Rate Gradient Approximation Attack Threats Deep Spiking Neural Networks

Tong Bu, Jianhao Ding, Zecheng Hao, Zhaofei Yu*
Peking University
putong30@pku.edu.cn, djh01998@stu.pku.edu.cn, {zechenghao, yuzf12}@pku.edu.cn

Abstract

Spiking Neural Networks (SNNs) have attracted significant attention due to their energy-efficient properties and potential application on neuromorphic hardware. State-of-the-art SNNs are typically composed of simple Leaky Integrate-and-Fire (LIF) neurons and have become comparable to ANNs in image classification tasks on large-scale datasets. However, the robustness of these deep SNNs has not yet been fully uncovered. In this paper, we first experimentally observe that layers in these SNNs mostly communicate by rate coding. Based on this rate coding property, we develop a novel rate coding SNN-specified attack method, Rate Gradient Approximation Attack (RGA). We generalize the RGA attack to SNNs composed of LIF neurons with different leaky parameters and input encoding by designing surrogate gradients. In addition, we develop the time-extended enhancement to generate more effective adversarial examples. The experiment results indicate that our proposed RGA attack is more effective than the previous attack and is less sensitive to neuron hyperparameters. We also conclude from the experiment that rate-coded SNN composed of LIF neurons is not secure, which calls for exploring training methods for SNNs composed of complex neurons and other neuronal codings. Code is available at https://github.com/putshua/SNN_attack_RGA

1. Introduction

As the third generation of artificial neural networks [47], Spiking Neural Networks (SNNs) have gained more attraction due to their spatio-temporal, discrete representation, and event-driven properties. These bio-inspired neural networks borrow the characteristics of spiking representations and neuronal dynamics from biological brains [23, 75]. Unlike traditional Analog Neural Networks (ANNs), SNNs utilize spiking neurons as their essential components, which accumulate current over time, emit spikes when the membrane potential exceeds the threshold, and pass on information through spike trains. The natural sparsity of the spike trains leads to the low power consumption of SNNs [59, 69].

SNNs are competitive in real-world vision applications. The development of neuromorphic computing [10, 11, 20, 54, 56, 76] has further magnified the advantages of low-power consumption properties of SNNs, so that they can be deployed in power-limited scenarios [8, 64], such as edge computing or mobile application. However, the training algorithms of SNNs are also improving. The most practical training methods are ANN-SNN conversion [7], supervised training [72], and hybrid training [57, 58].

When SNNs are applied to safety-critical systems, the reliability of SNNs should be a major concern. The adversarial attack is one of the most significant categories that threatens model security [24, 68]. Similar to ANNs, SNNs can also be fooled by crafting adversarial examples that are imperceptible to human eyes from gradient-based backpropagation [62], which may lead to catastrophic consequences when SNNs are deployed in safety-related scenarios. Nevertheless, SNNs are still considered to be more robust than ANNs. This robustness comes from inherent neural dynamics, such as forgetting historical information and discrete spikes [63]. Besides, the robustness of SNNs can be improved through special structural enhancements [9] or training techniques [37, 45, 71].

Effective attack examples of ANNs can be crafted from well-defined gradients on the activation functions [68]. For SNNs, a common way to construct gradient-based attacks is by backpropagating through a surrogate function over discrete spikes. In this way, the gradient may suffer from explosion and vanishment in temporal and layer-by-layer communication [72]; at the same time, the membrane potential of all historical time steps needs to be saved when backpropagation, which requires a large amount of memory. Currently, high-performance SNNs typically combine leaky integrate-and-fire models and rate-encoded inputs. While the rate coding scheme brings excellent performance to SNN, it also exposes shortcomings. If the rate coding nature in SNN is considered, can we construct a more powerful attack? After all, the activation functions of many ANNs are inspired by the firing rate of biological neurons [52].

* Corresponding author

In this paper, we develop a novel Rate Gradient Approximation Attack (RGA) based on components of rate coding in high-performance SNNs. RGA attack is more effective than previously used attacks as it makes better use of the rate coding feature. We expect our work to provide benchmarks for SNN defense against adversarial attacks and inspire future research for SNNs. The main contributions of this paper are:

- We observe that layers in SNN are mainly communicated by rate coding, either for converted SNN or for surrogate-trained SNN.
- We develop the Rate Gradient Approximation Attack based on rate coding and apply it to SNNs composed of different types of neurons and input codings. We further propose a time-extended variant to get more effective adversarial examples.
- Experiments prove that the RGA attack outperforms the STBP attack and is less sensitive to neuron hyper-parameters. Based on the proposed attack, we compare the robustness of SNNs using different leaky parameters with that of ANNs and manifest that the SNNs composed of LIF neurons cannot provide strong enough security. This conclusion inspires further research on networks with more complex neurons.

2. Related Works

Learning of SNNs. The most efficient and commonly used SNNs training methods are ANN-SNN conversion [7] and Spatio-temporal Backpropagation (STBP) [72]. The core idea of ANN-SNN conversion is mapping the weights of a pre-trained ANN into an SNN. Researchers found that adjusting the weights or threshold in the SNN can balance the trade between inference time-steps and performance after conversion [12, 14, 16, 25, 28, 50, 61]. Some recent works quantized the source ANN to boost the performance of converted SNNs using ultra-low time-steps [5, 41, 43, 74]. ANN-SNN conversion is the most practical training method to train SNNs on large-scale datasets, and the converted SNNs always have outstanding performance [6, 26, 27, 34]. The temporal characteristics of the spiking neuron make it similar to a Recurrent Neural Network (RNN). Based on this, Wu et al. [72] borrowed the idea of Back Propagation Through Time [70] and proposed the supervised learning way of STBP. As the firing of neurons is a non-differentiable Heaviside function, the surrogate gradient approximation is proposed to smooth the gradient [19,21,40,53,65,67,77]. Nowadays, the performance of SNNs trained with backpropagation is comparable to that of ANNs [13,17,22,29,33,35,39,73,80]. Time-based backpropagation is another supervised learning method that can maintain the sparsity of the gradient. However, these works can only extend to shallow networks [3, 51, 78, 79, 81].

