Towards Effective and Sparse Adversarial Attack on Spiking Neural Networks via Breaking Invisible Surrogate Gradients

Supplementary Material

S1. Derivation of Potential-Dependent Surrogate Gradient

We adopt the two-point zeroth-order method to calculate the gradient of the firing function approximately [22]:

$$G^2(u;z,\delta)=\frac{h(u+z\delta-V_{th})-h(u-z\delta-V_{th})}{2\delta}z. \tag{S1}$$

Due to the firing function defined in Eq. (2) of the main text, the two-point zeroth-order can be simplified as:

$$G^2(u;z,\delta)=\begin{cases}\frac{|z|}{2\delta}, & |u-V_{th}|<|z\delta|\\ 0, & \text{otherwise}\end{cases}. \tag{S2}$$

Since z is sampled from the distribution λ, the surrogate gradient equals to the expectation of the two-point zeroth-order [20]:

$$\begin{aligned}\frac{\partial s}{\partial u}&=\mathbb{E}_{z\sim\lambda}[G^2(u;z,\delta)]\\&=\int_{-\infty}^{+\infty}G^2(u;z,\delta)\lambda(z)dz\\&=2\int_{\frac{|u-V_{th}|}{\delta}}^{\infty}\frac{|z|}{2\delta}\lambda(z)dz\\&=\int_{\frac{|u-V_{th}|}{\delta}}^{\infty}\frac{|z|}{\delta}\lambda(z)dz.\end{aligned} \tag{S3}$$

As demonstrated in Sec. 4.1, $u+z\delta$ follows a normal distribution $\mathcal{N}(\mu,\sigma^2)$, where μ denotes the mean of membrane potential, and σ is the standard deviation of the membrane potential. Therefore, $z\sim\mathcal{N}(\frac{\mu-u}{\delta},\frac{\sigma^2}{\delta^2})$. Following the requirement $z\sim\mathcal{N}(0,1)$ in the two-point zeroth-order method, we set $\delta=\sigma$, and when $u\approx\mu$, we get:

$$\begin{aligned}\frac{\partial s}{\partial u}&=\int_{\frac{|u-V_{th}|}{\boldsymbol{\sigma}}}^{\infty}\frac{|z|}{\boldsymbol{\sigma}}\cdot\frac{1}{\sqrt{2\pi}}\exp(-\frac{z^2}{2})dz\\&=\frac{1}{\sqrt{2\pi}\boldsymbol{\sigma}}\int_{\frac{|u-V_{th}|}{\boldsymbol{\sigma}}}^{\infty}\exp(-\frac{z^2}{2})d(\frac{z^2}{2})\\&=\frac{1}{\sqrt{2\pi}\boldsymbol{\sigma}}\exp(-\frac{(\boldsymbol{u}^l[t]-V_{th})^2}{2\boldsymbol{\sigma}^2}).\end{aligned} \tag{S4}$$

Here, we follow the TAB [14] to adopt the temporal accumulated channel-wise standard deviation $\boldsymbol{\sigma}$ of membrane potential.

S2. Algorithm of Sparse Dynamic Attack

Algorithm 1 Sparse Dynamic Attack (SDA)

Input: Classifier f, benign image $\boldsymbol{x}$, label y.
Parameters: Initial gradient selection count k_{init}, maximum number of iterations N.
Output: Adversarial example $\boldsymbol{x}_{adv}$.

