Towards Reliable Evaluation of Adversarial Robustness for Spiking Neural Networks

Supplementary Material

Overview. The structure of this appendix is as follows:

Sec. S1 provides the proofs of the main theoretical results presented in this paper.

Sec. S2 describes the detailed dynamics of the neuron models used in our experiments.

Sec. S3 presents the full parameter settings for adversarial training and adversarial attacks.

Sec. S4 includes additional experimental results.

S1. Proofs and Derivations

Theorem 1. *For any surrogate function $g(x)$, the gradient-vanishing degree function $G(x) = \int_{-x}^{x} g(t)\,dt, x \geq 0$ is a concave function on $[0, +\infty)$*

Proof. Since g is even, we have $g(-t) = g(t)$ for all $t \in \mathbb{R}$, and hence

$$G(x) = \int_{-x}^{x} g(t)\,dt = 2\int_{0}^{x} g(t)\,dt, \quad x \geq 0. \tag{S1}$$

We first express the slope of the secant line of G over an interval $[x, y]$ with $0 \leq x < y$ in terms of the average of g on $[x, y]$. Using Eq. (S1), we obtain

$$\begin{aligned}\frac{G(y) - G(x)}{y - x} &= \frac{2\int_0^y g(t)\,dt - 2\int_0^x g(t)\,dt}{y - x} \\ &= \frac{2}{y - x}\int_x^y g(t)\,dt.\end{aligned}$$

For any $0 \leq x < y$ and any $t \in [x, y]$, as g is non-increasing on $[0, +\infty)$, we have:

$$g(y) \leq g(t) \leq g(x).$$

Integrating this inequality over $t \in [x, y]$ and dividing by $y - x > 0$, we obtain

$$g(y) \leq \frac{1}{y - x}\int_x^y g(t)\,dt \leq g(x).$$

This yields

$$2g(y) \leq \frac{G(y) - G(x)}{y - x} \leq 2g(x), \quad 0 \leq x < y. \tag{S2}$$

We now compare slopes on adjacent intervals. Let $0 \leq x < y < z$. Applying Eq. (S2) to the pair (x, y) gives

$$\frac{G(y) - G(x)}{y - x} \;\geq\; 2g(y),$$

while applying Eq. (S2) to the pair (y, z) gives

$$\frac{G(z) - G(y)}{z - y} \;\leq\; 2g(y).$$

Combining these two inequalities, we obtain

$$\frac{G(y) - G(x)}{y - x} \geq \frac{G(z) - G(y)}{z - y}, \quad 0 \leq x < y < z. \tag{S3}$$

The inequality Eq. (S3) states that the secant slopes of G are nonincreasing as the interval moves to the right. This property is equivalent to concavity. For completeness, we sketch the standard argument.

Fix $0 \leq x < z$ and $\lambda \in (0, 1)$, and define

$$y = \lambda z + (1 - \lambda)x,$$

so that $x < y < z$ and

$$y - x = \lambda(z - x), \quad z - y = (1 - \lambda)(z - x).$$

Substituting into Eq. (S3) and using $z - x > 0$ yields

$$\frac{G(y) - G(x)}{\lambda(z - x)} \;\geq\; \frac{G(z) - G(y)}{(1 - \lambda)(z - x)}.$$

Multiplying both sides by $\lambda(1 - \lambda)(z - x) > 0$ gives

$$(1 - \lambda)\big(G(y) - G(x)\big) \;\geq\; \lambda\big(G(z) - G(y)\big).$$

Rearranging terms, we obtain

$$(1 - \lambda)G(y) + \lambda G(y) \;\geq\; \lambda G(z) + (1 - \lambda)G(x),$$

that is,

$$G(y) \;\geq\; \lambda G(z) + (1 - \lambda)G(x) = \lambda G\big(z\big) + (1 - \lambda)G\big(x\big).$$