The robustness of SNN. SNNs are vulnerable to adversarial attacks as ANNs by adopting the gradient scheme in training. Firstly, SNNs obtained from ANN-SNN conversion can be attacked by the source ANNs with shared weights [62]. Besides, end-to-end trained SNNs suffer from the attack constructed from STBP. Sharmin et al. [62] summarized the two types of attacks as ANN-crafted and SNN-crafted attacks and revealed that attacks based on STBP are believed to be more powerful than those based on ANNs. Moreover, Liang et al. [44] exploited spike-compatible gradient to perform bit-flip attacks on SNN. Considering that SNNs are suitable for event-based tasks, various attacks on the event data of neuromorphic sensors are also explored [46, 49]. In these attacks, the STBP gradients are merged into sparse event data to construct attacks.

Although the performance of SNNs can be degraded by such a variety of attack schemes, SNNs are still considered to have additional robustness compared to ANNs [15]. This is because more encodings are supported in SNNs. Compared to direct coding, Poisson coding is believed to process more inherent robustness as it introduces a noisy discretization to SNNs [37, 63]. Even so, how strong an attack Poisson coding can withstand still remains unknown. Another key robustness component of SNNs is the leaky parameter [63]. The leaky parameter controls the forgetting of historical information in LIF neurons. El-Allami et al. [18] achieved improved robustness by only searching in the space of leaky parameters. The current understanding of the robustness of SNNs is mainly focused on rate coding, and LIF neurons are also thought to perform the same coding. Therefore, gaining new insights into the robustness of rate coding is crucial for the practical application of SNNs.

3. Preliminaries

3.1. Neuron Model for SNNs

In this paper, we consider the commonly used Leaky-Integrate-and-Fire (LIF) model and Integrate-and-Fire (IF) model [23, 31], the dynamics of membrane potential under firing threshold can be described by the following equations, respectively.

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -(u(t) - u_{rest}) + RI(t), \quad \textbf{(LIF)} \tag{1}$$

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = RI(t). \quad \textbf{(IF)} \tag{2}$$

Here τ_m and R denote the membrane time constant and the membrane resistance constant, respectively. $u(t)$ and $I(t)$ represent the membrane potential and input current at time t, respectively. Once the membrane potential reaches the firing threshold θ, the neuron will fire, and then the membrane potential will reset to the resting potential $u_{rest} < \theta$. Without loss of generality, to simulate the network of LIF neurons on a computer with a Von Neumann architecture, the above process can be simplified and discretized to another set of equations [4].

$$u_i(t) = \lambda u_i(t-1) + w_{ij}s_j(t) + b_i + \eta_i(t), \tag{3}$$

$$s_i(t) = \begin{cases} 1, & t = t_i^{(f)} \\ 0, & t \neq t_i^{(f)} \end{cases}, \tag{4}$$

where j and i indicate presynaptic and postsynaptic neurons, w_{ij} represents the connection strength between the two neurons, and b_i is an extra constant input current to neuron i. $s_i(t)$ describes whether neuron i fires at time-step t, and the trigger is membrane potential $u_i(t)$ reaches the firing threshold θ. We also mark the firing time-step of neuron i as $t_i^{(f)}$, $f = 1, 2, ...$ λ is the membrane leaky constant corresponding to the membrane time constant in Eq. (1). Note that if we explicitly λ to 1, this LIF neuron model will degenerate to the none-leaky IF model. Also, we add a reset term $\eta_i(t)$ on Eq. (3) to describe the neuron reset behavior. Despite the hard-reset function that resets the neuron membrane potential to resting potential (Eq. (5)), we also consider the soft-reset function (Eq. (6)) that directly subtracts the membrane potential by the threshold θ [25, 60].

$$\eta_i^{\text{hard}}(t) = \begin{cases} -\left(u_i(t_i^{(f)}) - u_{rest}\right) & , \ t = t_i^{(f)} \\ 0 & , \ t \neq t_i^{(f)} \end{cases}. \tag{5}$$

$$\eta_i^{\text{soft}}(t) = \begin{cases} -\theta & , \ t = t_i^{(f)} \\ 0 & , \ t \neq t_i^{(f)} \end{cases}, \tag{6}$$

3.2. Supervised Training of SNNs with STBP

Equation (3) implies that the LIF neuron has a function similar to that of an RNN. Thus, SNNs can be trained in the same way as RNNs with Back Propagation Through Time. STBP uses this idea to unroll the SNN over time-steps and accumulate gradient at each time-step [53, 72]. As illustrated in Fig. 1, the gradient of the loss function $\mathcal{L}$ with respect to the output spikes $s_j(t)$ is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial s_j(t)} = & \frac{\partial \mathcal{L}}{\partial s_i(t)} \frac{\partial s_i(t)}{\partial u_i(t)} \frac{\partial u_i(t)}{\partial s_j(t)} + \\ & \frac{\partial \mathcal{L}}{\partial s_j(t+1)} \frac{\partial s_j(t+1)}{\partial u_j(t+1)} \frac{\partial u_j(t+1)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial s_j(t)}. \end{aligned} \tag{7}$$

Note that as the derivative of spike with respect to the membrane potential $\frac{\partial s_i(t)}{\partial u_i(t)}$ is nondifferentiable, the surrogate gradient [53] is often used in backpropagation.

Since the STBP method enables supervised training of SNNs, this method can also generate gradient-based adversarial examples [62]. Therefore, we will use STBP as the baseline gradient obtain method.

