Exploring Adversarial Attack in Spiking Neural Networks With Spike-Compatible Gradient

Ling Liang, Xing Hu, *Member, IEEE*, Lei Deng, *Member, IEEE*, Yujie Wu, Guoqi Li, *Member, IEEE*, Yufei Ding, *Associate Member, IEEE*, Peng Li, *Fellow, IEEE*, and Yuan Xie, *Fellow, IEEE*

***Abstract*— Spiking neural network (SNN) is broadly deployed in neuromorphic devices to emulate brain function. In this context, SNN security becomes important while lacking in-depth investigation. To this end, we target the adversarial attack against SNNs and identify several challenges distinct from the artificial neural network (ANN) attack: 1) current adversarial attack is mainly based on gradient information that presents in a spatiotemporal pattern in SNNs, hard to obtain with conventional backpropagation algorithms; 2) the continuous gradient of the input is incompatible with the binary spiking input during gradient accumulation, hindering the generation of spike-based adversarial examples; and 3) the input gradient can be all-zeros (i.e., vanishing) sometimes due to the zero-dominant derivative of the firing function. Recently, backpropagation through time (BPTT)-inspired learning algorithms are widely introduced into SNNs to improve the performance, which brings the possibility to attack the models accurately given spatiotemporal gradient maps. We propose two approaches to address the above challenges of gradient-input incompatibility and gradient vanishing. Specifically, we design a gradient-to-spike (G2S) converter to convert continuous gradients to ternary ones compatible with spike inputs. Then, we design a restricted spike flipper (RSF) to construct ternary gradients that can randomly flip the spike inputs with a controllable turnover rate, when meeting all-zero gradients. Putting these methods together, we build an adversarial attack methodology for SNNs. Moreover, we analyze the influence of the training loss function and the firing threshold of the penultimate layer on the attack effectiveness. Extensive experiments are conducted to validate our solution. Besides the quantitative analysis of the influence factors, we also compare SNNs and ANNs against adversarial attacks under different attack methods. This work can help reveal what happens in SNN attacks and might stimulate more research on the security of SNN models and neuromorphic devices.**

Manuscript received December 19, 2019; revised August 30, 2020, January 9, 2021, and June 10, 2021; accepted August 11, 2021. *(Ling Liang and Xing Hu contributed equally to this work.) (Corresponding author: Lei Deng.)*

Ling Liang, Peng Li, and Yuan Xie are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: lingliang@ucsb.edu; lip@ucsb.edu; yuanxie@ucsb.edu).

Xing Hu is with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: huxing@ict.ac.cn).

Lei Deng, Yujie Wu, and Guoqi Li are with the Department of Precision Instrument, Center for Brain Inspired Computing Research, Tsinghua University, Beijing 100084, China (e-mail: leideng@mail.tsinghua.edu.cn; wu-yj16@tsinghua.org.cn; liguoqi@mail.tsinghua.edu.cn).

Yufei Ding is with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: yufeiding@cs.ucsb.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3106961.

Digital Object Identifier 10.1109/TNNLS.2021.3106961

***Index Terms*— Adversarial attack, backpropagation through time (BPTT), neuromorphic computing, spike-compatible gradient, spiking neural networks (SNNs).**

I. Introduction

SPIKING neural networks (SNNs) [1] closely mimic the behaviors of neural circuits via spatiotemporal neuronal dynamics and event-drive activities. They have shown promising ability in processing dynamic and noisy information with high efficiency [2], [3] and have been applied in a broad spectrum of tasks such as optical flow estimation [4], spike pattern recognition [5], SLAM [6], probabilistic inference [3], heuristically solving NP-hard problem [7], quickly solving optimization problem [8], sparse representation [9], robotics [10], and so forth. Besides the algorithm research, SNNs are widely deployed in neuromorphic devices for low-power brain-inspired computing [8], [11]–[13].