Recalling that $y = \lambda z + (1 - \lambda)x$, this is precisely the concavity inequality

$$G\big(\lambda z + (1 - \lambda)x\big) \;\geq\; \lambda G(z) + (1 - \lambda)G(x)$$

Hence G is concave on $[0, +\infty)$. □

Corollary 1. *Let $g(x) = \alpha k(\alpha x)$ a surrogate function, and the corresponding gradient-vanishing degree function $\int_{-x}^{x} \alpha k(\alpha t)\,dt = \int_{-\alpha x}^{\alpha x} k(\alpha t)\,d\alpha t = G(\alpha x), x \geq 0$. Assume that x follows a probability distribution $p(x)$ with finite expectation $\mathbb{E}[x] < +\infty$. Then, for any constant $A \in [0, 1]$, the expected gradient-vanishing degree satisfies*

$$\mathbb{E}_{x\sim p(x)}[G(\alpha x)] \leq A \text{ if } \alpha \leq \frac{G^{-1}(A)}{\mathbb{E}[x]} \tag{S4}$$

Proof. Recall that for $x \geq 0$,

$$G(\alpha x) = \int_{-x}^{x} \alpha k(\alpha t)\, dt = \int_{-\alpha x}^{\alpha x} k(u)\, du,$$

hence $G : [0, +\infty) \to [0, 1]$ is nondecreasing and satisfies $G(0) = 0$ (and typically $\lim_{z\to\infty} G(z) = 1$). According to Theorem 1, G is concave on $[0, +\infty)$.

Let $X \sim p(x)$ be a nonnegative random variable with $\mathbb{E}[X] < +\infty$. Since G is concave, Jensen's inequality gives

$$\mathbb{E}\big[G(\alpha X)\big] \leq G\big(\mathbb{E}[\alpha X]\big) = G\big(\alpha\, \mathbb{E}[X]\big).$$

Therefore, for any $A \in [0, 1]$, if

$$G\big(\alpha\, \mathbb{E}[X]\big) \leq A,$$

then automatically $\mathbb{E}[G(\alpha X)] \leq A$.

Because G is nondecreasing on $[0, +\infty)$, the condition $G(\alpha\mathbb{E}[X]) \leq A$ is equivalent to

$$\alpha\, \mathbb{E}[X] \leq G^{-1}(A),$$

where $G^{-1}(A)$ denotes the (generalized) inverse

$$G^{-1}(A) := \inf\{z \geq 0 :\ G(z) \geq A\}.$$

Thus, it suffices that

$$\alpha \leq \frac{G^{-1}(A)}{\mathbb{E}[X]}.$$

Combining with the Jensen bound yields

$$\mathbb{E}_{x\sim p(x)}[G(\alpha x)] \leq A \quad \text{if} \quad \alpha \leq \frac{G^{-1}(A)}{\mathbb{E}[x]},$$

which proves the corollary. □

According to Corollary 1, we present below a series of specific surrogate functions along with their corresponding methods for computing the adaptive sharpness.

(1) Arctangent (Atan) surrogate

$$g_{\mathrm{Atan}}(x) = \frac{\alpha}{2\left[1 + (\frac{\pi}{2}\alpha x)^2\right]} \tag{S5}$$

$$\alpha_{i,t}^{l} = \frac{2}{\pi\, \mathbb{E}[\left|u_{i,t}^{l}\right|]} \tan\Big(\frac{\pi}{2} A\Big) \tag{S6}$$

(2) Gaussian (Gauss) surrogate

$$g_{\mathrm{Gauss}}(x) = \frac{\alpha}{\sqrt{2\pi}} \exp\left(-\frac{(\alpha x)^2}{2}\right) \tag{S7}$$

$$\alpha_{i,t}^{l} = \Phi^{-1}(\frac{1+A}{2}) \frac{1}{\mathbb{E}\left[\left|u_{i,t}^{l}\right|\right]} \tag{S8}$$

where Φ denotes standard Gaussian PDF.