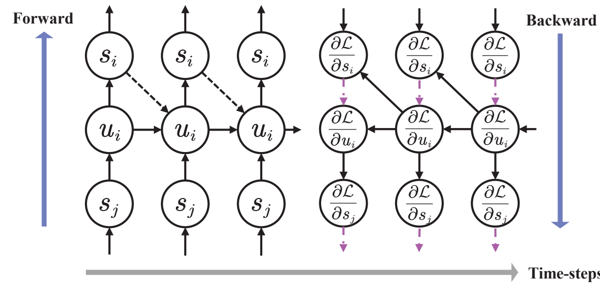


Figure 1. Forward pass and backward pass for STBP. The pink line shows that surrogate functions are applied in the non-differentiable process of neuron firing to obtain approximate gradients.

3.3. Adversarial Attacks

Adversarial attack is a method to generate imperceptible perturbations that can fool neural networks, which can be formulated as an optimization problem:

$$\arg\max_{\boldsymbol{\delta}} \mathcal{L}(f(\boldsymbol{x} + \boldsymbol{\delta}), y) \quad s.t. \ \|\boldsymbol{\delta}\|_p \leq \epsilon, \tag{8}$$

where $\mathcal{L}$ is the loss function, f is the network under attack, and $\boldsymbol{x}, y$ are the input images and output target of the given network, respectively. $\boldsymbol{\delta}$ is the adversarial perturbation we want to optimize. $\|\cdot\|_p$ is the L_p-norm, and parameter ϵ limits the strength of the perturbation to a level that is indistinguishable to the human eye. Here we consider two classic adversarial attack algorithms: Fast Gradient Sign Method (FGSM) [24] and Projected Gradient Descent (PGD) [38].

FGSM is a simple but effective attack method, which perturbs the data $\boldsymbol{x}$ along the sign of the gradient to increase the perturbed linear output, that is

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \ \text{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(f(\boldsymbol{x}, y))). \tag{9}$$

PGD is an iterative variant of FGSM. By iteratively optimizing the perturbation, PGD offers a more powerful attack [48]. The iteration can be summarized as:

$$\hat{\boldsymbol{x}}^k = \Pi_\epsilon \{\boldsymbol{x}^{k-1} + \alpha \ \text{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(f(\boldsymbol{x}^{k-1}, y)))\}, \tag{10}$$

where k is the number of the iteration step, and α is the step size of each iteration. Π_ϵ constrains the data in each iteration and projects it onto the space of the $\epsilon - l_p$ neighborhood of $\boldsymbol{x}$.

For all attack methods, we consider two different attack scenarios, white-box attack, and black-box attack. The white-box attack is the case that the hacker has complete access to the model topology, model parameters, and gradients, while the black-box attack is the case the hacker can only get the basic information of the model.

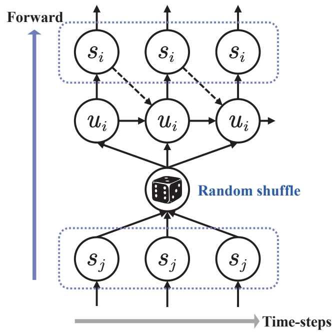


Figure 2. Experimental scheme of spike shuffle

4. Methods

In this section, we first show that the current high-performance SNNs for image classification tasks are encoded by firing rate and do not contain timing information, whether they are trained by STBP and ANN-SNN conversion. Based on this, we propose rate gradient approximation, a new attack method against SNNs, and apply it to SNNs composed of different types of neurons and input encodings. Finally, we propose the time-extended attack as an attack enhancement method.

4.1. Does well-trained SNNs contain timing information?

Here we design an experiment and test whether the well-trained SNNs are rate coded and whether they contain timing information. The definition of firing rate refers to the temporal average of the spikes. As illustrated in Eq. (11), the firing rate of one given neuron i is the total spike count in an interval of time-steps T divided by the duration T.

$$r_i = \frac{\sum_{t=1}^{T} s_i}{T}. \tag{11}$$

The detailed experimental scheme is shown in Fig. 2. We add a random number generator to shuffle each neuron's output spike firing order so that the spike trains will never contain temporal information. We apply this design to the pre-trained SNN models and test accuracy change before and after the spike shuffle.

The experiment is implemented on the CIFAR-10 dataset with VGG-11 network architecture. To test different SNNs, we choose combinations of different training methods, leakage parameters, and reset functions. We pre-train SNN for each combination and then substitute all the spiking neurons with shuffled neurons.

Tab. 1 reports the average performance of 10 runs, from which we find that the accuracy after spike shuffle is slightly different from the clean accuracy for all SNNs. For the model converted from ANNs, the model performance after the spike shuffle is slightly improved. The performance after spike shuffle is slightly reduced for other models obtained by supervised STBP training. In fact, even for models trained on the DVS dataset, the performance gap before and after shuffling is small, which proves that models trained on the DVS dataset contain very little timing information. Therefore, we assert that firing rates encode major information in these SNNs.

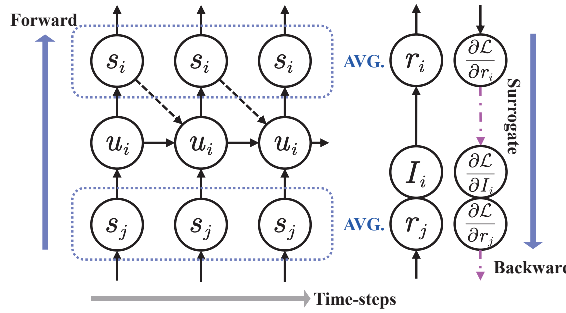


Figure 3. Forward pass and backward pass for RGA attack. The pink line shows that surrogate functions are applied to the approximation of the derivative for firing rate and input current.

