integrates seamlessly with networks of varying depths and consistently yields substantial robustness improvements on large-scale data.

[3]For experimental cases in which the surrogate is not explicitly specified, the Triangle surrogate is applied by default.

Table 1: White-box performance comparison with SOTA methods (%). To ensure fairness, only the best in each **VGG11** column is highlighted.

Arch.	Method	MRL	CIFAR-10					CIFAR-100				
			Clean	FGSM	PGD	$APGD_c$	$APGD_d$	Clean	FGSM	PGD	$APGD_c$	$APGD_d$
VGG11	DSD [57] *ICLR'26*	-	90.21	55.86	30.21	-	-	70.26	23.81	8.09	-	-
	TGO [52] *ICLR'26*	-	88.79	51.40	11.67	-	-	69.47	17.20	2.30	-	-
	FEEL [53] *NeurIPS'24*	-	93.29	44.96	28.35	-	-	**73.79**	9.60	2.04	-	-
	StoG [16] *ICML'24*	-	91.64	16.22	0.28	-	-	70.44	8.27	0.49	-	-
	DLIF [15] *AAAI'24*	-	92.39	15.24	0.23	0.17	0.08	70.51	8.72	0.77	0.48	0.94
	HoSNN [20] *TMLR'24*	-	92.43	54.76	15.32	10.35	27.39	71.98	13.48	0.19	2.55	0.02
	Rectangle	✗	87.71	7.55	0.01	0.54	0.92	68.62	3.76	0.07	0.03	0.08
		✓	82.04	46.55	28.87	15.11	25.48	62.91	22.44	**14.25**	3.47	1.86
	Sigmoid	✗	90.29	7.61	0.01	0.67	1.35	70.16	3.73	0.16	0.05	0.09
		✓	86.67	49.28	29.65	15.32	26.10	67.56	21.17	12.53	3.21	1.27
	Triangle	✗	**93.75**	8.19	0.03	0.61	2.06	72.39	4.55	0.19	0.09	0.12
		✓	92.57	**56.02**	**32.89**	**17.24**	**27.96**	70.37	**23.94**	12.78	**3.99**	**2.08**
ResNet18	Rectangle	✗	89.74	8.16	0.18	0.79	1.07	70.57	3.08	0.07	0.01	0.00
		✓	86.13	52.05	33.89	16.40	27.21	65.86	20.71	11.74	4.26	1.99
	Sigmoid	✗	91.51	10.19	0.36	1.12	1.98	73.50	4.71	0.27	0.07	0.07
		✓	89.21	60.60	36.80	19.26	30.03	68.42	22.04	12.63	4.12	2.68
	Triangle	✗	94.38	9.37	0.21	0.97	2.89	74.91	5.22	0.52	0.14	0.20
		✓	91.57	56.18	34.92	18.75	29.05	70.67	25.51	14.89	5.34	2.88

Black-box performance. We further evaluate MRL under black-box attacks across different perturbation strengths, and report the results together with the corresponding white-box attacks in Fig. 4. The black-box attacks are consistently less effective than their white-box counterparts, which is expected when gradients remain informative to the attacker. This comparison also supports the gradient obfuscation inspection in Sec. 4.3.

Table 2: White-box performance on ImageNet (%).

Arch.	MRL	ImageNet				
		Clean	FGSM	PGD	$APGD_c$	$APGD_d$
DSD [57]	-	53.47	14.69	4.30	-	-
NF-ResNet18	✗	57.84	4.99	0.01	0.02	0.08
	✓	56.01	17.26	6.52	3.71	3.62
NF-ResNet34	✗	63.86	6.21	0.01	0.03	0.13
	✓	60.98	20.97	8.17	4.57	4.69
NF-ResNet50	✗	65.50	7.02	0.02	0.05	0.21
	✓	62.32	22.72	9.33	7.57	7.38

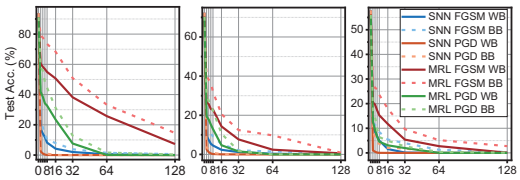

Figure 4: Black- and white-box comparison.

White-box performance on DVS datasets. For the DVS datasets, we follow the settings in [37, 57] and apply FGSM and PGD attacks directly to the preprocessed event frames during evaluation. As shown in Table 3, MRL also improves adversarial robustness on event-based datasets, consistent with the results on static image datasets, while still maintaining largely preserved clean inference performance.

Table 3: Performance on DVS datasets (%). The best in each column is highlighted.

Methods	DVS-CIFAR10			DVS-Gesture		
	Clean	FGSM	PGD	Clean	FGSM	PGD
SNN	**76.30**	17.20	5.00	**95.49**	39.24	9.72
SR [37]	75.50	64.60	61.20	-	-	-
TGO [52]	72.20	57.40	46.90	-	-	-
DSD [57]	75.10	65.80	61.20	93.75	**90.28**	55.21
MRL	74.60	**69.10**	**63.40**	94.44	**90.28**	**55.55**

Effect of MRL on flip rate and MO. We next examine flip rate and MO changes after applying MRL. For clarity, we report the network-level flip rate and aggregated MO $\mathcal{M} = \frac{1}{L}\sum_{\ell=1}^{L}\mathcal{M}_\ell$. As shown in Fig. 5, MRL consistently reduces MO across datasets and architectures, accompanied by lower flip rates. This supports our rationale that regularizing excessive threshold-margin occupancy suppresses perturbation-induced spike flips. In addition, we present two representative case studies in Fig. 6, including layer-wise changes in flip rates and MO, together with membrane-potential distributions of selected layers. Overall, MRL reduces MO and flip rates across layers. For layers with pronounced threshold-margin concentration, such as layer 3 in subplot (b), MRL reshapes the membrane-potential distribution into a more dispersed profile. In contrast, for layers whose membrane potentials are already well distributed, MRL largely preserves the original distribution, suggesting limited disruption to normal signal propagation.

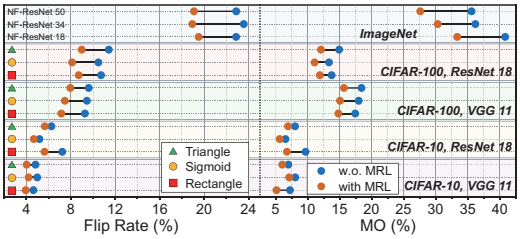

Figure 5: Flip rate and MO changes with MRL.

4.2 Ablation Study

Input/Output only MRL. Since MRL is a layer-wise regularizer, we first examine two simplified variants: applying MRL only to the input layer (Input only MRL), which targets early representations before feature propagation, and applying MRL only to the output layer (Output only MRL), which directly affects the final prediction layer. Both variants are plausible, but they regularize only a restricted part of the spike-processing pathway. We therefore evaluate them with VGG11.

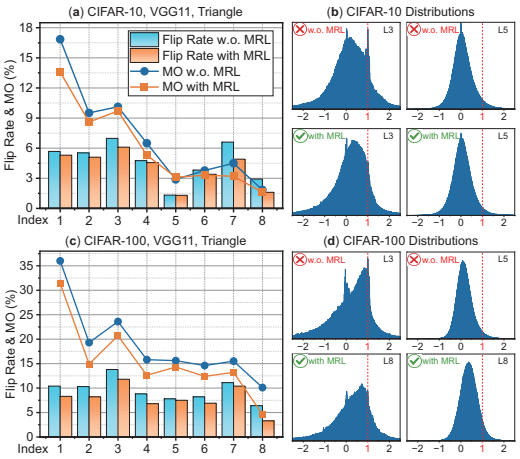

Figure 6: Cases for flip rate and MO changes.