(3) Sigmoid (Sig) surrogate

$$g_{\mathrm{Sig}}(x) = \alpha\, \sigma(\alpha x)\big(1 - \sigma(\alpha x)\big) \tag{S9}$$

where $\sigma(u) = \frac{1}{1+e^{-u}}$.

$$\alpha_{i,t}^{l} = \ln(\frac{2}{1-A} - 1) \frac{1}{\mathbb{E}\left[\left|u_{i,t}^{l}\right|\right]} \tag{S10}$$

(4) Rectangular (Rect) surrogate

$$g_{\mathrm{Rect}}(x) = \frac{\alpha}{2}\, \mathbf{1}\left\{|x| \leq \frac{1}{\alpha}\right\} \tag{S11}$$

$$\alpha_{i,t}^{l} = A \frac{1}{\mathbb{E}\left[\left|u_{i,t}^{l}\right|\right]} \tag{S12}$$

(5) Triangular (Tri) surrogate

$$g_{\mathrm{Tri}}(x) = \alpha\,(1 - \alpha|x|)\ \mathbf{1}\left\{|x| \leq \frac{1}{\alpha}\right\} \tag{S13}$$

$$\alpha_{i,t}^{l} = (1 - \sqrt{1-A}) \frac{1}{\mathbb{E}\left[\left|u_{i,t}^{l}\right|\right]} \tag{S14}$$

It can be observed that the adaptive sharpness parameters of different surrogate gradients share a unified form: $\alpha_{i,t}^{l} = \frac{\omega}{\mathbb{E}[\left|u_{i,t}^{l}\right|]}$. For the Atan, Gauss, and Sig functions, the valid range of ω is $[0, +\infty)$. In contrast, for the Rect and Tri functions, the valid range of ω is $[0, 1]$. This is because for g_{Rect} and g_{Tri}, the degree of gradient vanishing reaches its upper bound of 1 when $|x| > \frac{1}{\alpha}$, which leads to the constraint $\mathbb{E}\left[\left|u_{i,t}^{l}\right|\right] \leq \frac{1}{\alpha}$ when deriving the inverse function. However, this prerequisite cannot be guaranteed for all inputs and neurons during adversarial attacks. Therefore, we unify the range of ω of all surrogate functions to $[0, +\infty)$.

S2. Spiking Neuron Models

LIF-2. We use LIF-2 to denote a variant of the LIF neuron model, which has been adopted in prior works such as [3, 8, 10, 15, 21, 22]. The dynamics of LIF-2 are given as follows:

$$\boldsymbol{V}^{l}(t+1) = \lambda R\Big(\boldsymbol{V}^{l}(t), \boldsymbol{s}^{l}(t)\Big) + \boldsymbol{W}^{l}\boldsymbol{s}^{l-1}(t+1), \tag{S15}$$

$$R\Big(\boldsymbol{V}^{l}(t), \boldsymbol{s}^{l}(t)\Big) = \begin{cases} 0, & \boldsymbol{s}^{l}(t) = 1, \\ \boldsymbol{V}^{l}(t), & \boldsymbol{s}^{l}(t) = 0. \end{cases} \tag{S16}$$

The main difference between LIF-2 and the standard LIF neuron is that LIF-2 does not apply the $1 - \lambda$ normalization factor to the input current.