Table 1. Performance before and after the spike shuffle

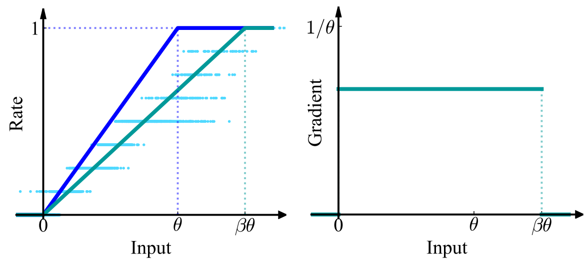
Dataset	Training Method	T	λ	Reset	Clean Acc.	Shuffled Acc.	Rate
CIFAR-10	ANNSNN	16	1.0	soft	93.25	93.358	✓
CIFAR-10	STBP	8	1.0	soft	92.75	92.086	✓
CIFAR-10	STBP	8	1.0	hard	93.06	92.214	✓
CIFAR-10	STBP	8	0.9	hard	93.03	92.545	✓
CIFAR-10	STBP	8	0.5	hard	91.48	91.225	✓
CIFAR10-DVS	STBP	10	0.9	hard	77.00	75.400	✓

4.2. Rate Gradient Approximation Attack

This section proposes the Rate Gradient Approximation (RGA) attack for spike count rate coding SNNs. In the above section, we have shown that the well-trained SNNs are all rate-encoded at each layer. Thus, we can approximate the backward pass of SNNs using only the average firing rate over time-steps to generate effective gradients. Specifically, we introduce an intermediate variable I_i to denote the average input current of neuron i in Eq. (3), which is defined by:

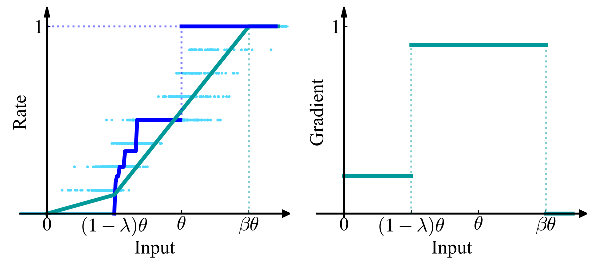
$$I_i = \sum_{t=1}^{T} w_{ij} s_j(t) + b_i \tag{12}$$

Note that I_i also represents the weighted firing rate from the previous layer. Fig. 3 illustrates how the gradient propa-



(a) IF R-I curve (b) Derivative of IF R-I curve

Figure 4. Surrogate function and gradient for IF neuron



(a) LIF R-I curve (b) Derivative of LIF R-I curve

Figure 5. Surrogate function and gradient for LIF neuron

gates between two adjacent neurons i and j with the defined weighted firing rate. The gradient propagation starts from the firing rate r_i of the i-th neuron, passes to the input current I_i, and finally arrives at the firing rate r_j of the j-th neuron, that is

$$\frac{\partial \mathcal{L}}{\partial r_j} = \frac{\partial \mathcal{L}}{\partial r_i}\frac{\partial r_i}{\partial I_i}\frac{\partial I_i}{\partial r_j}. \tag{13}$$

This result can be generalized to the network with the chain rule. If we use vectors $\boldsymbol{r}^{\boldsymbol{l}}$ and $\boldsymbol{I}^l$ to denote the firing rates and average input currents of all neurons in layer l, respectively, and use vector $\boldsymbol{r}^0$ to denote the input image. The gradient propagates from the loss function to the input image is formulated as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^0} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{r}^L}\left(\prod_{l=1}^{L}\frac{\partial \boldsymbol{r}^l}{\partial \boldsymbol{I}^l}\frac{\partial \boldsymbol{I}^l}{\partial \boldsymbol{r}^{l-1}}\right). \tag{14}$$

4.3. RGA Attack with Surrogate Gradient

According to Eq. (11) and (12), one can compute the gradient $\frac{\partial I_i}{\partial r_j}$, the only part we cannot directly calculate the gradient in Eq. (13) is $\frac{\partial r_i}{\partial I_i}$. To get a proper approximation for $\frac{\partial r_i}{\partial I_i}$, we need to use a differentiable surrogate function $f(\cdot)$ to approximate the relationship between output firing rate r_i and input current I_i. Here, we propose to use the static R-I curve, which refers to the relationship between the input current and output firing rate when the input is constant, as the approximation function. We discuss the cases of IF neurons and LIF neurons separately.

RGA Attack the Integrate-and-Fire Neuron

For IF neurons, the static R-I curve is the clipped ReLU function [7, 14], which is presented in Fig. 4a by the blue line. We also sampled from a pre-trained VGG-11 network to obtain the actual distribution of average input current and average output firing rate (more details are in the Appendix), which is represented by the light blue scatter in the figure. We find that if we simply set the static RI curve as the surrogate gradient, many data points would be wrongly assigned to zero derivatives. To avoid the derivatives being all zero in the interval $[\theta, +\infty]$, we propose a modified R-I curve function as the final surrogate function (green line in Fig. 4a). We introduce a relaxation parameter $\beta \in [1, +\infty]$ to keep the derivatives to be none-zero between $[0, \beta\theta]$. Then its corresponding derivative (surrogate gradient) is:

$$\frac{\partial r_i}{\partial I_i} = \begin{cases} 1/(\beta\theta), & 0 \leqslant I_i \leqslant \beta\theta \\ 0, & I_i > \beta\theta \text{ or } I_i < 0 \end{cases}. \tag{15}$$

With the formula for backpropagation and the derivatives of intermediate nodes, we can calculate the derivatives that are ultimately passed to the input image.