With more attention on SNNs, the security problem becomes quite important. Here we focus on adversarial attack [14], one of the most popular threat models for neural network security. In an adversarial attack, the attacker introduces imperceptible malicious perturbation into the input data to misleading the model's classification result. Although the adversarial attack is a very hot topic in artificial neural networks (ANNs), it is still in its infant stage in the SNN domain. Several related studies targeting this topic are based on the gradient-free attack methods (e.g., trial-and-error input perturbation [15], [16]) or the spatial-gradient-based SNN/ANN model conversion methods [17]. Former methods perturb the input in a trial-and-error manner by simply monitoring the output change without calculating the gradient; latter methods inherit adversarial examples generated by the ANN counterpart of the SNN model. The computational complexity of the trial-and-error input perturbation methods is quite high due to the large search space without the guidance of supervised gradients. Regarding the SNN/ANN model conversion methods, using a different model to find gradients and the missing temporal components will compromise the attack effectiveness. To address these issues, we propose spatiotemporal-gradient-based attacks for SNNs with both high attack efficiency and effectiveness.

We identify several challenges in attacking an SNN model using the gradient-based methodology. First, the input gradient in SNNs presents as a spatiotemporal pattern that is hard to obtain with traditional learning algorithms

like the gradient-free unsupervised learning [18], [19] and spatial-gradient-based ANN-to-SNN-conversion learning [20]. Second, the gradients are continuous values, incompatible with the binary spiking inputs. This data format incompatibility impedes the generation of spike-based adversarial examples via gradient accumulation. At last, there is severe gradient vanishing when the gradient crosses the step firing function with a zero-dominant derivative, which will interrupt the update of adversarial examples.

Recently, the backpropagation through time (BPTT) inspired supervised learning algorithms [2], [5], [21]–[25] are widely introduced into SNNs for a performance boost, which enables the direct acquisition of gradient information in both spatial and temporal dimensions. This brings the opportunity to realize an accurate SNN attack based on spatiotemporal input gradients directly calculated in SNNs without model conversion. Then, to address the mentioned issues of gradient-input incompatibility and gradient vanishing, we propose two approaches. We design a gradient-to-spike (G2S) converter to convert continuous gradients to ternary ones that are compatible with spike inputs. Then we design a restricted spike flipper (RSF) to construct ternary gradients that can randomly flip the spike inputs when facing all-zero gradient maps, where the turnover rate of inputs is controllable. Under this attack methodology for both untargeted and targeted attacks, we analyze the impact of two important factors on the attack effectiveness: the format of the training loss function and the firing threshold. We find a "trap" region for the model trained by cross-entropy (CE) loss, which makes it harder to attack when compared to the one trained by mean square error (MSE) loss. Fortunately, the "trap" region can be escaped by adjusting the firing threshold of the penultimate layer. We extensively validate our SNN attack methodology on both neuromorphic datasets (e.g., N-MNIST [26] and CIFAR10-DVS [27]) and image datasets (e.g., MNIST [28] and CIFAR10 [29]). We summarize our contributions as follows.

1) We identify the challenges of adversarial attacks against SNN models. Then, we realize an effective and efficient SNN attack via a spike-compatible spatiotemporal gradient. Specifically, we design a G2S converter to address the gradient-input incompatibility problem and an RSF to address the gradient vanishing problem.
2) We explore the influence of the training loss function and the firing threshold of the penultimate layer and propose threshold tuning to improve the attack effectiveness.
3) Extensive experiments are conducted on both neuromorphic and image datasets. [1] In addition, we design experiments to compare SNNs and ANNs against adversarial attacks under different attack methods.

The rest of this article is organized as follows. Section II provides some preliminaries of SNNs and adversarial attacks. Section III discusses the challenges in the SNN attack and our differences with prior work. Sections IV and V illustrate our attack methodology and the two factors that can affect the attack effectiveness; The experimental setup and the result analyses are shown in Section VI. Finally, Section VII concludes and discusses this article.

[1] https://github.com/liangling76/snn_attack

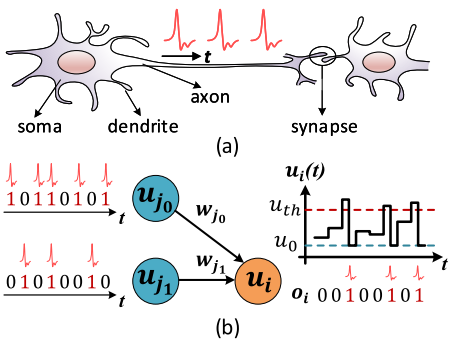


Fig. 1. Introduction of SNNs. (a) Neuronal components. (b) Computing model.