IF. The IF neuron further simplifies this by setting the decay coefficient to 1:

$$\boldsymbol{V}^{l}(t+1) = R\Big(\boldsymbol{V}^{l}(t), \boldsymbol{s}^{l}(t)\Big) + \boldsymbol{W}^{l}\boldsymbol{s}^{l-1}(t+1), \tag{S17}$$

$$R\Big(\boldsymbol{V}^l(t), \boldsymbol{s}^l(t)\Big) = \begin{cases} 0, & \boldsymbol{s}^l(t) = 1, \\ \boldsymbol{V}^l(t), & \boldsymbol{s}^l(t) = 0. \end{cases} \tag{S18}$$

PSN. The PSN is a neuron model that enables parallel processing along the temporal dimension, and its dynamics differ fundamentally from those of conventional neurons [14]. PSN removes the reset mechanism and treats the firing threshold at each time step as a learnable parameter:

$$\boldsymbol{V}^l = \boldsymbol{W}_T^l \boldsymbol{X}^l, \boldsymbol{W}_T^l \in \mathbb{R}^{T\times T}, \boldsymbol{X}^l \in \mathbb{R}^{T\times N} \tag{S19}$$

$$\boldsymbol{S}^l = H(\boldsymbol{V}^l - \boldsymbol{B}^l), \boldsymbol{B}^l \in \mathbb{R}^T, \boldsymbol{S}^l \in \{0,1\}^{T\times N} \tag{S20}$$

where $\boldsymbol{X}^l$, $\boldsymbol{V}^l$ and $\boldsymbol{S}^l$ denote the input current, membrane potential, and spike sequence, respectively, and $\boldsymbol{W}_T^l$ and $\boldsymbol{B}^l$ represent the learnable temporal weight matrix and threshold parameters. This formulation allows the membrane potential $\boldsymbol{V}^l(t)$ at each time step to integrate information from all temporal positions, rather than relying solely on the previous step as in conventional neuron models.

S3. Detailed Experimental Settings

S3.1. Adversarial Training Settings

In this work, we train robust SNN models using commonly adopted adversarial training schemes for evaluation. Both standard adversarial training (AT) and TRADES generate adversarial examples using PGD with a perturbation budget of $\varepsilon = 8/255$, a step size of $\eta = 4/255$, and 5 attack iterations. RAT and AT+SR follow the same AT setup but additionally incorporate their respective regularization terms, with the regularization coefficient set to 0.004 for both methods.

We adopt Temporal Efficient Training (TET) [12] as the loss function and optimize the models using SGD with a momentum of 0.9. All models are trained for 100 epochs. For SNNs using neuron models other than PSN, we employ a triangular surrogate gradient during training:

$$\frac{dH}{dx} \approx 2 \cdot \max\{0, 1 - |x|\} \tag{S21}$$

PSN-based SNNs are trained using the Atan surrogate gradient with $\alpha = 4$.

S3.2. Adversarial Attack Settings

For RGA and PDSG, we follow the parameter settings provided in their original papers [3, 22]. For ASSG, we set $\beta_1 = 0.9$ and $\beta_2 = 0.9$ when Poisson encoding is not used. When Poisson encoding is applied, we instead use $\beta_1 = 0.99$ and $\beta_2 = 0.99$. The initial values of $M_{i,t}^l$ and $D_{i,t}^l$ are set to $M_0 = 1.0$ and $D_0 = 0.0$, respectively, and the relaxation coefficient $\gamma = 1.5$. We search for the optimal value of A within the range $[0.82, 0.9]$ with a step size of 0.01. Although the ASR values of ASSG reported in Tab. 1-Tab. 3 correspond to the best-performing A, **??** and Fig. S1 demonstrate that the attack performance of ASSG is robust to the choice of A.

For SA-PGD, we set $\beta_m = 0.5$ and $\beta_v = 0.8$. When attacking SNNs with Poisson encoding, we estimate the gradient using the Expectation-over-Transformation (EOT) technique [2], with 10 EOT samples per iteration.