#Generation Process:
Initialize perturbation mask $\boldsymbol{m}\leftarrow 0$
Initialize contributing FDs $\boldsymbol{FD}^c\leftarrow\infty$
Initialize $\boldsymbol{x}^0\leftarrow\boldsymbol{x}$
for $n=0$ to $N-1$ **do**
  Calculate the gradient $\boldsymbol{g}(\boldsymbol{x}^n)$ ▷ Eq. (12)
  $\boldsymbol{g}^c\leftarrow\boldsymbol{g}\cdot((1-2\boldsymbol{x}^n)\cdot\boldsymbol{g}<=0)\cdot(1-\boldsymbol{m})$ ▷ Eq. (14)
  $k\leftarrow(n+1)k_{init}$
  $p_1,p_2,\ldots,p_k\leftarrow\text{argtop}k(|\boldsymbol{g}^c|)$ ▷ Eq. (15)
  for $i=1$ to k **do** ▷ Parallelized
    Calculate $FD_{p_i}(\boldsymbol{x}^n)$ ▷ Eq. (16)
    if $(1-2x^n_{p_i})\cdot FD_{p_i}<=0$ **then** ▷ Eq. (17)
      $FD^c_{p_i}\leftarrow FD_{p_i}$
      $m_{p_i}\leftarrow 1$
    end if
  end for
  $\boldsymbol{x}^{n+1}\leftarrow\boldsymbol{x}\cdot(1-\boldsymbol{m})+(1-\boldsymbol{x})\cdot\boldsymbol{m}$ ▷ Perturb
  if $\boldsymbol{x}^{n+1}$ is adversarial **then**
    $\boldsymbol{x}_{adv}\leftarrow\boldsymbol{x}^{n+1}$
    break
  end if
  if $n==N-1$ **then**
    Attack failed
  end if
end for
#Reduction Process:
Construct sorted perturbed indices $\mathcal{S}$ ▷ Eq. (18)
Initialize $L\leftarrow 0, R\leftarrow\text{len}(\mathcal{S})-1$
while $L<=R$ **do**
  $j\leftarrow\lfloor\frac{L+R}{2}\rfloor$
  $\boldsymbol{x}'_{adv}\leftarrow\boldsymbol{x}_{adv}$
  $\boldsymbol{x}'_{adv}[\mathcal{S}[0:j+1]]\leftarrow 1-\boldsymbol{x}'_{adv}[\mathcal{S}[0:j+1]]$
  if $\boldsymbol{x}'_{adv}$ is adversarial **then**
    $L\leftarrow j+1$
    $\boldsymbol{x}_{final}\leftarrow\boldsymbol{x}'_{adv}$
  else
    $R\leftarrow j-1$
  end if
end while
return final adversarial example $\boldsymbol{x}_{final}$

S3. Adversarial Threat Model

As shown in Fig. S1, in white-box attacks, the attacker leverages gradients to perform attacks. As the activation in ANNs has a well-defined gradient, the gradients can be directly calculated through the model weights and architecture, and the attacker does not require training details, which are useless for attack.

In contrast, the activation in SNNs does not have exact backward function. During the training stage, the surrogate gradient is adopted as the backward function. However, the inference model does not store the backward function used during training; further, as shown in Tab. 3, adopting it for attack does not guarantee the performance. Therefore, the **invisible surrogate gradients** means: the backward function in training stage is invisible during inference and attack, and the optimal backward function is uncertain.

In summary, the adversarial threat model in our paper is identical to white-box ANN attacks, which is: **the attacker knows the weights, architecture, and the activation's forward function of the victim inference model.** The backward function is inaccessible. This adversarial threat model is suitable for real-world situations: **the attacker obtains a neuromorphic device, where the backward function is served as a training skill and not stored in the device.** Instead of adopting model-independent backward functions [3, 11], our adaptive PDSG effectively increases the attack success rate.

S4. Details of Experiments

Details of datasets. CIFAR10/100 [16] dataset contains 60,000 images with 10/100 classes, which are split into the training set with 50,000 images and test set with 10,000 images. The input size is 32×32.

ImageNet [7] dataset contains 1,281,167 images as training set and 50,000 images as validation set. The number of classes is 1000, and the input size is 224×224.

NMNIST [23] dataset is constructed by saccading the MNIST dataset [17] using DVS. The training set contains 60000 samples, and the test set contains 10000 samples. The size of frames is 34×34.

DVS-Gesture [1] dataset includes samples of hand gesture recorded by DVS128 camera. The training set contains 1176 samples, and the test set contains 288 samples. The size of frames is 128×128.

CIFAR10-DVS [18] dataset is converted from CIFAR10 [16] dataset, including 10000 samples with 10 classes. We split these 10000 samples into 9000 training samples and 1000 test samples. The size of frames is 128×128.

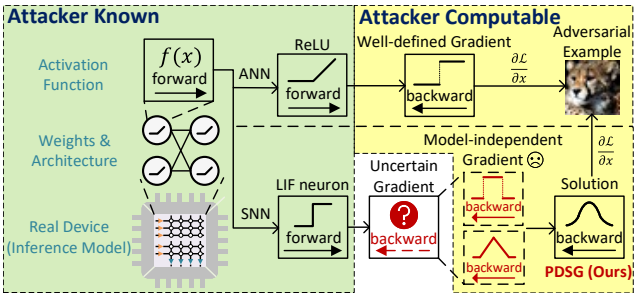

Figure S1. White-box attacker knows weights and architecture of the model, which is enough for attacking ANNs. For SNNs, the gradient of activation is not exposed during the inference. Our PDSG provides a solution for the problem of uncertain gradient.