Table 4 reports the results. Input only MRL provides some robustness improvement, but remains far weaker than applying MRL to the whole network. This suggests that even if early spike flips are reduced, high-MO regions can reappear in intermediate layers through subsequent nonlinear propagation, creating new sources of spike-pattern instability. Thus, suppressing threshold-margin occupancy only at the input layer is insufficient for sustained control of the overall flip rate. On the other hand, Table 4 shows that output-only MRL gives even smaller robustness gains. As shown in Figs. 2 and 3, the output layer typically has a lower flip rate than intermediate layers and does not exhibit strong threshold-margin concentration. MRL therefore acts on a relatively stable layer, which limits its effect on global flip statistics and leads to only marginal robustness improvement.

MRL without bound. To verify the necessity of the adaptive bound in Eq. (9), we consider No-bound MRL (NB-MRL), a variant that removes $\mathcal{B}_\ell$ and directly penalizes MO without an admissible margin budget. The results in Table 4 show that NB-MRL severely degrades optimization. Under the Rectangle surrogate, training nearly collapses on both CIFAR-10 and CIFAR-100, with accuracy close to random guessing. This is consistent with our design rationale: unconditional MO suppression expels membrane potentials from the threshold-adjacent region, which is also the functional region for surrogate-gradient propagation. Removing this region causes gradient vanishing and prevents effective learning. For the Sigmoid and Triangle surrogates, NB-MRL avoids complete collapse but still yields substantially worse performance than regular MRL. These surrogates provide smoother or broader gradient support, so optimization can still proceed to some extent. However, directly minimizing MO remains too restrictive and weakens effective gradient flow near the threshold. In contrast, Eq. (9) penalizes only excessive MO beyond $\mathcal{B}_\ell$, preserving a limited threshold-margin region for learning while reducing vulnerable spike flips, thereby better balancing stable optimization and adversarial robustness.

Table 4: MRL variant comparison (%).

Data.	Variant	Clean	FGSM	PGD	APGD_c	APGD_d
CIFAR-10	MRL	92.57	56.02	32.89	17.24	27.96
	Input only MRL	92.72	22.10	7.84	2.17	4.09
	Output only MRL	93.20	12.07	0.12	0.60	1.97
	NB-MRL Rec.	10.28	0.00	0.00	0.00	0.00
	NB-MRL Sig.	67.18	7.41	0.00	0.60	0.00
	NB-MRL Tri.	72.91	7.74	0.00	0.60	0.00
CIFAR-100	MRL	70.37	23.94	12.78	3.99	2.08
	Input only MRL	70.00	8.35	2.50	1.55	1.78
	Output only MRL	71.46	5.10	0.41	0.15	0.18
	NB-MRL Rec.	0.98	0.00	0.00	0.00	0.00
	NB-MRL Sig.	45.12	2.47	0.00	0.00	0.00
	NB-MRL Tri.	59.84	2.99	0.00	0.00	0.00

Exponent of the activity-adaptive budget. To examine the squared form in Eq. (10), we generalize the budget to $\mathcal{B}_\ell(p) = [r_\ell(1-r_\ell)]^p$ and evaluate different exponents on CIFAR-10 with VGG11, the Triangle surrogate, and $T = 4$. For a given firing rate, since $0 \le r_\ell(1-r_\ell) \le 1/4$, increasing p tightens the admissible MO budget. As $p \to +\infty$, the budget approaches zero, recovering the NB-MRL provided in Table 4.

Table 5: Ablation on the budget exponent p (%).

p	Clean	FGSM	PGD	APGD_c	APGD_d
1	**92.76**	16.01	3.95	2.61	3.57
1.5	91.75	39.85	19.76	8.97	20.12
2 (MRL)	92.57	56.02	32.89	**17.24**	**27.96**
2.25	88.05	**58.85**	**33.76**	16.01	25.91
3	80.15	29.48	13.23	5.70	7.24
5	74.75	12.76	1.10	1.27	0.78
$+\infty$ (NB-MRL)	72.91	7.74	0.00	0.60	0.00

Table 5 shows that both overly permissive and overly restrictive budgets compromise performance. At $p = 1$, the model retains the highest clean accuracy (92.76%) but achieves only 3.95% accuracy under PGD. Increasing p from 1.5 to 2 improves PGD accuracy from 19.76% to 32.89% while also improving clean accuracy. Further increasing p to 2.25 raises FGSM and PGD accuracy by 2.83%

and 0.87%, respectively, but reduces clean accuracy by 4.52% and lowers accuracy under both APGD attacks. At $p = 3$ or 5, both clean and adversarial accuracy decline substantially, consistent with the interpretation that excessive suppression of threshold-adjacent states weakens the effective learning region of surrogate gradients. Thus, although the squared form is not optimal for every individual attack, it combines high clean accuracy with the best accuracy under both APGD attacks in this sweep, providing empirical support for the choice of $p = 2$.

4.3 Gradient Obfuscation Inspection

Following the diagnostic criteria proposed by [2], which have become standard checks for evaluating adversarial defenses, we examine whether MRL suffers from gradient obfuscation and summarize the results in Table 6. For items (1) and (2), Fig. 4 reports MRL's accuracy under FGSM and PGD across perturbation bounds, together with white-box and black-box comparisons. MRL is consistently more vulnerable to iterative PGD than single-step FGSM, and white-box attacks cause larger degradation than black-box attacks. Items (3) and (4) are also supported by Fig. 4: accuracy drops as the perturbation bound increases, reaching 0% in several settings. Fig. 7 further shows that more PGD iterations progressively degrade performance before convergence. For item (5), both FGSM and PGD still find adversarial examples, indicating that gradient-based attacks remain effective. Overall, MRL does not rely on gradient obfuscation for robustness.

Table 6: Checklist for identifying gradient obfuscation.

Characteristics to identify gradient obfuscation	Pass?
(1) Single-step attack performs better compared to iterative attacks	✓
(2) Black-box attacks perform better compared to white-box attacks	✓
(3) Increasing perturbation bound can't increase attack strength	✓
(4) Unbounded attacks can't reach 100% success	✓
(5) Adversarial example can be found through random sampling	✓

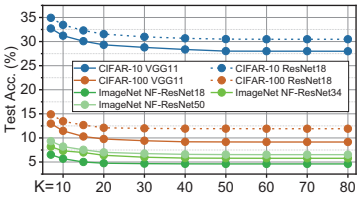

Figure 7: Comparison in multi-K.

5 Related Work

SNN defensive method. Recent studies [40, 48, 34, 6, 54] have observed that SNNs can exhibit a certain degree of inherent robustness due to their event-driven computation. However, such robustness is typically limited and highly dependent on specific network configurations, and does not obviate the need for explicit defense mechanisms. From a defense-mechanism perspective, existing robustness methods for SNNs can be broadly divided into two categories. The first follows ANN-inspired continuous robustness principles, for example, gradient- [37], Lipschitz- [14], input-preprocessing- [31], or gating-based [16] defenses from conventional neural networks to the spiking domain. The second category instead exploits SNN-specific properties, such as spike-based representations [17], threshold adaptation [20], encoding design [53], to enhance robustness in a manner aligned with event-driven computation.

Membrane potential regulation. Regulating membrane potential dynamics and distributions is a common and effective strategy in SNNs, widely used to improve training stability and performance beyond robustness-specific settings [24, 23, 49]. Existing methods generally follow two principles: (i) constraining membrane potentials within an effective operating interval to avoid saturation or degeneration [15, 24, 55, 44], and (ii) preventing undesirable distribution drift across layers or time to maintain stable neural dynamics [24, 23, 25, 51]. Our method MRL strictly follows these principles while addressing a distinct robustness-oriented objective as Fig. 6 shows. Within robustness-related studies, DLIF [15] and TGO [52] are most related to our work. DLIF improves robustness by stabilizing continuous membrane-potential perturbation dynamics, while TGO identifies threshold-neighboring neurons as a robustness bottleneck and pushes membrane potentials away from the threshold, together with noisy spiking neurons, to reduce state-changing probability. These works show that membrane-potential geometry is central to robust SNNs. MRL further refines this view from the perspective of spike-pattern stability: instead of suppressing global membrane perturbations or moving all threshold-neighboring states away, MRL measures perturbation-induced spike flips and regularizes only excessive threshold-margin occupancy. With an activity-adaptive bound, MRL preserves the threshold region needed for surrogate-gradient learning while reducing membrane states most likely to cause vulnerable spike flips.