RGA Attack the Leaky-Integrate-and-Fire Neuron

Similar to the RGA Attack for IF neurons, the surrogate function for LIF neurons is inspired by the static R-I curve. Following the work of [23, 30], we derive the R-I curve function for the LIF neurons. Supposing that a LIF neuron receives a constant input I_i and $u_{rest} = 0$, the LIF neuron (Eq. (3)) can be simplified to:

$$u_i(t) = \lambda u_i(t-1) + I_i. \tag{16}$$

We then iterate this recursive formula to get the neuron membrane potential as a function of time.

$$u_i(t) = \frac{I_i}{\lambda - 1}\lambda^t - \frac{I_i}{\lambda - 1} \tag{17}$$

From the formula above, we can calculate the time required for the neuron to accumulate from 0 to θ, and finally obtain the firing rate by inverse the calculated time interval.

$$r_i = \left\lceil \log_\lambda \frac{I_i + \theta(\lambda - 1)}{I_i} \right\rceil^{-1} \tag{18}$$

As shown in Fig. 5a, the light blue scatter represents the actual distribution of average input current and average output firing rate from a pre-trained VGG-11 network with LIF neurons. Although the static RI curve matches the actual situation, the static RI curve (blue curve) contains the non-derivable equation. The derivative is always zero when $x < (1-\lambda)\theta$ or $x > \theta$. Meanwhile, as $\lim_{x \to (1-\lambda)\theta^+}$, the derivative of the function approaches infinity. This will lead to gradient vanishing and exploding problems in backpropagation. Therefore, it is impractical to calculate the derivatives directly from the static R-I curve.

According to the characteristics of this static R-I curve, we segment it at the position of $(1-\lambda)\theta$ and then separately perform piecewise linear interpolation at $x \in [0, (1-\lambda)\theta]$ and $x \in [(1-\lambda)\theta, \beta\theta]$, where β is a relaxation parameter to control the end point of interpolation. Thus, we obtain two straight lines (green line in Fig. 5a). In order to avoid zero derivatives over the interval $[0, (1-\lambda)\theta]$, we also add a hyperparameter γ which is the derivative of the first half of the linear function. In conclusion, the first part of the linear interpolation start at data point $(0, 0)$ and end at $((1-\lambda)\theta, (1-\lambda)\theta\gamma)$. The second part of the interpolation start at $((1-\lambda)\theta, (1-\lambda)\theta\gamma)$ and end at $(\beta\theta, 1)$.

Therefore, the final surrogate gradient can be written as

$$\frac{\partial r_i}{\partial I_i} = \begin{cases} \gamma, & 0 \leqslant I_i \leqslant (1-\lambda)\theta \\ \dfrac{1-\gamma\theta+\gamma\theta\lambda}{(\beta+\lambda-1)\theta}, & (1-\lambda)\theta < I_i \leqslant \beta\theta \\ 0, & I_i > \beta\theta \ \text{ or } \ I_i < 0 \end{cases}. \quad (19)$$

The derivative of this surrogate gradient function is γ in the first interval and $(1-\gamma\theta+\gamma\theta\lambda)/((\beta+\lambda-1)\theta)$ in the second interval. Note that when we set the leaky parameter $\lambda = 1$, this function degenerates into the same form as the surrogate function for the IF neuron (Eq. (15)). Thus, this function can apply to both types of neurons.

The hyperparameter β and γ are mainly smoothing terms to prevent the gradient from disappearing. We set β to 2 and γ to 0.2 in the rest of this paper. More ablation experiments of the hyperparameters are provided in the Appendix.

4.4. RGA Attack the Poisson Encoding

Our proposed method can generalize to SNNs that receive spike inputs. For Poisson input SNN, we can regard it as a combined structure of a Poisson encoder and an end-to-end SNN receives spike input. Therefore, we also need to attack the Poisson encoder rather than only perturbing the network. We can consider the Poisson encoder as a random transformation and use a straight through estimator [2] to attack this random transformation [1]. The gradient can be written as:

$$\frac{\partial \text{Poisson}(x)}{\partial x} \approx \frac{\partial \mathbb{E}_x\left(\text{Poisson}(x)\right)}{\partial x} = 1. \quad (20)$$

The Adversarial example generation methodology proposed by Sharmin et al. [63] also includes the attack on Poisson encoded attack. Note that although our attack on the Poisson encoder is implemented differently from theirs, it is mathematically equivalent, and the generated adversarial examples are the same.

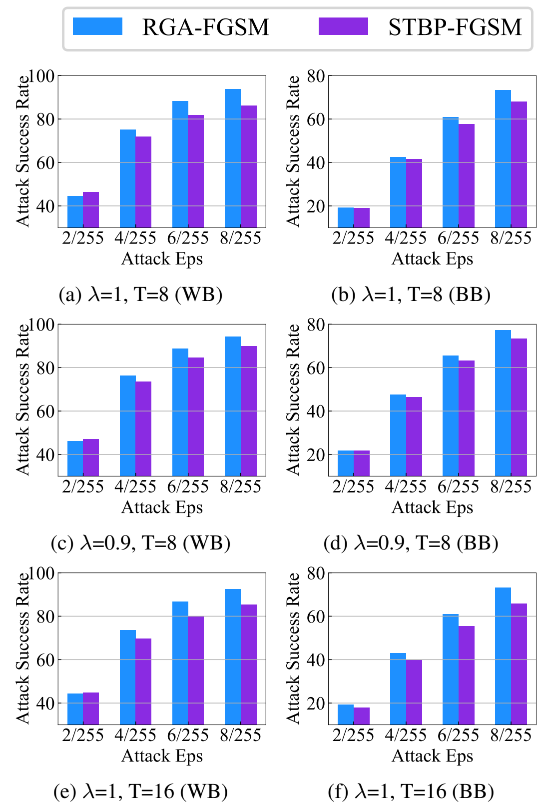


Figure 6. The attack success rate change with respect to the attack strength for VGG-11 model on the CIFAR-10 dataset. Here WB stands for white box attack while BB stands for black box attack. The blue and purple bar display the attack success rates (%) for RGA-FGSM and STBP-FGSM, respectively.