Details of models. We adopt spiking ResNet-18 [13], spiking VGG-11 [3], VGGSNN [8], PLIFNet [10], and hierarchical spiking transformer (HST) [27] in our experiments. The spiking ResNet-18 and spiking VGG-11 maintain the same architecture as the original ResNet-18 [12] and VGG-11 [24], respectively, with the activation function replaced by LIF neurons. The VGGSNN removes the last two linear layers of the spiking VGG-11. The PLIFNet contains three convolutional layers and two linear layers for NMNIST classification. The HST attains 84.28% accuracy on ImageNet, surpassing other current spiking transformer architectures.

The timestep of models for static datasets is set to 4, and for dynamic datasets is set to 10. We adopt $\tau = 0.5$ and $V_{th} = 1$ for all LIF neurons.

Training details. All experiments are conducted on NVIDIA Tesla A100 GPU with 40GB memory. We train all SNN models with STBP [25] for 600 epochs (static datasets) or 200 epochs (dynamic datasets). We adopt the stochastic gradient descent optimizer with 0.1 learning rate and 0.9 momentum for spiking ResNet-18, and adopt the adam [15] optimizer with 0.001 learning rate for other models. The weight decay is set to 0, and we use the cosine annealing scheduler [21] to adjust the learning rate. Additionally, TET [8] loss is utilized to improve the accuracy. The seed is set to 0 across all experiments.

S5. Details of Fixed Surrogate Gradients

In Sec 4.3 of the main text, we conduct extensive experiments to validate the effectiveness of various attack-phase SG, including fixed SGs and our PDSG. The fixed SGs consist of rectangular SG [25], triangle SG [8], and ATan SG [9]. The rectangular SG is described as:

$$\frac{\partial s}{\partial u} = \begin{cases} \frac{1}{2w}, & -w < |u - V_{th}| < w \\ 0, & \text{otherwise} \end{cases}. \tag{S5}$$

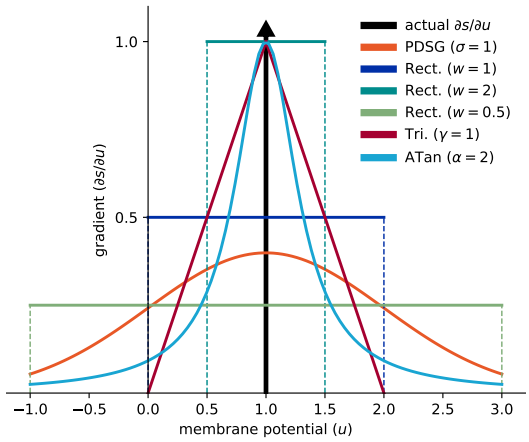

Figure S2. Illustration of various fixed SGs and our PDSG.

Here w represents the width of the SG. Typically, w is a hyper-parameter, and we adopt $w = 1, 2, 0.5$ in experiments. The triangle SG is:

$$\frac{\partial s}{\partial u} = \frac{1}{\gamma^2}\max\{0, \gamma - |u - V_{th}|\}. \tag{S6}$$

Here γ controls the shape of the SG, and we set $\gamma = 1$ in our experiments, which is the default setting in [8]. The ATan SG is denoted as:

$$\frac{\partial s}{\partial u} = \frac{\alpha}{2(1 + (\frac{\pi}{2}\alpha(u - V_{th}))^2)}. \tag{S7}$$

We use the default $\alpha = 2$ in our experiments. We depict all SGs above and our PDSG in Fig. S2.

S6. Visualization

In this section, we present the visualization result of our SDA and the SpikeFool [4] in attacking binary dynamic images. The visualization on NMNIST dataset is shown in Fig. S3. After attacking, the label of the original image is changed from 5 to 8. Our SDA modifies 144 pixels, which is only 0.62% of the pixels of the image. In contrast, the SpikeFool modifies 321 pixels, indicating that the perturbations are easier to be detected.

The visualization on DVS-Gesture dataset is displayed in Fig. S4. Our SDA only modifies 0.1% of the pixels, rendering the adversarial example virtually indistinguishable from the original image to both human observers and automated detection systems.

We also depict the visualization result on CIFAR10-DVS dataset in Fig. S5. In this case, our SDA modifies a mere 0.05% of the pixels, and the perturbations only exist in the first two frames. Therefore, it suggests that the model focuses on the first two frames to perform classification, and our SDA exploits this behavior to generate imperceptible perturbations.