6 Discussion

Conclusion. This paper studies adversarial robustness in SNNs from a spike-centric perspective. Instead of treating robustness only as continuous membrane-potential stability, we analyze how perturbations alter the spatio-temporal spike patterns that mediate SNN inference. We formalize this effect through spike flips and the flip rate, and further show that flip-prone behavior is closely associated with membrane potentials concentrated near the firing threshold. Based on this analysis, we introduce margin occupancy as a distribution-level indicator of threshold-margin vulnerability and propose Margin-Resculpted Learning (MRL) to regularize excessive occupancy in this region. Experiments across static and event-based datasets, multiple architectures, surrogate functions, and attack settings show that MRL consistently improves adversarial robustness and remains compatible with large-scale SNN training. These results suggest that spike-pattern stability provides a useful basis for understanding and improving robust SNNs.

Limitation. Our method resculpts membrane potential distributions away from the functional operating region of surrogate gradients, which inevitably leads to a reduction in clean inference accuracy. This clean-adversarial trade-off is widely observed in existing studies [57, 52, 20, 53, 16] and remains an open challenge that requires further investigation. In addition, our method introduces a modest increase in training cost, with detailed evaluations provided in Appendix I.

Acknowledgment

This work was supported in part by UKRI Grant No. EP/X0838866/1 and Horizon EU Grant No. 101086159.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[3] Bahareh Kaviani Baghbaderani and Hossein Shahinzadeh. Adversarial attacks and defense mechanisms in spiking neural networks: A comprehensive review. In *2025 29th International Computer Conference, Computer Society of Iran (CSICC)*, pages 01–07. IEEE, 2025.

[4] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

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

[6] Julian Büchel, Gregor Lenz, Yalun Hu, Sadique Sheik, and Martino Sorbaro. Adversarial attacks on spiking convolutional neural networks for event-based vision. *Frontiers in Neuroscience*, 16:1068193, 2022.

[7] Weiran Chen, Qi Sun, and Qi Xu. Defending spiking neural networks against adversarial attacks through image purification. *arXiv preprint*, 2024.

[8] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. routledge, 2013.

[9] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

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

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

[12] Xin Du Di Yu, Linshan Jiang, Wentao Tong, and Shuiguang Deng. Ec-snn: Splitting deep spiking neural networks for edge devices. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 5389–5397, 2024.

[13] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[14] Jianhao Ding, Tong Bu, Zhaofei Yu, Tiejun Huang, and Jian Liu. Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training. *Advances in Neural Information Processing Systems*, 35:24780–24793, 2022.

[15] Jianhao Ding, Zhiyu Pan, Yujia Liu, Zhaofei Yu, and Tiejun Huang. Robust stable spiking neural networks. *arXiv preprint arXiv:2405.20694*, 2024.

[16] Jianhao Ding, Zhaofei Yu, Tiejun Huang, and Jian K Liu. Enhancing the robustness of spiking neural networks with stochastic gating mechanisms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 492–502, 2024.

[17] Jianhao Ding, Zhaofei Yu, Jian K Liu, and Tiejun Huang. Neuromorphic computing paradigms enhance robustness through spiking neural networks. *Nature Communications*, 16(1):10175, 2025.

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

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

[20] Hejia Geng and Peng Li. Hosnn: Adversarially-robust homeostatic spiking neural networks with adaptive firing thresholds. *arXiv preprint arXiv:2308.10373*, 2023.

[21] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[22] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[23] Yufei Guo, Xiaode Liu, Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17391–17401, 2023.

[24] Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 326–335, 2022.

[25] Yufei Guo, Yuhan Zhang, Yuanpei Chen, Weihang Peng, Xiaode Liu, Liwen Zhang, Xuhui Huang, and Zhe Ma. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19420–19430, 2023.

[26] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420–428, 2006.

[27] Luca Herranz-Celotti and Jean Rouat. Stabilizing spiking neuron training. *arXiv preprint arXiv:2202.00282*, 2022.

[28] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[29] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, Anna Hambitzer, and Priyadarshini Panda. Exploring temporal information dynamics in spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 8308–8316, 2023.

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

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

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

[33] Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In *IJCAI*, pages 3002–3010, 2023.

[34] Ling Liang, Xing Hu, Lei Deng, Yujie Wu, Guoqi Li, Yufei Ding, Peng Li, and Yuan Xie. Exploring adversarial attack in spiking neural networks with spike-compatible gradient. *IEEE transactions on neural networks and learning systems*, 34(5):2569–2583, 2021.

[35] Ling Liang, Kaidi Xu, Xing Hu, Lei Deng, and Yuan Xie. Toward robust spiking neural network against adversarial perturbation. *Advances in Neural Information Processing Systems*, 35:10244–10256, 2022.

[36] Yanzhen Liu, Zhijin Qin, and Geoffrey Ye Li. Energy-efficient distributed spiking neural network for wireless edge intelligence. *IEEE Transactions on Wireless Communications*, 23(9):10683–10697, 2024.

[37] Yujia Liu, Tong Bu, Jianhao Ding, Zecheng Hao, Tiejun Huang, and Zhaofei Yu. Enhancing adversarial robustness in snns with sparse gradients. *arXiv preprint arXiv:2405.20355*, 2024.

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

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[40] Alberto Marchisio, Giorgio Nanfa, Faiq Khalid, Muhammad Abdullah Hanif, Maurizio Martina, and Muhammad Shafique. Is spiking secure? a comparative study on the security vulnerabilities of spiking and deep neural networks. In *2020 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

[41] Fernando S Martínez, Jordi Casas-Roma, Laia Subirats, and Raúl Parada. Spiking neural networks for autonomous driving: A review. *Engineering Applications of Artificial Intelligence*, 138:109415, 2024.

[42] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6166–6176, 2023.

[43] Bhaskar Mukhoty, Hilal AlQuabeh, Giulia De Masi, Huan Xiong, and Bin Gu. Certified adversarial robustness for rate encoded spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[44] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

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

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

[47] Albert Shalumov, Raz Halaly, and Elishai Ezra Tsur. Lidar-driven spiking neural network for collision avoidance in autonomous driving. *Bioinspiration & Biomimetics*, 16(6):066016, 2021.

[48] Saima Sharmin, Priyadarshini Panda, Syed Shakib Sarwar, Chankyu Lee, Wachirawit Ponghiran, and Kaushik Roy. A comprehensive analysis on adversarial robustness of spiking neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[49] Hangchi Shen, Qian Zheng, Huamin Wang, and Gang Pan. Rethinking the membrane dynamics and optimization objectives of spiking neural networks. *Advances in Neural Information Processing Systems*, 37:92697–92720, 2024.

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[51] Xingming Tang, Tao Chen, Qian Cheng, Hangchi Shen, Shukai Duan, and Lidan Wang. Spatio-temporal channel attention and membrane potential modulation for efficient spiking neural network. *Engineering Applications of Artificial Intelligence*, 148:110131, 2025.

[52] Shuai Wang, Malu Zhang, Yulin Jiang, Dehao Zhang, Ammar Belatreche, Yu Liang, Yimeng Shan, Zijian Zhou, Yang Yang, and Haizhou Li. Robust spiking neural networks against adversarial attacks. In *The Fourteenth International Conference on Learning Representations*.