4.5. Time Extended RGA Attack

Here we introduce an SNN-specific attack enhancement method. When generating attack samples, we expect to generate more effective adversarial samples by increasing the inference time of SNNs. We call this method as Time Extended attack. When we apply this time-extended enhancement with RGA attack, the effects of randomness can be reduced by increasing the simulation time of attacking, resulting in a more accurate estimation of the firing rate. Therefore, the time-extended enhancement is generally more suitable for RGA-based attacks and is more effective for attacking against SNNs with Poisson inputs.

5. Experiments

In this section, we conduct various experiments to evaluate the effectiveness of the proposed methods [32, 55]. We first test our attack method on the CIFAR-10, CIFAR-100 dataset [36] and CIFAR10-DVS dataset [42] with the VGG-11 [66] and ResNet-17 architecture [80]. We implement both non-iterative attack FGSM and iterative attack PGD

Architecture	Dataset	Input	T	λ	TE	Attack	Clean Acc.	White Box Attack		Black Box Attack	
								ASR. (STBP)	ASR. (RGA)	ASR. (STBP)	ASR. (RGA)
VGG-11	CIFAR-10	Direct	8	1.0	-	FGSM	93.06	86.2777	**93.7352**	68.0314	**73.2646**
VGG-11	CIFAR-10	Direct	8	1.0	2×	FGSM	93.06	86.0735	**94.7346**	64.8399	**73.6192**
VGG-11	CIFAR-10	Direct	8	1.0	-	PGD	93.06	99.4949	**99.8281**	86.4604	**87.1266**
VGG-11	CIFAR-10	Direct	16	1.0	-	FGSM	93.03	85.3273	**92.4218**	65.7960	**73.1269**
VGG-11	CIFAR-10	Direct	16	1.0	-	PGD	93.03	99.3658	**99.8388**	85.5853	**87.4234**
VGG-11	CIFAR-10	Poisson	16	1.0	-	FGSM	86.72	54.9798	**58.0328**	40.8673	**44.2259**
VGG-11	CIFAR-10	Poisson	16	1.0	2×	FGSM	86.72	56.8106	**60.5296**	42.8440	**46.9085**
VGG-11	CIFAR-10	Poisson	16	1.0	-	PGD	86.72	51.9022	**57.1412**	37.0917	**41.1887**
VGG-11	CIFAR-10	Direct	8	0.5	-	FGSM	91.48	91.7140	**93.6270**	77.7656	**79.6458**
VGG-11	CIFAR-10	Direct	8	0.5	-	PGD	91.48	**99.8251**	99.7704	**93.6817**	93.0367
VGG-11	CIFAR-10	Direct	8	0.9	-	FGSM	93.03	89.9065	**94.4104**	73.4494	**77.2761**
VGG-11	CIFAR-10	Direct	8	0.9	-	PGD	93.03	99.7313	**99.8280**	**91.7661**	91.3899
ResNet-17	CIFAR-10	Direct	8	0.9	-	FGSM	93.04	84.2433	**92.9278**	67.1109	**80.1053**
ResNet-17	CIFAR-10	Direct	8	0.9	-	PGD	93.04	99.9248	**100.000**	92.0034	**97.5172**
VGG-11	CIFAR-100	Direct	8	0.9	-	FGSM	73.28	92.8766	**94.7189**	80.8952	**84.2658**
VGG-11	CIFAR-100	Direct	8	0.9	-	PGD	73.28	99.7544	**99.8499**	**92.2353**	92.0579
ResNet-17	CIFAR-100	Direct	8	0.9	-	FGSM	72.05	85.6627	**92.0611**	74.2956	**81.1936**
ResNet-17	CIFAR-100	Direct	8	0.9	-	PGD	72.05	99.5836	**99.8890**	87.6336	**95.2949**
VGG-11	CIFAR10-DVS	Frame	10	0.9	-	FGSM	77.00	59.5084	**59.5607**	**48.4967**	47.9275

Table 2. Results of RGA based attack and STBP based attack on different type of SNNs. The seven columns on the left describes the parameter settings of the attack object, including architecture, datasets, input coding, simulation time-steps, leaky parameters, time extant (TE), and attack method. ASR. is short for attack success rate. For clarity, the better of the two attack results is bolded.

and combine them with the proposed RGA attack and compare the results with STBP-based attacks. Note that when attacking DVS-related models, we directly generate and add adversarial examples on the preprocessed frames.

As for training methods, since the ANN-SNN conversion method is proved insecure [63], we only consider SNNs obtained by supervised STBP training. We used the attack success rate as our metric for each attack, which is the proportion of samples that fool the network into misclassification. The details of the training configurations for pre-trained networks are provided in the Appendix.

5.1. Effectiveness of the RGA Attack

Here we test the effectiveness of the proposed RGA attack. We choose the STBP attack as a baseline and compare the attack success rate. To compare the effectiveness of the attack on various neuron settings, we select SNNs composed of LIF neurons with a leaky parameter of 0.9 and none-leaky IF neurons with a leaky parameter of 1.0. Also, the inference time-steps vary from 8 to 16.

As shown in Fig. 6, we find that the attack success rate of our RGA attack method surpasses the previously commonly used STBP attack in most cases. When the attack strength is small, the attack success rate of the RGA attack and STBP attack is not much different. However, when the attack strength gradually increases to 8/255, the attack success rate of the RGA attack will be much higher than that obtained by the STBP attack. These results demonstrate that the RGA attack is more effective than the STBP attack for different SNN settings.