T	Acc. (%)	Attack	Static Evaluation	Dynamic Evaluation		
			ASR. (%) ($\ell_0 < 200/800$)	ASR. (%)	Mean ℓ_0	Median ℓ_0
5	76.5	SpikeFool	45.0/99.0	**100.0**	270.24	230.00
		SDA(Ours)	**77.0/100.0**	**100.0**	**131.08**	**86.50**
10	78.2	SpikeFool	19.0/70.0	**100.0**	674.89	491.00
		SDA(Ours)	**38.0/82.0**	**100.0**	**458.02**	**303.00**
20	82.4	SpikeFool	4.0/11.0	72.0	**3733.49**	2705.00
		SDA(Ours)	**5.0/13.0**	**89.0**	4905.00	**2570.00**

Table S1. Discussion of attacking spiking ResNet-18 with various timesteps on CIFAR10-DVS dataset. T denotes the timestep, and ASR. denotes the attack success rate. The best results are in bold.

S7. Discussion of Timesteps in Binary Attack

Since the performance of the SNN model depends on the timestep, we discuss the impact of the timestep in attacking binary dynamic images. As the imperceptibility of the SCG [19] and the GSAttack [26] is insufficient, we only compare our SDA with the SpikeFool [4]. The results of attacking spiking ResNet-18 on CIFAR10-DVS dataset is illustrated in Tab. S1. Our SDA outperforms the SpikeFool in terms of the attack success rate and sparsity. In timestep $= 20$, the difficulty of the attacks increases as the number of the input pixels is large, while our SDA still exhibits stable performance of 89% ASR. Since we only record ℓ_0 of successful attack, the mean of ℓ_0 of our SDA is higher than that of SpikeFool. Notably in timestep $= 5$, our SDA achieves a median of 86.5 ℓ_0, which is only 0.05% of the input pixels.

S8. Discussion of Initial Selection Count in SDA

In this section, we conduct experiments with various choices of k_{init} in our SDA. The results are shown in Tab. S2. We first set $k = k_{init}$ for each iteration, indicating that the k is fixed. As the calculation of the gradients is a course estimation , significant gradients are easy to be ignored when k is fixed at 10, causing low attack success rate. The mean and median of ℓ_0 is extremely low since we only record ℓ_0 and count of iterations for successful attacks. Therefore, selecting a fixed low k induces poor attack performance. Conversely, setting a fixed $k = 100$ achieves 100% attack success rate but at the cost of a relatively larger ℓ_0.

To achieve a stable attack and avoid the hyper-parameter significantly influencing the performance of the attack, we adopt the incremental k strategy in our SDA. The motivation comes from preventing gradient vanishing. In the early stages of the generation process, the model's output is distant from the classification boundary, causing substantial gradients becoming zero. In this case, only a few gradients are valid and we only require to leverage these gradients to calculate their FDs. However, in the later stages, any