[53] Mengting Xu, De Ma, Huajin Tang, Qian Zheng, and Gang Pan. Feel-snn: Robust spiking neural networks with frequency encoding and evolutionary leak factor. *Advances in Neural Information Processing Systems*, 37:91930–91950, 2024.

[54] Nuo Xu, Kaleel Mahmood, Haowen Fang, Ethan Rathbun, Caiwen Ding, and Wujie Wen. Attacking the spike: On the security of spiking neural networks to adversarial examples. *Neurocomputing*, page 131506, 2025.

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

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

[57] Desong Zhang, Jia Hu, and Geyong Min. Robustify spiking neural networks via dominant singular deflation under heterogeneous training vulnerability. In *The Fourteenth International Conference on Learning Representations*.

[58] Guanlei Zhang, Lei Feng, Fanqin Zhou, Zhixiang Yang, Qiyang Zhang, Alaa Saleh, Praveen Kumar Donta, and Chinmaya Kumar Dehury. Spiking neural networks in intelligent edge computing. *IEEE Consumer Electronics Magazine*, 2024.

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

[60] Rui-Jie Zhu, Ziqing Wang, Leilani Gilpin, and Jason Eshraghian. Autonomous driving with spiking neural networks. *Advances in Neural Information Processing Systems*, 37:136782–136804, 2024.

A Adversarial Attacks and Configurations

We apply commonly used attack methods in the SNN community, including the Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and the stronger Auto-PGD (APGD).

FGSM. FGSM [22] is a simple yet effective technique to generate adversarial examples. In FGSM, given an input x with its true label y, a perturbation is computed in the direction of the gradient of the loss function with respect to x. The perturbation is defined as:

$$\delta = \varepsilon \cdot \operatorname{sign}\left(\nabla_x \mathcal{L}(f(x), y)\right). \tag{12}$$

where ε controls the magnitude of the perturbation and $h(x)$ represents the model's output. The adversarial example is then constructed as $x + \delta$, which is designed to force the model into misclassification.

PGD. PGD [39] is an iterative method for generating adversarial examples and can be regarded as an extension of the FGSM. PGD updates the adversarial example iteratively by performing a gradient ascent step and then projecting the result back onto the feasible set defined by the L_p-norm constraint. Formally, the update rule is given by:

$$x^{t+1} = \Pi_{\mathcal{B}(x,\varepsilon)}\Big(x^t + \alpha \cdot \operatorname{sign}\big(\nabla_x \mathcal{L}(f(x^t), y)\big)\Big). \tag{13}$$

where α denotes the step size, $\mathcal{L}(f(x^t), y)$ is the loss function of the model h with true label y, and $\Pi_{\mathcal{B}(x,\varepsilon)}$ is the projection operator that projects the perturbed example back into the ball $\mathcal{B}(x, \varepsilon) = \{x' : \|x' - x\|_p \leq \varepsilon\}$. By iterating this process, PGD effectively seeks a perturbation that maximizes the loss while ensuring that the adversarial example remains within the specified perturbation budget.

APGD. Auto-PGD (APGD) [9] is an iterative adversarial attack based on PGD, equipped with an adaptive step-size strategy and a momentum-like update. Given a perturbation budget ε, the APGD update is defined as

$$x^{t+1} = \Pi_{\mathcal{B}(x,\varepsilon)}\big(x^t + \alpha_t \cdot \operatorname{sign}\big(\nabla_x L(f(x^t), y)\big)\big), \tag{14}$$

where α_t denotes the step size at iteration t. APGD further maintains an auxiliary momentum variable:

$$z^t = x^t + \beta_t(x^t - x^{t-1}), \qquad x^{t+1} = \Pi_{\mathcal{B}(x,\varepsilon)}\big(z^t + \alpha_t \cdot \operatorname{sign}\big(\nabla_x L(f(z^t), y)\big)\big), \tag{15}$$

where β_t is the momentum coefficient.

Hyperparameters. To ensure fair comparison, hyperparameters are set the same as existing works [20, 15, 53, 16]. Specifically, for the attack methods introduced above, we set $\varepsilon = 8/255$ for all experimental cases. For PGD, step size $\alpha = 0.01$ and step number $K = 7$. For APGD, there are two loss versions: APGDCE (Cross Entropy loss) and APGDDLR (Difference of Logits Ratio loss), we use the L_∞-bounded setting with perturbation budget $\varepsilon = 8/255$, initial step size $\alpha_0 = 2/255$.

B Practical Implementation of Spike Flip Rate Statistics

This appendix describes the practical implementation of the spike flip rate statistics used in the main paper. Eq. (4) defines the flip rate by averaging paired clean and adversarial spike differences over samples, time steps, neurons, and layers. Directly recording all spike events or membrane states for every neuron in modern SNNs can be expensive, especially for deep architectures and large-scale datasets. We therefore use a fixed neuron-sampling strategy that preserves the paired comparison in the formal definition while keeping the statistic scalable.

Given a trained SNN, we compare clean and adversarial inference in a paired manner. For each evaluated input, the network state is reset and the clean input is forwarded for T time steps. We record the binary spike outputs at selected neuron locations. The network state is then reset again, and the corresponding adversarial input is forwarded under the same time horizon. A spike flip is counted when the clean and adversarial spike outputs differ at the same sample, layer, neuron, and time step. The same sampled neuron indices are used for both clean and adversarial inference, ensuring that all comparisons are location-wise paired.

Specifically, the overall procedure is summarized in Algorithm 1. For memory efficiency, we estimate each layer-wise average over a fixed subset of neurons $\Omega_\ell \subseteq \mathcal{I}_\ell$. When $\Omega_\ell = \mathcal{I}_\ell$, the algorithm computes Eq. (4) exactly. Otherwise, it gives a Monte Carlo estimate of the layer-wise flip rate.

Algorithm 1 Flip Rate Estimation

Input: Trained SNN $\mathcal{N}$, evaluation set $\{(x^{(m)}, y^{(m)})\}_{m=1}^{N}$, time steps T, threshold u_{th}, sampled neuron sets $\{\Omega_\ell\}_{\ell=1}^{L}$
Output: Estimated adversarial flip rate $\widehat{\gamma}^{\mathrm{adv}}$ and layer-wise rates $\{\widehat{\gamma}_\ell^{\mathrm{adv}}\}_{\ell=1}^{L}$
Initialize layer-wise flip counters $C_\ell \leftarrow 0$ for all layers $\ell = 1, \ldots, L$.
for each layer ℓ **do**
  If no subsampling is used, set $\Omega_\ell \leftarrow \mathcal{I}_\ell$.
  Fix Ω_ℓ throughout evaluation for paired clean and adversarial comparison.
end for
for $m = 1$ **to** N **do**
  Generate adversarial example $(x^{\mathrm{adv}})^{(m)}$ from $x^{(m)}$ under the chosen attack.
  Reset the network state and run clean inference on $x^{(m)}$ for T time steps.
  Record $s_{\ell,i}^{(m)}[t]$ for all ℓ, $i \in \Omega_\ell$, and t.
  Reset the network state and run adversarial inference on $(x^{\mathrm{adv}})^{(m)}$ for T time steps.
  Record $(s^{\mathrm{adv}})_{\ell,i}^{(m)}[t]$ for all ℓ, $i \in \Omega_\ell$, and t.
  for each layer ℓ **do**
    for each sampled neuron $i \in \Omega_\ell$ **do**
      for $t = 1$ **to** T **do**
        $f_{\ell,i}^{(m)}[t] \leftarrow \mathbb{I}\Big[s_{\ell,i}^{(m)}[t] \neq (s^{\mathrm{adv}})_{\ell,i}^{(m)}[t]\Big]$.
        $C_\ell \leftarrow C_\ell + f_{\ell,i}^{(m)}[t]$.
      end for
    end for
  end for
end for
for each layer ℓ **do**
  $\widehat{\gamma}_\ell^{\mathrm{adv}} \leftarrow C_\ell / (NT|\Omega_\ell|)$.
end for
$\widehat{\gamma}^{\mathrm{adv}} \leftarrow \frac{1}{L}\sum_{\ell=1}^{L} \widehat{\gamma}_\ell^{\mathrm{adv}}$.
return $\widehat{\gamma}^{\mathrm{adv}}$, $\{\widehat{\gamma}_\ell^{\mathrm{adv}}\}_{\ell=1}^{L}$

In our implementation, the sampling budget is fixed to $|\Omega_\ell| = 20000$ for each layer, or all neurons if $|\mathcal{I}_\ell| < 20000$. For each layer ℓ, Ω_ℓ is sampled uniformly once before evaluation and then kept fixed across all samples, time steps, and inference modes. This fixed sampling ensures that the clean and adversarial spike events are compared at exactly the same neuron-time locations.