5.2. Effectiveness of Time Extended Attack

To demonstrate the effectiveness of the time-extended attack, we pre-train two SNNs with different neurons and input encoding. We then double and triple the inference time-steps to attack these SNNs while keeping the origin inference time-steps for evaluation. The SNN configuration and the final results are shown in Fig. 7. The combination of the time-extended enhancement and RGA-based attack can achieve better attack performance since the attack effectiveness increases as the simulation time-step increases. However, applying the time-extended enhancement attack on the STBP-based attack cannot improve the attack. Meanwhile, we also find that the time-extended enhancement method is more effective for the Poisson input model.

5.3. Generalizability of the RGA Attacks

To discuss the generalizability of RGA attacks across different SNNs, we conduct further experiments across various SNNs. Tab. 2 reports the detailed comparison of RGA attack and STBP attack over various SNN neurons, architecture, and dataset settings. In this experiment, we set the attack strength ϵ to 8/255 for CIFAR-10/100 dataset and 0.02 for the CIFAR10-DVS dataset. Also, the step size and

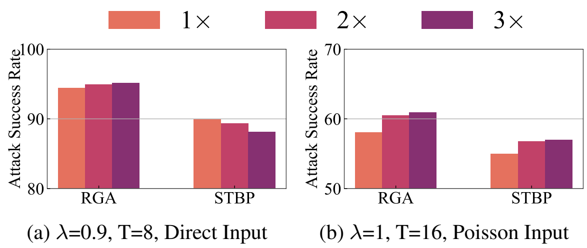


Figure 7. White box attack results of the time-extended attack with different settings. The 1× represents the baseline attack, while 2× and 3× represent double or triple the simulation time when attacked. The model under attack is VGG-11 composed of LIF neurons on the CIFAR-10 dataset.

the number of steps for the PGD attack are set to 2/255 and 5, respectively.

From Tab. 2, one can find that all RGA-based attacks outperform STBP-based attacks when conducting single-step attacks. In attacks cases without time-extended enhancement, the largest performance gap exceeds 8.6% and 13% in ResNet-17 for white box and black box attacks, respectively. When conducting multi-step attacks, the proposed RGA-based attack achieves better performance than the STBP-based attack in most cases and gets comparable performance for the rest of the cases. In attack cases with time-extended enhancement, the performance of the RGA attack not only exceeds the STBP attack, but also exceeds the performance of RGA attack without time-extended enhancement. Even in attacks on the CIFAR10-DVS dataset, the RGA-based attack gets comparable performance as the STBP-based attack.

In conclusion, the proposed RGA attack can produce stronger adversarial examples in most cases. The RGA attack is also insensitive to neuron hyperparameters.Currently, high-performance SNNs are mainly applied in tasks related to static images or DVS datasets. We have proved through experiments that those SNNs contain very limited temporal information and they are vulnerable to the designed RGA-based attack. This demonstrates the importance and generalizability of the RGA-based attack method.

5.4. LIF Neuron is not that Robust

Previous research suggested that the VGG5 SNNs trained with LIF neurons are more robust than the ones trained with IF neurons [63]. Here, we test whether this conclusion can be generalized to deeper networks with RGA-based attacks. We train multiple networks with the same architecture, including one quantized ANN and six SNNs with leaky parameters ranging from 0.5 to 1. We select VGG-11 as the model architecture and CIFAR-10 as the evaluation dataset. Then, we apply the white box FGSM attack to the quantized ANN and both RGA-FGSM and STBP-FGSM attacks to the six SNNs.

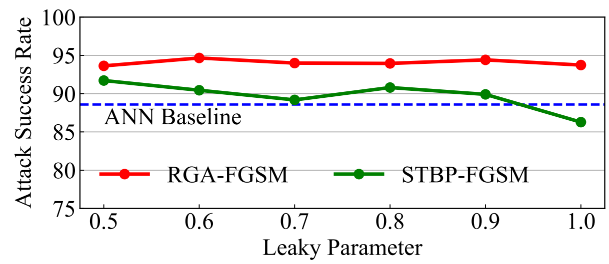


Figure 8. The white box attack success rate changes with respect to the leaky parameter of the spiking neuron. The red line represents the RGA attack, while the green line represents the STBP attack. The blue dashed line illustrates the attack success rate of a quantized ANN under the FGSM attack. The selected leaky parameters range from 0.5 to 1.0. This experiment is conducted on the CIFAR-10 dataset with VGG-11 architecture.

Fig. 8 demonstrates the results. The blue dashed line represents the attack success rate of ANN being attacked by FGSM. The red and green lines represent the attack success rate of the RGA-FGSM and STBP-FGSM, respectively. According to this figure, one can find that compared to the STBP attack (green line), the RGA attack (red line) is more stable. Meanwhile, the red line is always above the green line, indicating that the RGA attack can not only generate stronger adversarial examples, but is also less sensitive to neuron leakage parameters.

In addition, the performance of all types of SNNs under STBP-based attacks fluctuates around the baseline, while the RGA attack success rate is always higher than the baseline. Therefore, we found that SNN composed of LIF neurons with different leakage parameters has no advantage over ANNs in terms of adversarial robustness.

6. Conclusion

In this paper, we propose a new attack method for SNNs that outperforms previous methods. Our approach can serve as a benchmark for future research on SNN adversarial robustness. In addition, considering the strength of this attack method and the lower time cost compared to the STBP attack, we also look forward to the research on adversarial training based on this attack method. Our findings show that the rate-coded SNN composed of LIF neurons is not secure against stronger adversarial attacks, highlighting the need for exploring training methods for SNNs utilizing complex neurons and other neuronal codings.

7. Acknowledgement

This work is funded by STI2030Project (2021ZD0200300), and the National Natural Science Foundation of China (621 76003, 62088102).