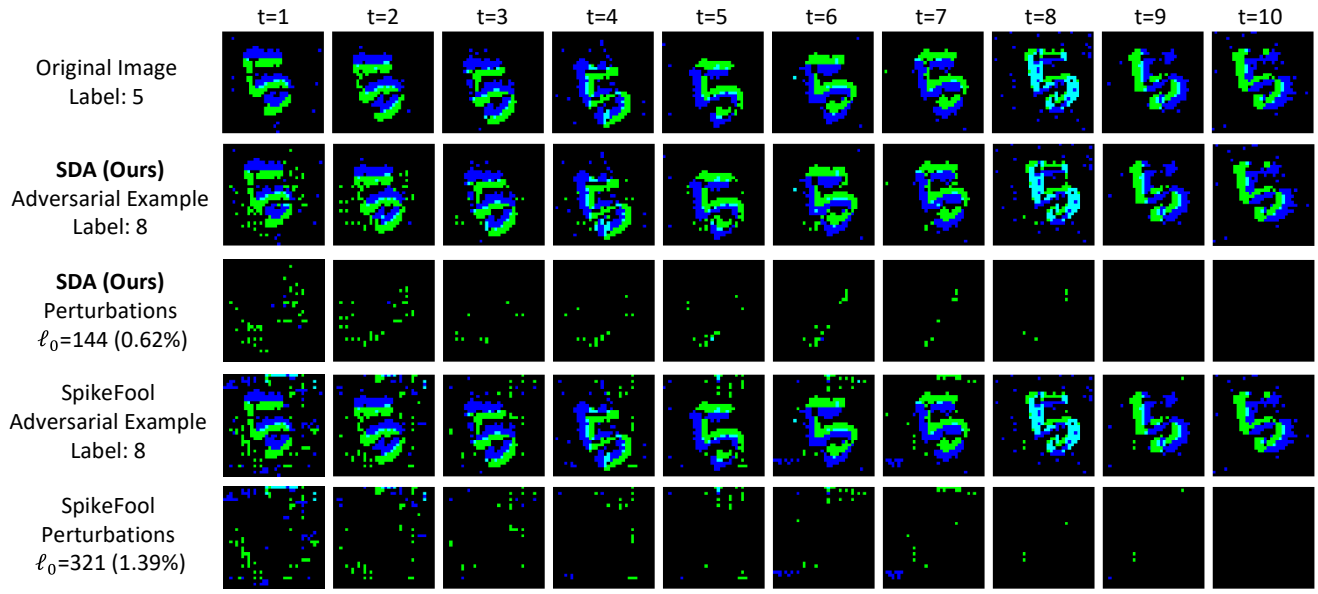

Figure S3. Visualization of the our SDA and SpikeFool on NMNIST dataset. The channel of $p = on$ and $p = off$ is indicated in green and blue color, respectively. Our SDA modifies only 0.62% of pixels to change the classification result from 5 to 8.

k_{init}	Static Evaluation	Dynamic Evaluation			
	ASR. (%) ($\ell_0 < 200/800$)	ASR. (%)	Mean ℓ_0	Median ℓ_0	Mean Iterations
10 (Fixed)	12.0/12.0	12.0	13.67	12.50	3.17
20 (Fixed)	27.0/32.0	32.0	85.47	54.00	10.53
50 (Fixed)	38.0/82.0	92.0	361.86	256.50	16.78
100 (Fixed)	34.0/83.0	100.0	464.53	309.50	9.97
1 (Incremental)	41.0/84.0	99.0	426.97	280.00	56.11
5 (Incremental)	38.0/84.0	100.0	439.55	285.50	17.82
10 (Incremental)	38.0/82.0	100.0	458.02	303.00	12.33
20 (Incremental)	37.0/80.0	100.0	466.72	314.50	8.70
50 (Incremental)	34.0/75.0	100.0	518.95	341.00	5.64

Table S2. Attack success rate and dynamic evaluation for attacking spiking ResNet-18 on CIFAR10-DVS dataset with various choices of k_{init} in our SDA. Fixed represents k is equal to k_{init} in each iteration. Incremental denotes k is incremental by k_{init} in each iteration.

τ	Acc. (%)	Attack	ASR. ($\epsilon = 2/255$) / ($\epsilon = 8/255$)			
			STBP	RGA	HART	**PDSG (Ours)**
0.25	94.52	FGSM	41.81/63.79	29.88/49.35	42.90/57.80	**45.04/82.88**
		PGD	73.48/99.89	62.00/96.05	**79.68**/99.41	70.41/**99.99**
0.5	94.72	FGSM	38.21/52.36	31.14/45.80	37.30/46.72	**43.98/79.56**
		PGD	66.74/99.81	61.97/92.47	66.77/98.64	**69.62/100.0**
0.75	94.33	FGSM	32.67/45.22	26.99/34.47	33.62/43.74	**42.82/79.64**
		PGD	60.60/99.42	48.69/89.30	61.16/97.71	**68.00/99.96**
1.0	94.24	FGSM	29.57/40.57	27.48/35.42	31.52/41.99	**43.37/77.98**
		PGD	54.35/95.39	48.10/87.99	57.23/94.89	**68.21/99.94**

Table S3. Attack success rate for attacking spiking ResNet-18 with various leakage factors on CIFAR10 dataset. τ denotes the leakage factor. The best results are in bold.

modified pixel could potentially make the input adversarial, necessitating consideration of a wider range of pixels with contributing gradients. Consequently, we adopt the incremental k in our SDA, indicating that k is incremental by k_{init} in each iteration.

As shown in Tab. S2, the sparsity of perturbations decreases with an increase of k_{init}. Since the contributing FDs and reduction process effectively remove redundant perturbations, the ℓ_0 and attack success rate will not change drastically with variations in k_{init}. However, an extremely low k_{init} may cause failed attacks (99% ASR in $k_{init} = 1$). Additionally, a low k_{init} implies that the generation process requires more iterations to find an adversarial example, thus increasing the attack time. Therefore, to make a trade-off between the imperceptibility of perturbations and the time costs of the attack, while ensuring 100% attack success rate, we choose $k_{init} = 10$ in our SDA.