II. Preliminaries

A. SNNs

In SNNs, a neuron is the basic structural unit as shown in Fig. 1, which is comprised of the dendrite, soma, and axon. Many neurons connected by weighted synapses form an SNN, in which the binary spike events carry information for interneuron communication. Dendrite integrates the weighted presynaptic inputs, and soma consequently updates the membrane potential and determines whether to fire a spike or not. When the membrane potential crosses a threshold, a spike will be fired and sent to postneurons through the axon.

The leaky integrate-and-fire (LIF) model [30] is the most widely adopted SNN model. The behavior of each LIF neuron can be briefly expressed as

$$\begin{cases} \tau \dfrac{du(t)}{dt} = -u(t) + \sum_j w_j o_j(t) \\ o(t) \mathrel{+}= \mathrm{Dirac}\left(t - t'\right) \ \& \ u\left(t'\right) = u_0, \ \text{if } u\left(t'\right) \geq u_{\text{th}} \end{cases} \tag{1}$$

where t denotes the timestep, τ is a time constant, and u and o represent the membrane potential and spike, respectively. w_j is the synaptic weight between the jth preneuron and the current neuron, and o_j is the output spike of the jth preneuron. u_{th} is the firing threshold and u_0 is the reset potential used after firing a spike. Note that in the continuous time domain, in order to change the membrane potential by integrating spikes, each spike is modeled as a Dirac delta function whose only infinite peak lies at the firing time.

The network structure of feedforward SNNs can be similar to that of ANNs, including convolutional (Conv) layer, pooling layer, and fully connected (FC) layer. The network inputs can be spike events captured by dynamic vision sensors [31] (i.e., neuromorphic datasets) or converted from normal image datasets through Bernoulli sampling [2]. The classification is conducted based on the spikes of the output layer.

B. Gradient-Based Adversarial Attack

We take the gradient-based adversarial attack in ANNs as an illustrative example. The neural network is actually a map from inputs to outputs, i.e., $y = f(x)$, where x and y denote inputs and outputs, respectively, and $f : R^m \to R^n$ is the map

function. Usually, the inputs are static images in convolutional neural networks. In an adversarial attack, the attacker attempts to manipulate the victim model to produce incorrect outputs by adding imperceptible perturbations δ in the input images. We define $x' = x + \delta$ as an adversarial example. The perturbation is constrained by $\|\delta\|_p = \|x'-x\|_p \leq \epsilon$, where $\|\cdot\|_p$ denotes the p-norm and ϵ reflects the maximum tolerable perturbation.

Generally, the adversarial attack can be categorized into untargeted attack and targeted attack according to the different attack goals. Untargeted attack fools the model to classify the adversarial example into any other classes except for the original correct one, which can be illustrated as $f(x+\delta) \neq f(x)$. In contrast, for targeted attack, the adversarial example must be classified in to a specified class, i.e., $f(x+\delta) = y_{\text{target}}$. With these preliminary knowledge, the adversarial attack can be formulated as an optimization problem as below to search the smallest perturbation

$$\begin{cases} \arg\min_{\delta} \|\delta\|_p, & \text{s.t. } f(x+\delta) \neq f(x), \quad \text{if untargeted} \\ \arg\min_{\delta} \|\delta\|_p, & \text{s.t. } f(x+\delta) = y_{\text{target}}, \quad \text{if targeted.} \end{cases} \tag{2}$$

There are several widely adopted adversarial attack algorithms to find an approximated solution. Here we introduce two of them: the fast gradient sign method (FGSM) [32] and the basic iterative method (BIM) [33].

1) FGSM: The main idea of FGSM is to generate the adversarial examples based on the gradient information of the input. Specifically, it calculates the gradient map of an input image, and then adds or subtracts the sign of this input gradient map in the original image with multiplying a small scaling factor. The generation of adversarial examples can be formulated as

$$\begin{cases} x' = x + \eta \cdot \text{sign}\big(\nabla_x L\big(\theta, x, y_{\text{orginal}}\big)\big), & \text{if untargeted} \\ x' = x - \eta \cdot \text{sign}\big(\nabla_x L\big(\theta, x, y_{\text{target}}\big)\big), & \text{if targeted} \end{cases} \tag{3}$$

where L and θ denote the loss function and parameters of the victim model. η is used to control the magnitude of the perturbation. In untargeted attack, the adversarial example will drive the output away from the original correct class, which results from the gradient ascent-based input modification; while in targeted attack, the output under the adversarial example goes toward the targeted class, owing to the gradient descent-based input modification.