Why is random sampling necessary? Recording all spike events or membrane states across all layers, time steps, and samples would require storage on the order of

$$\mathcal{O}\left(N \times T \times \sum_{\ell=1}^{L} |\mathcal{I}_\ell|\right), \tag{16}$$

which is expensive for deep SNNs and large-scale datasets. In addition, registering dense hooks for all neurons introduces non-negligible runtime overhead. Uniform neuron sampling provides a practical alternative that enables large-scale spike flip analysis while keeping memory usage and runtime manageable.

Monte Carlo estimation. The layer-wise flip rate in Eq. (4) is an average over neurons, time steps, and input samples. Therefore, uniformly sampling neurons within each layer yields an unbiased estimator of the full layer-wise average. Taking expectation over the sampled neuron sets gives

$$\mathbb{E}_\Omega\big[\widehat{\gamma}^{\mathrm{adv}}\big] = \gamma^{\mathrm{adv}}. \tag{17}$$

In practice, the large sampling budget used in our experiments keeps the estimator stable. We also empirically observe that increasing the sampling size beyond 20000 neurons per layer does not change the qualitative trends of the reported flip-rate statistics.

C Surrogate Gradient Methods

This appendix summarizes the three surrogate gradients used throughout the paper. We present, for each surrogate, (i) the primitive function $\tilde{s}(\cdot)$, (ii) its hyperparameter(s), and (iii) the associated gradient profile.

C.1 Notation and definition of gradient profile

We use the threshold-centered variable

$$v \triangleq u - u_{\mathrm{th}}, \tag{18}$$

so that $v = 0$ corresponds to the firing boundary. Given a primitive function $\tilde{s}(v)$, we define its gradient profile as

$$g(v) \triangleq \frac{\partial \tilde{s}(v)}{\partial v}. \tag{19}$$

Since $v = u - v_{\mathrm{th}}$ is an affine reparameterization, we also have

$$\frac{\partial \tilde{s}(u - v_{\mathrm{th}})}{\partial u} = \frac{\partial \tilde{s}(v)}{\partial v} = g(v). \tag{20}$$

C.2 Rectangle surrogate

Gradient profile. The Rectangle surrogate is specified directly by its gradient profile as a constant window:

$$g_{\mathrm{rec}}(v) = \begin{cases} \frac{1}{2\alpha}, & |v| \le \alpha, \\ 0, & |v| > \alpha, \end{cases} \tag{21}$$

where $\alpha > 0$ is the half-width of the active region.

Basic properties. The profile is nonnegative, symmetric, and compactly supported:

$$g_{\mathrm{rec}}(v) \ge 0, \qquad g_{\mathrm{rec}}(v) = g_{\mathrm{rec}}(-v), \qquad \mathrm{supp}(g_{\mathrm{rec}}) \subseteq [-\alpha, \alpha]. \tag{22}$$

The maximum value is constant within the support:

$$\max_v g_{\mathrm{rec}}(v) = \frac{1}{2\alpha}. \tag{23}$$

C.3 Sigmoid surrogate

Primitive function. The Sigmoid surrogate uses a smooth approximation

$$\tilde{s}_{\mathrm{sig}}(v) = \sigma(\alpha v) = \frac{1}{1 + \exp(-\alpha v)}, \tag{24}$$

where $\alpha > 0$ is a sharpness hyperparameter.

Gradient profile. Differentiating $\tilde{s}_{\mathrm{sig}}(v)$ yields

$$g_{\mathrm{sig}}(v) = \frac{\partial \tilde{s}_{\mathrm{sig}}(v)}{\partial v} = \frac{\partial \sigma(\alpha v)}{\partial v} = \alpha\, \sigma(\alpha v)\big(1 - \sigma(\alpha v)\big). \tag{25}$$

Equivalently, using $\sigma(z)(1 - \sigma(z)) = \exp(-z)/(1 + \exp(-z))^2$,

$$g_{\mathrm{sig}}(v) = \alpha\, \frac{\exp(-\alpha v)}{\big(1 + \exp(-\alpha v)\big)^2}. \tag{26}$$

Basic properties. The profile is nonnegative and symmetric about $v = 0$:

$$g_{\mathrm{sig}}(v) \ge 0, \qquad g_{\mathrm{sig}}(v) = g_{\mathrm{sig}}(-v). \tag{27}$$

Its maximum is attained at $v = 0$:

$$g_{\mathrm{sig}}(0) = \alpha/4. \tag{28}$$

The support is infinite, but the tails decay exponentially as $|v| \to \infty$.

C.4 Triangle surrogate

Primitive function. We adopt the commonly used piecewise-polynomial primitive function

$$\tilde{s}_{\mathrm{tri}}(x) = \begin{cases} 0, & x < -\frac{1}{\alpha}, \\ -\frac{1}{2}\alpha^2|x|x + \alpha x + \frac{1}{2}, & |x| \le \frac{1}{\alpha}, \\ 1, & x > \frac{1}{\alpha}, \end{cases} \tag{29}$$

where $\alpha > 0$ controls the support width around the threshold.

Gradient profile. Differentiating $\tilde{s}_{\mathrm{tri}}(x)$ yields a compactly supported Triangle gradient profile. For $|x| \le 1/\alpha$, we consider two cases: (i) if $x \ge 0$, then $|x|x = x^2$ and $\frac{d\tilde{s}_{\mathrm{tri}}}{dx} = -\alpha^2 x + \alpha = \alpha(1 - \alpha x)$; (ii) if $x < 0$, then $|x|x = -x^2$ and $\frac{d\tilde{s}_{\mathrm{tri}}}{dx} = \alpha^2 x + \alpha = \alpha(1 + \alpha x)$. Combining both cases gives

$$g_{\mathrm{tri}}(x) \triangleq \frac{d\tilde{s}_{\mathrm{tri}}(x)}{dx} = \begin{cases} \alpha\big(1 - \alpha|x|\big), & |x| \le \frac{1}{\alpha}, \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Thus, $g_{\mathrm{tri}}(x)$ peaks at the threshold with $g_{\mathrm{tri}}(0) = \alpha$ and decays linearly to zero at $|x| = 1/\alpha$.

C.5 Unified view and hyperparameter role

Across all three surrogates, the hyperparameter α controls how strongly gradients concentrate around the threshold. Larger α in the Sigmoid surrogate sharpens the profile around $v = 0$ (higher peak and narrower effective region), whereas larger α in the Rectangle and Triangle surrogates enlarges the compact support $[-\alpha, \alpha]$ (wider working region). In all cases, the gradient profile $g(v)$ defines the threshold-adjacent working region where surrogate gradients are active and thus determines which membrane states are most emphasized during learning.