S9. Discussion of Leakage Factors

To verify the generalization abilities of our PDSG and SDA, we conduct experiments on models with various leakage factors. First, we validate the performance of our PDSG in attacking spiking ResNet-18 on CIFAR10 dataset. The results are illustrated in Tab. S3. Although our PDSG is surpassed by HART [11] in PGD ($\epsilon = 2/255$) attack when $\tau = 0.25$, likely due to the compatibility of HART's surrogate function with the model, our PDSG exhibits superior performance in all other experiments.

In Tab. S4, we demonstrate the performance of our SDA

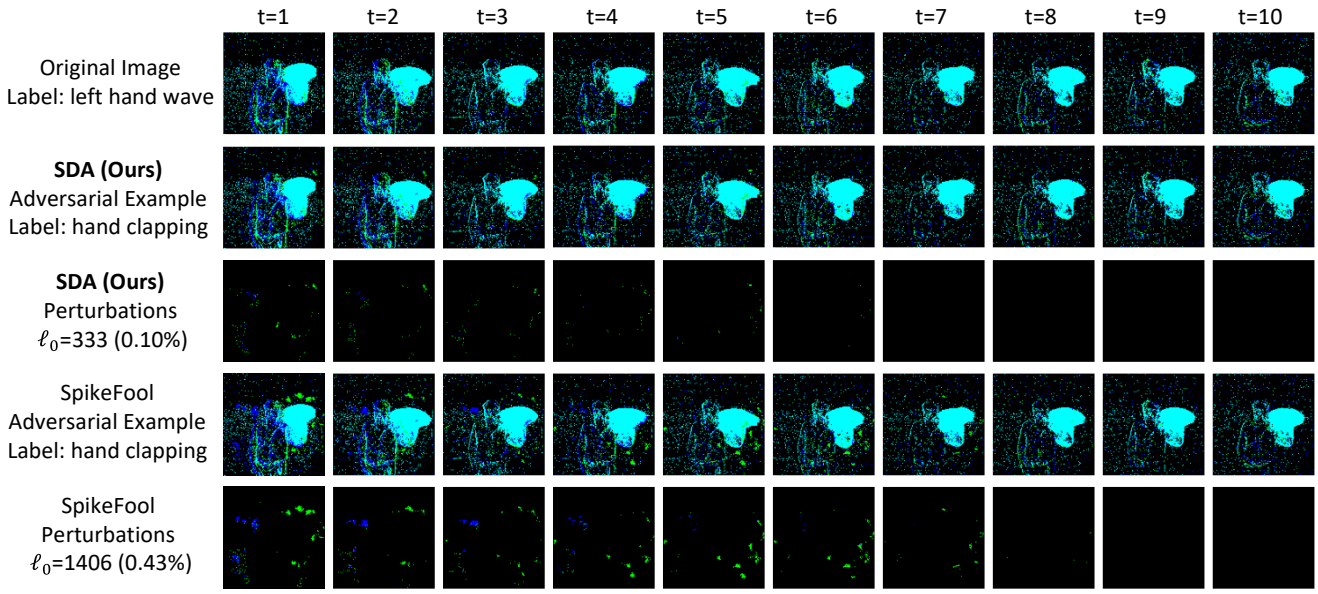

Figure S4. Visualization of the our SDA and SpikeFool on DVS-Gesture dataset. The channel of $p = on$ and $p = off$ is indicated in green and blue color, respectively. Our SDA modifies only 0.10% of pixels to change the classification result from left hand wave to hand clapping.

τ	Acc. (%)	Attack	Static Evaluation	Dynamic Evaluation		
			ASR. (%) ($\ell_0 < 200/800$)	ASR. (%)	Mean ℓ_0	Median ℓ_0
0.25	82.5	SpikeFool	18.0/48.0	**100.0**	1263.20	896.50
		SDA(Ours)	**27.0/67.0**	**100.0**	**639.13**	**402.00**
0.5	78.2	SpikeFool	19.0/70.0	**100.0**	674.89	491.00
		SDA(Ours)	**38.0/82.0**	**100.0**	**458.02**	**303.00**
0.75	78.0	SpikeFool	38.0/87.0	**100.0**	374.43	271.00
		SDA(Ours)	**57.0/92.0**	**100.0**	**261.16**	**152.50**
1.0	76.9	SpikeFool	39.0/97.0	**100.0**	309.43	253.00
		SDA(Ours)	**67.0/99.0**	**100.0**	**175.59**	**105.50**

Table S4. Attack success rate and dynamic evaluation for models with various leakage factors on binary dynamic images. τ denotes the leakage factor. The best results are in bold.