2) BIM: BIM algorithm is actually the iterative version of the above FGSM, which updates the adversarial examples in an iterative manner until the attack succeeds. The generation of adversarial examples in BIM is governed by

$$\begin{cases} x'_{k+1} = x'_k + \eta \cdot \text{sign}\big(\nabla_{x'_k} L\big(\theta, x'_k, y_{\text{orginal}}\big)\big), & \text{if untargeted} \\ x'_{k+1} = x'_k - \eta \cdot \text{sign}\big(\nabla_{x'_k} L\big(\theta, x'_k, y_{\text{target}}\big)\big), & \text{if targeted} \end{cases} \tag{4}$$

where k is the iteration index. Specifically, x'_k equals the original input when $k = 0$.

In ANNs, several advanced attack methods can be potentially extended beyond BIM-based algorithm by optimizing the perturbation bound [34]–[37] or avoiding the gradient calculation [38]–[41]. In this work, we aim at the preliminary exploration of an effective gradient-based SNN attack, thus adopting the most classic BIM algorithm in our design. We leave the SNN attack with different approaches in future work.

III. Challenges in SNN Attack

Even though the attack methodology can be independent of how the model is trained (e.g., gradient-free unsupervised learning [42] or spatial-gradient-based supervised learning [17]) and it is not necessary to compute gradients when finding adversarial examples (e.g., using trial-and-error methods [15], [16]), we take SNN models trained by BPTT with high recognition accuracy for example and focus on the spatiotemporal-gradient-based attack due to the potential for high attack success rate. Therefore, all our following discussions about the challenges are restricted in this context. Fig. 2(a) briefly illustrates the workflow of adversarial attacks based on gradients. There are three stages: forward pass to obtain the model prediction, backward pass to calculate the input gradient, and input update to generate the adversarial example. This flow is straightforward to implement in ANNs, as shown in Fig. 2(b). However, the case becomes complicated in the SNN scenario, where the processing is based on binary spikes with temporal dynamics rather than continuous activations with immediate response. According to Fig. 2(c), we attempt to identify the challenges in SNN attack to distinguish from the ANN attack and compare our solution with prior studies in Sections III-A and III-B.

A. Challenges and Solutions

1) Acquiring Spatiotemporal Gradients: For SNNs, it is difficult to acquire the spatiotemporal gradients using conventional SNN learning algorithms for the generation of adversarial examples with both spatial and temporal components. For example, the unsupervised learning rules such as spike timing dependent plasticity (STDP) [18] update synapses according to the activities of local neurons, which cannot help calculate the input gradients. The ANN-to-SNN-conversion learning methods [20] simply convert an SNN learning problem into an ANN one with only spatial information, leading to the incapability in capturing temporal input gradients. Recently, the BPTT based learning algorithm [2], [5], [21]–[24] is broadly studied. This emerging supervised learning promises accurate SNN attack via the direct acquisition of input gradients in both spatial and temporal dimensions, which is adopted by us.

2) Incompatible Format Between Gradients and Inputs: The input gradients are in continuous values, while the SNN inputs are in binary spikes (see the left of Fig. 2(c), each point represents a spike event, i.e., "1"; otherwise it is "0"). This data format incompatibility impedes the generation of spike-based adversarial examples if we consider the conventional gradient accumulation. In this work, we propose a G2S converter to convert continuous gradients to spike-compatible ternary gradients. This design exploits probabilistic sampling, sign extraction, and overflow-aware transformation, which can