We design Adam-PGD as a baseline for comparison with SA-PGD. The update rule in Adam-PGD follows the Adam optimization mechanism, with an additional projection step to ensure that the updated input remains within the L_∞ constraint:

$$\boldsymbol{m}_k = \beta_m \boldsymbol{m}_{k-1} + (1-\beta_m)\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_k\right), y\right) \tag{S22}$$

$$\boldsymbol{v}_k = \beta_v \boldsymbol{v}_{k-1} + (1-\beta_v)\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_k\right), y\right)^2 \tag{S23}$$

$$\widehat{\boldsymbol{m}}_k = \frac{\boldsymbol{m}_k}{1-\beta_m^k}, \qquad \widehat{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k}{1-\beta_v^k}, \tag{S24}$$

$$\boldsymbol{x}_{k+1} = \Pi_\varepsilon^\infty\left(\boldsymbol{x}_k + \eta_k \frac{\widehat{\boldsymbol{m}}_k}{\sqrt{\widehat{\boldsymbol{v}}_k} + \xi}\right) \tag{S25}$$

where η_k adopt the same learning rate adjustment strategy as in APGD and SA-PGD. We set $\beta_m = 0.8$ and $\beta_v = 0.9$ in Adam-PGD.

S4. Additional Results

S4.1. Loss Curves of more Iteration Numbers

We further plot the loss curves of different gradient approximation methods under SA-PGD at 400 and 1000 attack iterations to more comprehensively examine their convergence behavior. As shown in Fig. S2, ASSG only begins to converge when the iteration reaches 1000 steps, demonstrating its ability to continuously and effectively optimize adversarial perturbations. This requirement for extremely large iteration budgets also highlights the inherent difficulty of generating adversarial examples for SNNs.

S4.2. Additional Comparative Experiments between Fixed-α and ASSG

In this section, we present additional results on models trained with various adversarial training methods, reporting the ASR for attacks using Atan surrogate gradients with different values of fixed-α and ASSG with different values of A in Eq. (S5)-Eq. (S6). As shown in the Fig. S1, the results across more models further demonstrate that ASSG's input-dependent, spatio-temporal adaptive adjustment of the sharpness parameter $\alpha_{i,t}^l$ is superior to directly searching for a globally optimal fixed sharpness parameter α.