Dataset	Architecture	ASR. (%) ($\epsilon = 2/255$)			ASR. (%)($\epsilon = 8/255$)		
		PDSG (PGD)	Square	RayS	**PDSG** (PGD)	Square	RayS
CIFAR10	ResNet18	**69.62**	29.79	13.40	**100.00**	66.10	52.96
	ResNet18 (Adv. trained)	10.68	**44.79**	18.10	**62.16**	56.54	33.30
	VGG11	**39.20**	26.86	12.81	**99.71**	57.42	56.92
CIFAR100	ResNet18	**78.50**	50.67	32.64	**99.83**	78.80	71.80

Table S5. Attack success rate under comparison with black-box attacks. All inputs adopt direct coding. The best results are in bold.

in attacking spiking ResNet-18 on CIFAR10DVS dataset. Our SDA outperforms the SpikeFool across diverse leakage factors. It is noteworthy that the ℓ_0 of the perturbations increases as the leakage factor decreases, indicating that the model may exhibit greater robustness with a lower leakage factor.

S10. Comparison with Black-box Attacks

In contrast to white-box attacks, black-box attacks also threaten neural network models. Without accessing the weights and architectures of the models, black-box attacks only require the inputs and outputs of models, and leverage them to generate adversarial examples. Transfer-based black-box attacks are already evaluated in [3, 11]. To validate our PDSG's ability of optimizing the gradient flow, we conduct experiments with score-based black-box Square Attack [2] and decision-based black-box attack RayS [5] in Tab. S5. The results demonstrate that our PDSG outperforms black-box attacks across various models and datasets, except adversarially trained models. As ResNet18 is specifically adversarially trained by PGD attack with $\epsilon = 2/255$, the PDSG performs poorly when $\epsilon = 2/255$. However, when the attack intensity increases to $\epsilon = 8/255$, our PDSG surpasses other black-box attacks.

S11. Results of Adaptive Attack

In attacking static images, we conduct experiments of APGD [6] attack, which is an adaptive version of the PGD attack. In Tab. S6, the results show the same trend as the PGD attack in Tab. 1, demonstrating that our PDSG performs the best and has stable performance.

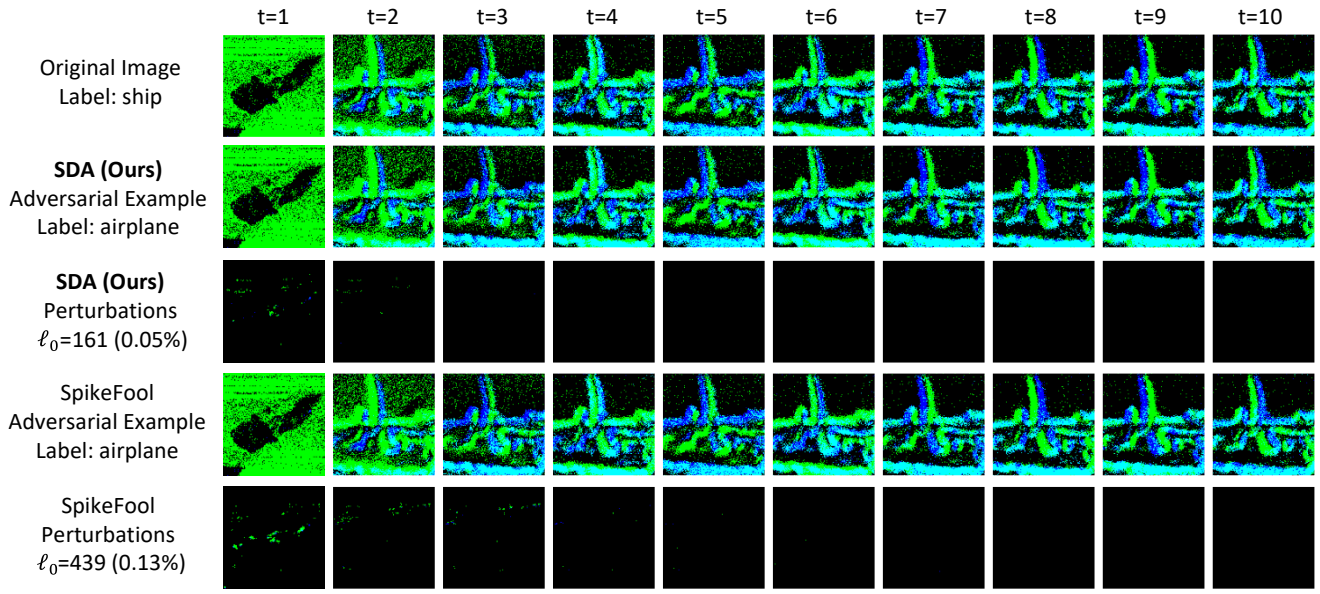

Figure S5. Visualization of the our SDA and SpikeFool on CIFAR10-DVS dataset. The channel of $p = on$ and $p = off$ is indicated in green and blue color, respectively. Our SDA modifies only 0.05% of pixels to change the classification result from ship to airplane.