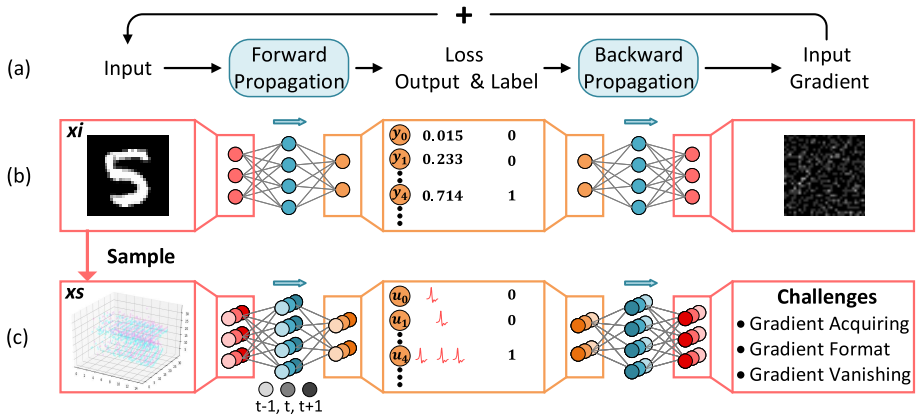


Fig. 2. Illustration of gradient-based adversarial attack. (a) Overall flow including forward pass, backward pass, and input update. (b) Adversarial attack in ANNs. (c) Adversarial attack in SNNs and its challenges. xi and xs represent an input in image and spike formats, respectively.

simultaneously maintain the spike format and control the perturbation magnitude.

3) Gradient Vanishing Problem: The firing function in the LIF model in (1) is actually a step function that is nondifferentiable. To address this issue, an approximation function is introduced to simulate the derivative of the firing activity [21]. However, this approximation brings abundant zero gradients outside the gradient window (to be shown later), leading to severe gradient vanishing during backpropagation. We find that the input gradient map can be all-zero sometimes, which interrupts the gradient-based update of adversarial examples. To this end, we propose an RSF to construct ternary gradients that can randomly flip the binary inputs in the case of all-zero gradients. We use a baseline sampling factor to bound the overall turnover rate, making the perturbation magnitude controllable.

B. Comparison With Prior Work on SNN Attack

The study on SNN attack is still in its infant stage. We only find several related works talking about this topic. In this subsection, we summarize their approaches and clarify our differences compared with them.

1) Trial-and-Error Input Perturbation: Such attack algorithms perturb inputs in a trial-and-error manner by monitoring the variation of outputs. For example, Marchisio *et al.* [15] modify the original image inputs before spike sampling. They first select a block of pixels in the images and then add a positive or negative unit perturbation onto each pixel. During this process, they always monitor the output change to determine the perturbation until the attack succeeds or the perturbation exceeds a threshold. However, this image-based perturbation is not suitable for the data sources with only spike events [26], [27]. In contrast, Bagheri *et al.* [16] directly perturb the spike inputs rather than the original image inputs. The main idea is to flip the input spikes and also monitor the outputs.

2) SNN/ANN Model Conversion: Sharmin *et al.* [17] convert the SNN attack problem into an ANN one. They first build an ANN substitute model that has the same network structure and parameters copied from the trained SNN model. The gradient-based adversarial attack is then conducted on the built ANN counterpart to generate the adversarial examples.

These existing works suffer from several drawbacks that would eventually degrade the attack effectiveness. For the trial-and-error input perturbation methods, the computational complexity is quite high due to the large search space without the guidance of supervised gradients. Specifically, each selected element of the inputs needs to run the forward pass once (for spike perturbation) or twice (for image perturbation) to monitor the outputs. The total computational complexity is Iter $\times$ N $\times$ C_{FP}, where Iter is the number of attack iterations, N represents the size of search space, and C_{FP} is the computational cost of each forward pass. This complexity is much higher than the normal one, i.e., Iter $\times$ $(C_{\mathrm{FP}} + C_{\mathrm{BP}})$, due to the large N. Because it is difficult to find the optimal perturbation in such a huge space, the attack effectiveness cannot be satisfactory given a limited search time in reality. Regarding the SNN/ANN model conversion method, an extra model transformation is needed and the temporal gradient information is lost during the ANN pretraining. Using a different model to find gradients and the missing of temporal components will compromise the attack effectiveness in the end. Moreover, this method is not applicable to the spiking data sources without the help of extra signal conversion.