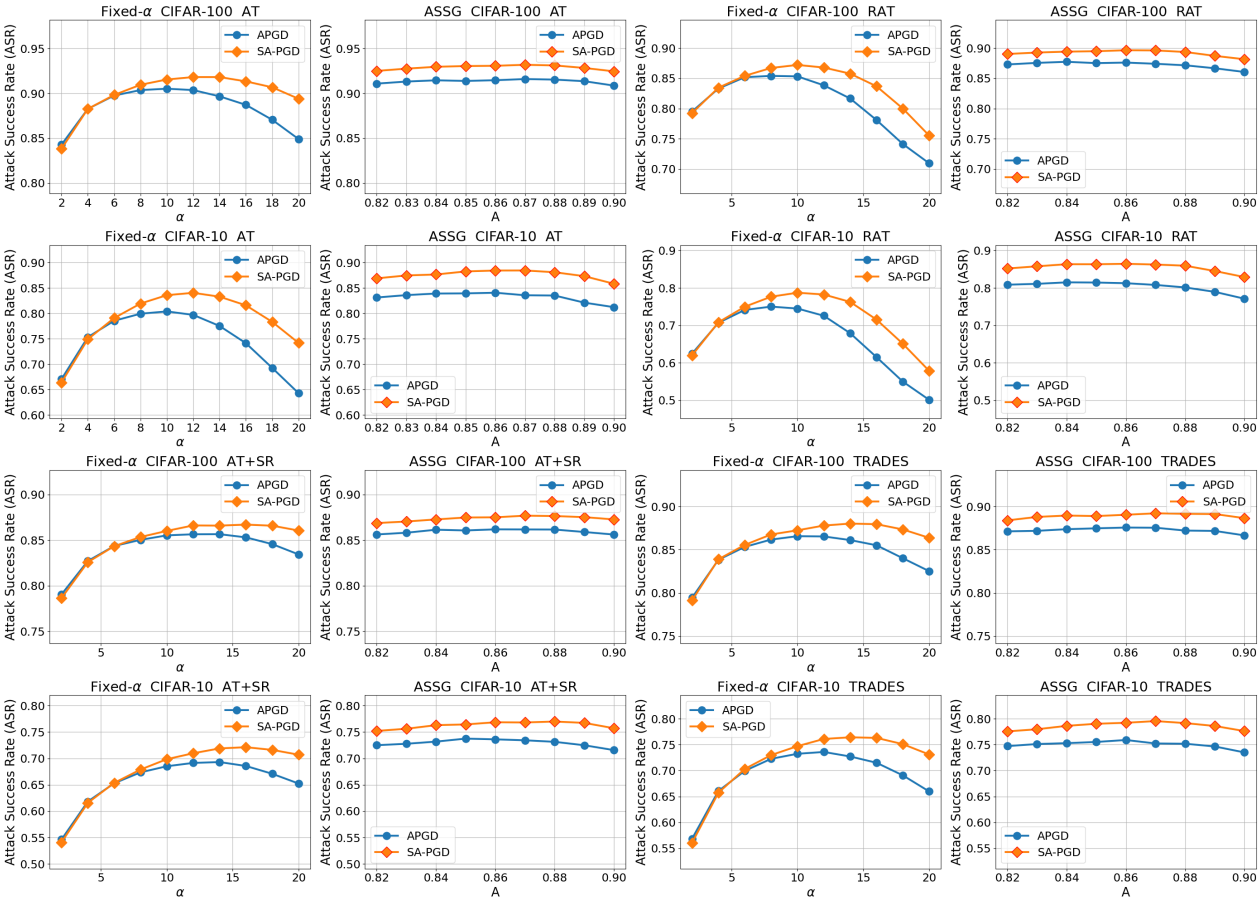

Figure S1. ASR for attacks using Atan surrogate gradients with different values of α and ASSG with different values of A.

Table S1. Computational cost of training one epoch on CIFAR-10

	BPTR	RGA	PDSG	HART	STBP	ASSG
Runtime(s)	7.7	8.0	13.7	11.2	11.3	15.4
Memory Usage(MB)	7161	8171	11265	9473	9473	13577

S4.3. Results on Different Network Architectures and Depths

Tab. S2 reports the attack success rates of various attack methods on SNNs with different network architectures and depths. All target models are trained with AT, and the training settings are consistent with Sec. S3.1. ASSG demonstrates significantly stronger attack performance across different network architectures and depths, indicating its strong generalization capability with respect to network structures.

S4.4. Computational Cost Comparison

All adversarial attacks are implemented and evaluated in PyTorch on a single NVIDIA A100-PCIE-40GB GPU. We compare the memory usage and runtime of different surrogate gradient methods under the same settings. Specifically, we report the GPU memory consumption and time required to execute 100 steps of SA-PGD on a single CIFAR-10 batch (batch size = 256). As shown in the Tab. S1, although ASSG incurs additional memory and runtime due to its spatio-temporally heterogeneous sharpness parameters $\alpha_{i,t}^{l}$, this overhead remains acceptable given the substantial improvements in attack effectiveness.

S4.5. Effectiveness of the Relaxation Term

ASSG incorporates a relaxation term that leverages second-order moment information to prevent underestimation of $\mathbb{E}[|u_{i,t}^{l}|]$. Without this term, the computed sharpness parameter $\alpha_{i,t}^{l}$ may become excessively large, causing the gradient-vanishing degree to exceed the theoretical upper

Table S2. ASR on different network architectures and depths.