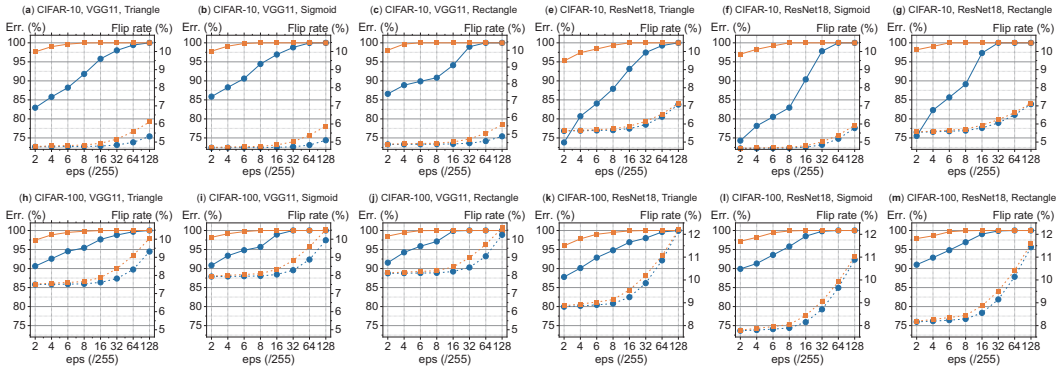

Figure 8: Relationship curves between model robustness and flip rate. This is the full version of Fig. 1. The curve styles are defined as follows: the blue solid line with circular markers denotes FGSM error, the blue dashed line with circular markers denotes FGSM flip rate, the orange solid line with circular markers denotes PGD error, and the orange dashed line with circular markers denotes PGD flip rate. These legend conventions are identical to those used in Fig. 1.

D Configurations and Extensions of Experiments of Robustness v.s. Flip Rate in Sec. 3.1

Here we provide the complete experimental configuration and extended results for the preliminary experiments comparing model robustness and flip rate. In this study, we demonstrate the negative correlation between robustness and flip rate by evaluating different combinations of datasets, network architectures, and surrogate gradient functions.

Hyperparameter configurations. Specifically, we consider CIFAR-10 and CIFAR-100 as datasets (introduced in Appendix G); VGG11 and ResNet18 as network architectures; and the commonly used Triangle, Sigmoid, and Rectangle surrogate functions (described in Appendix C). Except for these

factors, all other hyperparameters are kept identical across experiments: 200 training epochs, batch size 128, membrane time constant 1.1, $T = 4$, and the SGD optimizer with a learning rate of 0.1.

Experimental results. We measure the flip rate under FGSM and PGD attacks (detailed in Appendix A), resulting in a total of 12 experimental settings. The corresponding results are shown in Fig. 8, where $\mathrm{eps}/255$ indicates perturbation strength ε in Eq. (2).

E Motivation and Design of Margin Occupancy

This appendix provides further motivation and justification for the proposed margin occupancy (MO). In the main text, MO is introduced as a compact statistic for measuring how much a layer's membrane-potential distribution occupies the threshold-sensitive region. Here, we explain why such a distribution-level metric is needed and why the surrogate-gradient-induced construction is a principled choice.

In particular, we discuss:

- why spike flips can be viewed as threshold-crossing events (Sec. E.1),
- why simple margin statistics are insufficient (Sec. E.2),
- how margin vulnerability should be quantified at the distribution level (Sec. E.3),
- why surrogate-gradient-consistent profiles provide a natural instantiation (Sec. E.4).

E.1 Spike flips as threshold-crossing events

Given a clean input and its adversarial counterpart, define the induced membrane-potential change at neuron i in layer ℓ and time step t as

$$\Delta u_{\ell,i}[t] = u^{\mathrm{adv}}_{\ell,i}[t] - u_{\ell,i}[t].$$

Since a spike is generated by thresholding the membrane potential, a spike flip can occur only when the clean and adversarial membrane potentials lie on different sides of the firing threshold. Therefore, a necessary condition for $f_{\ell,i}[t] = 1$ is

$$\big|u_{\ell,i}[t] - u_{\mathrm{th}}\big| \le \big|\Delta u_{\ell,i}[t]\big|. \tag{31}$$

If the induced change is bounded by $\widetilde{\Delta u}$ at this neuron and time step, then

$$\big|u_{\ell,i}[t] - u_{\mathrm{th}}\big| \le \widetilde{\Delta u}. \tag{32}$$

This shows that, for a given membrane-potential perturbation, states farther from the threshold require larger changes to flip, whereas threshold-adjacent states are more vulnerable. Thus, spike flip behavior is closely tied to the distribution of membrane potentials relative to the firing threshold.

E.2 Why raw or normalized margins are insufficient

A natural way to characterize flip vulnerability is to consider the margin $m = |u - u_{\mathrm{th}}|$, or its normalized variant. However, pointwise margins or their averages do not capture how vulnerable mass is distributed near the threshold.

To see this, consider two stylized margin distributions P_A and P_B:

$$P_A: \quad m = 0 \text{ with prob. } 0.1, \;\; m = 10 \text{ with prob. } 0.9, \tag{33}$$

$$P_B: \quad m = 1 \text{ with prob. } 1. \tag{34}$$

Their average margins are

$$\mathbb{E}_{P_A}[m] = 9, \qquad \mathbb{E}_{P_B}[m] = 1. \tag{35}$$

Although P_A has a much larger average margin, it places a non-negligible fraction of membrane potentials exactly at the threshold, which can be flipped by arbitrarily small perturbations in the appropriate direction. In contrast, all membrane potentials in P_B are separated from the threshold by a positive margin. This example shows that an average margin can rank a distribution as safer while missing a small but highly vulnerable concentration near the boundary.

Another possible statistic is the fraction of membrane potentials within a fixed window around the threshold:

$$\mathbb{P}(|u - u_{\mathrm{th}}| \le \varepsilon). \tag{36}$$

However, this hard-window statistic assigns the same vulnerability to all states inside the window and zero vulnerability to states just outside it. As a result, it cannot reflect the gradual change of flip vulnerability with distance from the threshold, and small shifts in the window or the distribution may lead to abrupt changes in the measured value.

These limitations motivate a soft distribution-level measure that emphasizes threshold-adjacent membrane potentials while smoothly downweighting states farther from the threshold.

E.3 Distribution-level margin vulnerability

The above discussion suggests that a useful vulnerability measure should satisfy three properties. First, it should assign the highest score to membrane potentials at the firing threshold. Second, it should decrease as the membrane potential moves away from the threshold, rather than using a hard cutoff. Third, it should aggregate vulnerability over the whole membrane-potential distribution of a layer.

Let

$$m_{\ell,i}[t] = \frac{|u_{\ell,i}[t] - u_{\mathrm{th}}|}{\mathrm{std}(u_\ell - u_{\mathrm{th}})}$$

denote the normalized margin for neuron i in layer ℓ at time step t. A margin vulnerability profile $\kappa(\cdot)$ maps this normalized margin to a relative vulnerability score. Aggregating this profile over samples, time steps, and neurons yields the margin occupancy:

$$\mathcal{M}_\ell = \mathbb{E}_{x,t,i}[\kappa(m_{\ell,i}[t])]\,.$$

Thus, MO measures how densely a layer's membrane potentials populate the threshold-sensitive region, while preserving distributional information that would be lost in simple average-margin statistics.

E.4 Surrogate-gradient-consistent margin vulnerability

A remaining question is how to choose the vulnerability profile $\kappa(\cdot)$ without manually specifying a window shape. We use the surrogate gradient employed during SNN training.

Let $v = u - u_{\mathrm{th}}$ denote the threshold-centered membrane potential. In surrogate-based learning, the hard threshold is optimized through a differentiable surrogate primitive $\tilde{s}(v)$, whose gradient profile is

$$g(v) = \frac{\partial \tilde{s}(v)}{\partial v}. \tag{37}$$

This profile specifies the region around the firing threshold where membrane-potential changes are treated as influential during optimization. Since spike flips also occur through threshold crossings, the same profile provides a natural soft weighting for threshold-adjacent vulnerability.