Compared with the above works we calculate the gradients in both spatial and temporal dimensions without extra model conversion, which matches the natural SNN behaviors. Then, the proposed G2S and RSF enable the generation of spiking adversarial examples based on the continuous gradients even if when meeting the gradient vanishing. This direct generation of spiking adversarial examples makes our methodology suitable for the spiking data sources. For the SNN models using image-based data sources, our solution is also applicable with a simple temporal aggregation of spatiotemporal gradients. In summary, Table I shows the differences between our work and prior work. We use effectiveness to assess an attack method. Usually, an effective attack method can achieve a high attack success rate with relatively low complexity and better compatibility of input data formats.

Please note that we focus on the white-box attack in this article. Specifically, in the white-box attack scenario, the adversary knows the network structure and model parameters (e.g., weights, u_{th}, etc.) of the victim model. The reason for this scenario selection lies in that the white-box attack

TABLE I
COMPARISON WITH PRIOR WORK ON SNN ATTACK

Attack Method	Data Source	Spatiotemporal Gradient	Computational Complexity	Attack Effectiveness
Trial-and-Error [15]	Image	✗	$Iter \times N \times 2C_{FP}$	Low
Trial-and-Error [16]	Spike	✗	$Iter \times N \times C_{FP}$	Low
Model Conversion [17]	Image	✗	$Iter \times (C_{FP} + C_{BP})$	Low
This Work	Spike/Image	✓	$Iter \times (C_{FP} + C_{BP})$	High

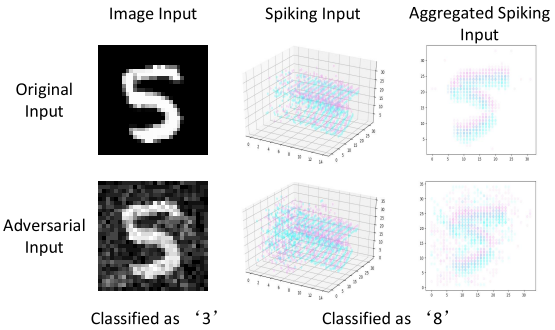


Fig. 3. Data format of original inputs and adversarial examples. The red and blue colors denote two spike channels induced by dynamic vision sensors [26], [31].

is the fundamental step to understand the adversarial attacks. Furthermore, the methodology built for the white-box attack can be easily transferred to the black-box attack in the future.

IV. ADVERSARIAL ATTACKS AGAINST SNNS

In this section, we first introduce the input data format briefly and then explain the flow, approach, and algorithm of our attack methodology in detail.

Input Data Format: It is natural for an SNN model to handle spike signals. Therefore, considering the datasets containing spike events, such as N-MNIST [26] and CIFAR10-DVS [27], is the first choice. In this case, the input is originally in a spatiotemporal pattern with a binary value for each element (0-nothing; 1-spike). The attacker can flip the state of selected elements, while the binary format must be maintained. The image datasets are also widely used in the SNN field by converting them into spiking version. There are different ways to perform the data conversion, such as rate coding [2], [5], [43] and latency coding [44]–[46]. In this work, we adopt the former scheme based on Bernoulli sampling that converts the pixel intensity to a spike train, where the spike rate is proportional to the intensity value. In this case, the attacker can modify the intensity value of selected pixels by adding the continuous perturbation. Fig. 3 illustrates the adversarial examples in these two cases.

A. Attack Flow Overview

The overview of the proposed adversarial attack against SNNs is illustrated in Fig. 4. The basic flow adopts the BIM method given in (4). The perturbation for spikes can only flip the binary states of selected input elements rather than add continuous values. Therefore, to generate spiking adversarial examples, the search of candidate elements is more important than the perturbation magnitude. FGSM cannot do this since it only explores the perturbation magnitude, while BIM realizes this by searching for new candidate elements in different attack iterations. Next, we describe the specific flow for spiking inputs and image inputs individually.