References

[1] 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*, 2018. 6

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 6

[3] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *European Symposium on Artificial Neural Networks*, 2000. 2

[4] Romain Brette, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, James M Bower, Markus Diesmann, Abigail Morrison, Philip H Goodman, Frederick C Harris, et al. Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, 2007. 3

[5] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 2

[6] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022. 2

[7] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 2015. 1, 2, 5

[8] Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State transition of dendritic spines improves learning of sparse spiking neural networks. In *International Conference on Machine Learning*, 2022. 1

[9] Xiang Cheng, Yunzhe Hao, Jiaming Xu, and Bo Xu. LISNN: Improving spiking neural networks with lateral interactions for robust object recognition. In *International Joint Conferences on Artificial Intelligence*, 2020. 1

[10] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018. 1

[11] Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 2019. 1

[12] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021. 2

[13] 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

[14] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, 2015. 2, 5

[15] Jianhao Ding, Tong Bu, Zhaofei Yu, Tiejun Huang, and Jian K Liu. Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training. In *Advances in Neural Information Processing Systems*, 2023. 2

[16] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *International Joint Conference on Artificial Intelligence*, 2021. 2

[17] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In *Advances in Neural Information Processing Systems*, 2022. 2

[18] Rida El-Allami, Alberto Marchisio, Muhammad Shafique, and Ihsen Alouani. Securing deep spiking neural networks against adversarial attacks through inherent structural parameters. In *Design, Automation & Test in Europe Conference & Exhibition*, 2021. 2

[19] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 2016. 2

[20] Biao Fang, Yuhao Zhang, Rui Yan, and Huajin Tang. Spike trains encoding optimization for spiking neural networks implementation in fpga. In *International Conference on Advanced Computational Intelligence*, 2020. 1

[21] 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*, 2021. 2

[22] 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 *International Conference on Computer Vision*, 2021. 2

[23] 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. 1, 2, 5

[24] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015. 1, 3

[25] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *International Conference on Computer Vision*, 2020. 2, 3

[26] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. *arXiv preprint arXiv:2302.02091*, 2023. 2

[27] Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. In *International Conference on Learning Representations*, 2023. 2

[28] Nguyen-Dong Ho and Ik-Joon Chang. TCL: an ANN-to-SNN conversion with trainable clipping layers. In *Design Automation Conference*, 2021. 2

[29] Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2018. 2

[30] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with LIF neurons. *arXiv preprint arXiv:1510.08829*, 2015. 5

[31] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 2004. 2

[32] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020. 6

[33] Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation-and timing-based learning rules for spiking neural networks. *Advances in Neural Information Processing Systems*, 2020. 2

[34] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-YOLO: Spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2

[35] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 2020. 2

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[37] 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 *International Conference on Computer Vision*, 2021. 1, 2

[38] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations, Workshop Track Proceedings*, 2017. 3

[39] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 2020. 2

[40] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 2016. 2

[41] Chen Li, Lei Ma, and Steve B Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 2022. 2

[42] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in Neuroscience*, 2017. 6

[43] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, 2021. 2

[44] 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*, 2021. 2

[45] Ling Liang, Kaidi Xu, Xing Hu, Lei Deng, and Yuan Xie. Toward robust spiking neural network against adversarial perturbation. *arXiv preprint arXiv:2205.01625*, 2022. 1

[46] Xuanwei Lin, Chen Dong, Ximeng Liu, and Yuanyuan Zhang. SPA: An efficient adversarial attack on spiking neural networks using spike probabilistic. In *IEEE International Symposium on Cluster, Cloud and Internet Computing*, 2022. 2

[47] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 1997. 1

[48] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 3

[49] Alberto Marchisio, Giacomo Pira, Maurizio Martina, Guido Masera, and Muhammad Shafique. DVS-Attacks: Adversarial attacks on dynamic vision sensors for spiking neural networks. In *International Joint Conference on Neural Networks*, 2021. 2

[50] Qingyan Meng, Shen Yan, Mingqing Xiao, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training much deeper spiking neural networks with a small number of time-steps. *Neural Networks*, 2022. 2

[51] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. 2

[52] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010. 1

[53] 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*, 2019. 2, 3

[54] Nicolas Perez Nieves and Dan F. M. Goodman. Sparse spiking gradient descent. In *Advances in Neural Information Processing Systems*, 2021. 1

[55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 2019. 6

[56] 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*, 2019. 1

[57] Nitin Rathi and Kaushik Roy. DIET-SNN: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 1

[58] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*, 2020. 1

[59] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 2019. 1

[60] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 2017. 3

[61] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 2019. 2

[62] 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 *International Joint Conference on Neural Networks*, 2019. 1, 2, 3

[63] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and nonlinear activations. In *European Conference on Computer Vision*, 2020. 1, 2, 6, 7, 8

[64] Jiangrong Shen, Qi Xu, Jian K. Liu, Yueming Wang, Gang Pan, and Huajin Tang. ESL-SNNs: An evolutionary structure learning strategy for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. 1

[65] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in Neural Information Processing Systems*, 2018. 2

[66] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[67] Kenneth Michael Stewart and Emre Neftci. Meta-learning spiking neural networks with surrogate gradient descent. *Neuromorphic Computing and Engineering*, 2022. 2

[68] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 1

[69] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 2019. 1

[70] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 1990. 2

[71] Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

[72] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 2018. 1, 2, 3

[73] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. 2

[74] Zhanglu Yan, Jun Zhou, and Weng-Fai Wong. Near lossless transfer learning for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2

[75] Friedemann Zenke, Sander M Bohté, Claudia Clopath, Iulia M Comşa, Julian Göltz, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O Neftci, Mihai A Petrovici, et al. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 2021. 1

[76] Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 2021. 1

[77] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 2021. 2

[78] Malu Zhang, Jiadong Wang, Jibin Wu, Ammar Belatreche, Burin Amornpaisannon, Zhixuan Zhang, Venkata Pavan Kumar Miriyala, Hong Qu, Yansong Chua, Trevor E Carlson, et al. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 2

[79] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 2020. 2

[80] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2, 6

[81] Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. In *Advances in Neural Information Processing Systems*, 2022. 2