Datasets	Architectures	Clean	Attack	BPTR	RGA	PDSG	HART	STBP	**ASSG**
CIFAR -10	VGG9	82.09	APGD	62.86	67.15	67.60	77.39	72.37	**83.97**
			SA-PGD	62.86	67.59	67.04	79.86	72.21	**88.16**
	SEWResNet19	78.69	APGD	66.45	63.94	69.31	77.22	75.38	**84.06**
			SA-PGD	66.85	65.08	68.76	79.49	75.11	**88.44**
	SEWResNet31	77.50	APGD	65.72	60.82	68.38	75.73	74.49	**83.18**
			SA-PGD	65.72	61.63	67.27	77.90	74.31	**87.38**
	SEWResNet43	74.80	APGD	57.86	59.25	66.15	75.61	72.79	**88.05**
			SA-PGD	56.87	59.08	64.73	77.41	71.76	**91.43**
CIFAR -100	VGG9	54.74	APGD	77.28	82.16	81.10	88.06	84.26	**89.94**
			SA-PGD	77.33	82.16	80.63	88.94	84.28	**91.77**
	SEWResNet19	51.90	APGD	82.83	82.50	84.21	89.88	88.22	**91.60**
			SA-PGD	82.99	83.07	84.00	90.80	88.26	**93.19**
	SEWResNet31	51.02	APGD	82.38	80.49	83.86	88.78	87.60	**89.74**
			SA-PGD	82.68	81.08	83.59	89.59	87.66	**91.44**
	SEWResNet43	49.27	APGD	83.24	79.37	84.72	88.00	87.66	**89.00**
			SA-PGD	83.29	80.22	84.46	88.69	87.82	**90.60**

Table S3. Comparison of ASR between ASSG with and without the relaxation term.

Datasets	Methods	$\gamma = 0$	$\gamma = 0.5$	$\gamma = 1.0$	$\gamma = 1.5$	$\gamma = 2.0$	$\gamma = 2.5$	$\gamma = 3.0$	$\gamma = 3.5$
CIFAR -10	AT	86.33	87.45	87.97	88.44	88.83	88.96	89.16	89.30
	RAT	85.39	85.92	86.32	86.44	86.78	86.95	87.22	87.27
	AT+SR	75.08	76.13	76.87	77.00	77.48	77.68	77.75	77.89
	TRADES	77.00	78.36	78.97	79.59	79.86	80.19	80.49	80.57
CIFAR -100	AT	92.45	92.87	93.14	93.19	93.40	93.58	93.67	93.76
	RAT	89.02	89.44	89.55	89.64	89.75	89.80	89.93	89.89
	AT+SR	86.72	87.22	87.53	87.68	87.87	87.92	87.98	88.11
	TRADES	88.43	88.91	89.05	89.22	89.35	89.40	89.46	89.43

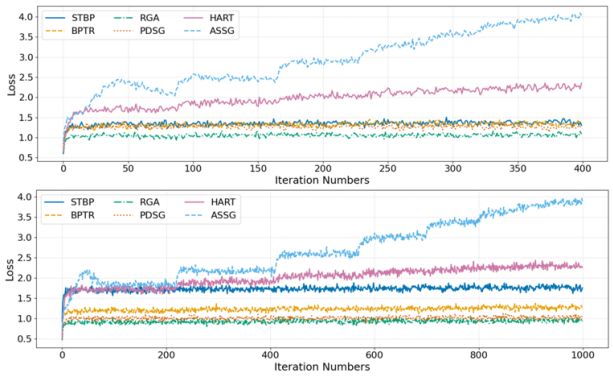

Figure S2. The loss curves of SA-PGD for different gradient approximation methods and iteration numbers.

bound. Table S3 presents a comparison of ASR between ASSG with different relaxation coefficients. As the appropriate range for tuning the parameter A depends on factors such as the relaxation coefficient, the dataset, and the neuron model, we use a ternary search over the interval $[0.6, 0.95]$ to identify the optimal A under each configuration.

As shown in Tab. S3, introducing the relaxation term effectively improves the ASR of ASSG. Although the ASR is relatively robust to the choice of the relaxation coefficient, larger relaxation coefficients generally yield higher ASR. We leave a deeper investigation of this phenomenon—particularly the role of second-order moment information in improving gradient estimation—to future work.