Accordingly, we instantiate the margin vulnerability profile as the normalized surrogate-gradient magnitude:

$$\kappa(m_{\ell,i}[t]) = \frac{|g(\mathrm{std}(u_\ell - u_{\mathrm{th}}) \cdot \mathrm{sign}(u_{\ell,i}[t] - u_{\mathrm{th}}) \cdot m_{\ell,i}[t])|}{|g(0)|}. \tag{38}$$

The sign term maps the nonnegative normalized margin back to the corresponding side of the threshold before applying the surrogate-gradient profile. This construction gives the highest score to states at the firing boundary and smoothly, or according to the surrogate's active region, downweights states farther away.

Different surrogate functions induce different gradient profiles and thus different effective widths of the threshold-sensitive region. This is desirable, because MO is then measured consistently with the surrogate used during training. Empirically, although absolute MO values may vary across surrogate choices, the layer-wise trends remain consistent as shown in Finding 3, supporting that MO captures stable properties of membrane-potential distributions rather than artifacts of a particular surrogate.

F Motivation and Design of the Activity-Adaptive Bound

This section provides the motivation and mathematical properties of the adaptive upper bound used in MRL:

$$\mathcal{B}_\ell = \big(r_\ell(1 - r_\ell)\big)^2, \tag{39}$$

where r_ℓ denotes the average firing rate of layer ℓ. The role of $\mathcal{B}_\ell$ is to specify how much threshold-margin occupancy is admissible before the MO regularizer begins to penalize a layer. **We do not claim that Eq. (39) is the only possible choice.** Rather, we show that it is a conservative, hyperparameter-free, and activity-adaptive budget that satisfies the desiderata required by MRL.

Background and design objective. Margin occupancy $\mathcal{M}_\ell$ measures how much the membrane-potential distribution of layer ℓ occupies the threshold-sensitive region. As discussed in Sec. 3.2, a high MO indicates that many membrane states lie close to the firing threshold, where small perturbations are more likely to change spike emissions. Thus, reducing excessive MO can help suppress vulnerable spike flips.

However, threshold-adjacent membrane states are not purely harmful. They also form the effective working region of surrogate-gradient learning, since surrogate gradients provide nonzero learning signals around the firing threshold [44, 33, 27, 56]. Therefore, directly driving $\mathcal{M}_\ell$ to zero would remove the region where spike decisions are trainable and may lead to gradient vanishing or degraded normal spiking dynamics. The purpose of $\mathcal{B}_\ell$ is therefore to preserve an admissible amount of threshold-margin occupancy while penalizing only its excessive concentration:

$$\max(0, \mathcal{M}_\ell - \mathcal{B}_\ell).$$

A reasonable adaptive bound should satisfy the following properties:

- it should vanish when a layer is nearly deterministic, i.e., almost silent or almost saturated;
- it should remain positive when spike decisions are uncertain, so that surrogate-gradient learning can still operate;
- it should be conservative in sparse firing regimes, which are common in SNNs;
- it should not introduce an additional dataset- or architecture-dependent hyperparameter.

Bernoulli uncertainty as a first-order budget. For layer ℓ, define the average firing rate as

$$r_\ell = \mathbb{E}_{x,t,i}\big[s_{\ell,i}[t]\big],$$

where the expectation is taken over samples, time steps, and neurons in layer ℓ. If we view a randomly selected spike event from layer ℓ as a Bernoulli random variable $s_\ell \in \{0, 1\}$ with mean r_ℓ, its variance is

$$\mathrm{Var}(s_\ell) = r_\ell(1 - r_\ell). \tag{40}$$

This quantity is a natural measure of spike-decision uncertainty. It is zero when $r_\ell = 0$ or $r_\ell = 1$, where spike outcomes are deterministic, and it is maximized at $r_\ell = 1/2$, where firing and non-firing outcomes are most balanced.

This uncertainty term provides a natural first-order admissible budget for threshold-margin occupancy. When a layer is nearly silent or nearly saturated, persistent accumulation near the threshold is not functionally justified: the layer produces almost deterministic spike outcomes, and such threshold-adjacent states mainly increase flip vulnerability. When firing is more uncertain, a nonzero threshold-margin region is more justifiable, because it supports trainable spike decisions under surrogate gradients.

However, using $r_\ell(1 - r_\ell)$ directly as the bound can be overly permissive in sparse SNN regimes. For example, when $r_\ell = 0.05$, the Bernoulli uncertainty is

$$r_\ell(1 - r_\ell) = 0.05 \times 0.95 = 0.0475.$$

This would allow nearly 5% threshold-margin occupancy, comparable to the firing activity itself. In sparse layers, such a large admissible threshold mass may leave many silent-but-near-threshold states vulnerable to perturbation-induced false spikes. Thus, while $r_\ell(1 - r_\ell)$ captures the correct activity-dependent shape, it does not impose a sufficiently conservative constraint in the sparse regime.

Why the squared form. To obtain a more conservative admissible budget, we use the Bernoulli uncertainty not only as a measure of spike-decision uncertainty, but also as an activity gate for how much of this uncertainty should be allocated to the threshold-margin region. This gives

$$\mathcal{B}_\ell = \underbrace{r_\ell(1-r_\ell)}_{\text{spike-decision uncertainty}} \cdot \underbrace{r_\ell(1-r_\ell)}_{\text{activity-dependent margin gate}}. \tag{41}$$

The first factor identifies when uncertain spike decisions make a nonzero margin budget meaningful. The second factor makes this budget conservative when the layer is sparse or near-saturated. Equivalently, Eq. (39) keeps the same uncertainty-dependent shape as the Bernoulli variance but tightens its magnitude.

This squared form has several useful mathematical properties. Let

$$q(r) = r(1-r), \qquad B(r) = q(r)^2.$$

For $r \in [0,1]$, we have $q(r) \in [0, 1/4]$, and therefore

$$0 \le B(r) \le \frac{1}{16}. \tag{42}$$

Moreover,

$$B(r) = 0 \quad \text{if and only if} \quad r \in \{0,1\}, \tag{43}$$

so the admissible MO vanishes exactly in deterministic firing regimes. The maximum occurs at $r = 1/2$:

$$B(1/2) = \left(\frac{1}{4}\right)^2 = \frac{1}{16}. \tag{44}$$

Thus, the bound is largest when spike decisions are most uncertain, but remains conservative even at its maximum.

The squared form is also always no larger than the first-order Bernoulli uncertainty:

$$B(r) = q(r)^2 \le q(r), \tag{45}$$

because $0 \le q(r) \le 1$. Therefore, the squared bound preserves the activity-adaptive structure of $q(r)$ while tightening the admissible budget.

This tightening is particularly important in sparse firing regimes. When $r \ll 1$,

$$B(r) = \big(r(1-r)\big)^2 = r^2 - 2r^3 + r^4 = r^2 + \mathcal{O}(r^3). \tag{46}$$

In contrast, the first-order uncertainty satisfies

$$q(r) = r(1-r) = r + \mathcal{O}(r^2).$$

Hence, in sparse layers, the squared bound scales quadratically rather than linearly with the firing rate. This prevents the admissible threshold-margin occupancy from remaining overly large when only a small fraction of neurons are active. By symmetry, the same behavior holds near saturation:

$$B(r) = (1-r)^2 + \mathcal{O}((1-r)^3) \quad \text{as } r \to 1.$$

Interpretation. The bound $\mathcal{B}_\ell$ can be interpreted as a conservative activity-adaptive budget for threshold-margin occupancy. When a layer is nearly silent, $\mathcal{B}_\ell$ becomes very small, so MRL strongly discourages membrane potentials from accumulating near the threshold and producing perturbation-sensitive false spikes. When a layer is nearly saturated, the bound also becomes small, preventing unstable near-threshold states in an otherwise deterministic firing regime. When firing decisions are more balanced, the bound remains positive, allowing a limited threshold-margin region to support surrogate-gradient learning.