Dataset	Architecture	ASR. (%) ($\epsilon = 2/255$)				ASR. (%) ($\epsilon = 8/255$)			
		STBP	RGA	HART	**PDSG (Ours)**	STBP	RGA	HART	**PDSG (Ours)**
CIFAR10	ResNet18	71.36	67.04	71.49	**75.18**	99.67	94.77	98.54	**99.97**
	ResNet18 (Adv. trained)	14.34	17.93	21.20	**21.56**	41.39	57.37	70.74	**71.92**
	VGG11	46.96	46.40	**54.82**	45.58	99.25	88.51	98.13	**99.84**
CIFAR100	ResNet18	85.12	82.62	**89.54**	83.21	99.68	98.62	99.67	**99.91**

Table S6. Comparison with state-of-the-art approaches on attacking static images using APGD attack. ASR. denotes the attack success rate. ϵ is the attack intensity. STBP denotes attacking using training-phase SG. All inputs adopt direct coding. The best results are in bold.

S12. Evaluation of Computational Cost

To evaluate the computational cost of our method, we adopt $batch\_size = 1$ and perform attacks on both static and dynamic datasets. In Tab. S7, since our PDSG requires the computation of the standard deviation of membrane potential, the efficiency of our PDSG is slightly lagging behind. As shown in Tab. S8, when attacking binary dynamic images, our SDA performs more efficiently than SpikeFool. Although SCG and GSAttack execute fast, their ℓ_0 are much larger than ours. Specifically, when cooperating with the PDSG, our SDA achieves a significant efficiency improvement, since the PDSG optimizes the gradient flow and effectively reduces the number of iterations.

Dataset	Architecture	Attack	Attack time per sample (s)			
			STBP	RGA	HART	**PDSG (Ours)**
CIFAR10	ResNet18	FGSM	0.33	0.31	0.40	0.55
		PGD	2.15	1.58	2.08	3.22

Table S7. Computational costs in attacking static images.

Dataset	Architecture	Gradient	Attack time per sample (s)			
			SCG	SpikeFool	GSAttack	**SDA (Ours)**
N-MNIST	PLIFNet	STBP	0.26	12.44	3.65	7.48
		PDSG (Ours)	0.24	27.18	2.14	0.95

Table S8. Computational costs in attacking binary dynamic images.

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

[2] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient

black-box adversarial attack via random search. In *European conference on computer vision*, pages 484–501. Springer, 2020. 5

[3] Tong Bu, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. Rate gradient approximation attack threats deep spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7896–7906, 2023. 2, 5

[4] 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. 3

[5] Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1739–1747, 2020. 5

[6] 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. 5

[7] 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. 2

[8] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022. 2, 3

[9] 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. 2

[10] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021. 2

[11] Zecheng Hao, Tong Bu, Xinyu Shi, Zihan Huang, Zhaofei Yu, and Tiejun Huang. Threaten spiking neural networks through combining rate and temporal information. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 4, 5

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[13] Yulong Huang, Xiaopeng LIN, Hongwei Ren, Haotian FU, Yue Zhou, Zunchang LIU, biao pan, and Bojun Cheng. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. In *Forty-first International Conference on Machine Learning*, 2024. 2

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

[15] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009. 2

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[18] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017. 2

[19] 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. 3

[20] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020. 1

[21] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 2

[22] Bhaskar Mukhoty, Velibor Bojkovic, William de Vazelhes, Xiaohan Zhao, Giulia De Masi, Huan Xiong, and Bin Gu. Direct training of snn using local zeroth order method. *Advances in Neural Information Processing Systems*, 36: 18994–19014, 2023. 1

[23] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015. 2

[24] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[25] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018. 2

[26] Yanmeng Yao, Xiaohan Zhao, and Bin Gu. Exploring vulnerabilities in spiking neural networks: Direct adversarial attacks on raw event data. In *ECCV*, 2024. 3

[27] Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. QKFormer: Hierarchical spiking transformer using q-k attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2