Therefore, Eq. (39) balances two competing requirements: it suppresses excessive threshold-margin occupancy to reduce spike-flip vulnerability, while preserving a nonzero learning window when spike decisions are genuinely uncertain. This is why MRL penalizes $\mathcal{M}_\ell$ only when it exceeds $\mathcal{B}_\ell$, rather than minimizing MO unconditionally.

Table 7: Hyperparameter settings for experiments. *[5]

Dataset	Model	LearningRate	WeightDecay	Epoch	BatchSize	TimeStep
CIFAR-10	VGG11	0.1	5e-5	200	128	4
	ResNet18	0.1	5e-5	200	128	4
CIFAR-100	VGG11	0.1	5e-4	200	128	4
	ResNet18	0.1	5e-4	200	128	4
ImageNet	NF-ResNet18*	0.1	1e-5	100	1024	4
	NF-ResNet34*	0.1	1e-5	100	512	4
	NF-ResNet50*	0.1	1e-5	100	256	4
DVS-CIFAR10	VGG11	0.05	5e-4	200	128	10
DVS-Gesture	VGG11	0.05	5e-4	200	8	20

G Dataset

CIFAR-10. The CIFAR-10 dataset [30] consists of 60,000 color images, each of size 32×32 pixels, divided into 10 different classes, such as airplanes, cars, birds, cats, and dogs. Each class has 6,000 images, with 50,000 images used for training and 10,000 for testing. Normalization, random horizontal flipping, random cropping with 4 padding, and CutOut [11] are applied for data augmentation.

CIFAR-100. The CIFAR-100 dataset [30] consists of 60,000 color images, each of size 32×32 pixels, categorized into 100 different classes. Each class contains 600 images, with 500 used for training and 100 for testing. The same processing methods as for dataset CIFAR-10 are applied to dataset CIFAR-100.

ImageNet. We evaluate on the ILSVRC-2012 ImageNet dataset [10], which contains ∼1.28M training images and 50,000 validation images spanning 1,000 classes. Images are of variable resolution; following common practice and our implementation, training augmentation includes RandomResizedCrop to 224×224, RandomHorizontalFlip, conversion to tensors, and channel-wise normalization. For test, images are resized to have a shorter side of 256 pixels and then center-cropped to 224×224 before applying the same normalization.

DVS-CIFAR10. The DVS-CIFAR-10 dataset ([32]) is a neuromorphic version of the traditional CIFAR-10 dataset. DVS-CIFAR10 captures the visual information using a Dynamic Vision Sensor (DVS), which records changes in the scene as a series of asynchronous events rather than as a sequence of frames. The dataset consists of recordings of 10 object classes, corresponding to the original CIFAR-10 categories, with each object presented in front of a DVS camera under various conditions. The dataset contains 10,000 128×128 images, of which 9,000 are used as the training set and the remaining 1,000 as the test set.

DVS-Gesture. The DVS-Gesture dataset ([1]) is a neuromorphic dataset, consisting of 11 different hand gesture classes, such as hand clapping, arm rolling, and air guitar, performed by 29 subjects under various lighting conditions. Each gesture is represented by a sequence of events rather than frames. The dataset contains 1,176 training samples and 288 testing samples.

H Hyperparameter Settings for MRL

In our experiments, all cases are implemented using PyTorch [45] with the SpikingJelly [18] framework. For each dataset, we utilize the hyperparameters listed in Table 7, consistently employing the SGD optimizer and setting the membrane time constant τ to 1.1. If no declaration is made, the default surrogate gradient we used in the experiment is Triangle. All experiments on CIFAR-10, CIFAR-100, DVS-CIFAR10, and DVS-Gesture are conducted using an NVIDIA GeForce RTX 4070 Ti GPU, while all ImageNet experiments are performed on an NVIDIA H100 SXM GPU. Notably, for DVS and ImageNet experiments, the Triangle surrogate function is used by default unless otherwise specified.

I Train Overhead Evaluation

We evaluate the additional training cost introduced by MRL, with the results reported in Table 8 and 9. Notably, MRL is applied only during training and incurs **no additional overhead at inference time**.

Table 8: Training cost evaluation. Epochtime indicates the average time per epoch (measured with an NVIDIA GeForce RTX 4070 Ti GPU).

Method	Batchsize	Batchnum	MRL	Memory	Epochtime
CIFAR-10, 100	128	390	✗	1.41GB	34"
			✓	1.82GB ▲29%	45" ▲32%
DVS-CIFAR10	128	71	✗	5.18GB	38"
			✓	6.09GB ▲18%	46" ▲21%
DVS-Gesture	8	73	✗	8.47GB	63"
			✓	9.34GB ▲10%	78" ▲24%

Table 9: Training cost evaluation on ImageNet Dataset. Epochtime indicates the average time per epoch (measured with an NVIDIA H100 SXM GPU).

Arch.	Batchsize	Batchnum	MRL	Memory	Epochtime
NF-ResNet18	1024	1251	✗	56.34GB	16'10"
			✓	67.65GB ▲20%	19'51" ▲23%
NF-ResNet34	512	2502	✗	43.87GB	20'01"
			✓	52.17GB ▲19%	28'25" ▲42%
NF-ResNet50	256	5004	✗	60.70GB	47'34"
			✓	73.45GB ▲21%	62'49" ▲32%

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 main contributions, including spike flips, flip rate, margin occupancy, and MRL. The claims are supported by the analysis in Sec. 3, the robustness evaluations in Sec. 4, and the limitations discussed in Sec. 6.

   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: The limitations are discussed in Sec. 6, including the clean-adversarial trade-off caused by resculpting threshold-adjacent membrane states and the additional training cost. Detailed training-cost evaluations are provided in Appendix I.

   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: The paper provides analytical arguments rather than formal theorem statements. The assumptions, definitions, and derivations supporting the flip-rate analysis, margin occupancy design, and activity-adaptive bound are given in Sec. 3 and Appendices E–F.

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: The paper specifies the datasets, model architectures, attacks, surrogate gradients, training settings, and hyperparameters used in the experiments. Additional implementation details are provided in Appendices A, B, G, H, and I.

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: The code is provided in the supplementary materials, and the experiments use public benchmark datasets. Dataset descriptions, preprocessing details, attack configurations, and hyperparameters are documented in the appendices.

   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: Training and evaluation details are described in Sec. 4 and Appendices A, G, and H, including datasets, architectures, attacks, optimizer, learning rates, batch sizes, training epochs, time steps, and surrogate-gradient settings.

   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: [No]

Justification: The paper reports statistical significance for the correlation analysis between margin occupancy and flip rate using Spearman's rank correlation and p-values. However, robustness tables do not report error bars over multiple independent training runs due to computational cost.

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: The compute resources are reported in Appendix H and Appendix I, including GPU types, memory usage, and average epoch time for the evaluated datasets and architectures.

   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 research uses public datasets and standard adversarial robustness evaluation protocols, and does not involve human subjects, private data, or high-risk data collection. The work is intended to improve the reliability of SNNs under adversarial perturbations.

   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: [No]

    Justification: The paper is primarily methodological and defense-oriented, and does not include a separate broader-impact discussion. Potential positive impacts include improving robustness and reliability of energy-efficient SNNs, while misuse risks are limited because the work focuses on defense rather than attack generation.

    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 models, scraped datasets, generative models, or assets with substantial misuse potential. The released code is intended to reproduce the proposed defense and evaluation.

    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: The paper cites properly the original sources of the datasets, software frameworks, and baseline methods used in the experiments.

    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 paper introduces code for reproducing MRL experiments as a new research asset, provided in the supplementary materials with implementation details. No new dataset is introduced.

    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, user studies, or research with human subjects.

    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 crowdsourcing or research with human subjects, so IRB approval or equivalent review is not applicable.

    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 used as any important, original, or non-standard component of the scientific method, experimental design, analysis, or conclusions. They are used only for auxiliary writing support, including language polishing, formatting assistance, and improving the aesthetic presentation of figures.

    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.