1) Spiking Inputs: The blue arrows in Fig. 4 illustrate this case. The generation of spiking adversarial examples relies on three steps as follows. In step ①, the continuous gradients are calculated in the FP and BP stages by

$$\begin{cases} \delta s'_k = \nabla_{xs'_k} L\big(\theta, xs_k, y_{\text{original}}\big), & \text{if untargeted} \\ \delta s'_k = -\nabla_{xs'_k} L\big(\theta, xs_k, y_{\text{target}}\big), & \text{if targeted} \end{cases} \tag{5}$$

where $\delta s'_k$ represents the input gradient at the kth iteration. Since all elements in $\delta s'_k$ are continuous values, they cannot be directly accumulated onto the spiking inputs xs_k. Therefore, in step ②, we propose G2S to convert the continuous gradient to a ternary one compatible with the spike input, which can simultaneously maintain the input data format and control the perturbation magnitude. When the input gradient vanishes (i.e., all elements in $\delta s'_k$ are zero), we propose RSF to construct a ternary gradient that can randomly flip the input spikes with a controllable turnover rate. At last, step ③ accumulates the ternary gradients onto the spiking input.

2) Image Inputs: Sometimes, the benchmarking models convert image datasets to spike inputs via Bernoulli sampling. In this case, one more step is needed to generate image-style adversarial examples, which is shown by the red arrows in Fig. 4. After the above step ②, the ternary gradient map should be aggregated in the temporal dimension according to $\delta i_k = (1/T)\sum_{t=1}^{T} \delta s_k^t$. In each update iteration, the intensity value of xi_k will be clipped within [0, 1].

B. Acquisition of Spatiotemporal Gradients

We introduce the state-of-the-art supervised learning algorithms for SNNs [5], [21], [23], which are inspired by the BPTT to acquire the gradients in both spatial and temporal dimensions. Usually, the original LIF neuron model in (1) is converted to its equivalent iterative version. Specifically, we have

$$\begin{cases} u_i^{t+1,n+1} = e^{-\frac{dt}{\tau}} u_i^{t,n+1}\Big(1 - o_i^{t,n+1}\Big) + \sum_j w_{ij}^n o_j^{t+1,n} \\ o_i^{t+1,n+1} = \text{fire}\Big(u_i^{t+1,n+1} - u_{\text{th}}\Big) \end{cases} \tag{6}$$

where t and n represent the timestep and layer, respectively. dt is the timestep length, and $e^{-(dt/\tau)}$ reflects the leakage effect of the membrane potential. fire$(\cdot)$ is a step function, which satisfies fire$(x) = 1$ when $x \geq 0$, otherwise fire$(x) = 0$. Note that a spike can be simply modeled as a binary event (1 or 0) in the above discrete-time domain, which differs from that in the continuous time domain in (1).

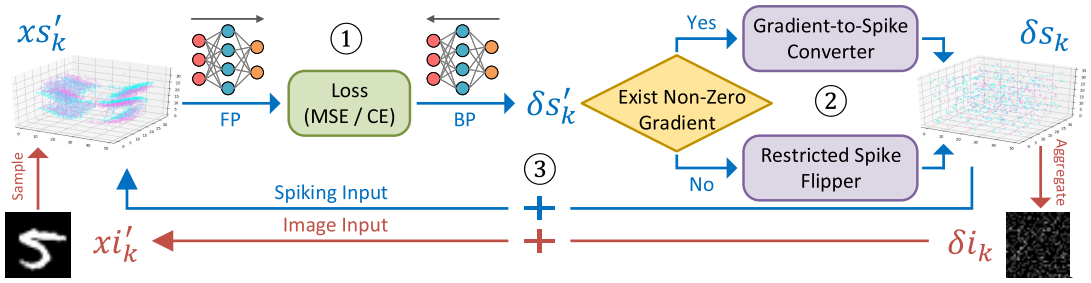


Fig. 4. Overview of the adversarial attack flow for SNNs with spiking or image inputs. The flow consists of: ① calculating continuous spatiotemporal input gradients via BPTT; ② generating spike-compatible input gradients; and ③ updating adversarial examples. For image-based inputs, an additional aggregation of the input gradients along the temporal dimension is needed.

In the output layer, we adopt the commonly-used spike rate coding scheme for the recognition, i.e., the neuron firing the most spikes becomes the winner that indicates the class prediction. The spatiotemporal spike pattern of the output layer is converted into a spike rate vector, described as

$$y_i = \frac{1}{T}\sum_{t=1}^{T} o_i^{t,N} \tag{7}$$

where N is the output layer index. This spike rate vector can be regarded as the normal output vector in ANNs. With this output conversion, the typical loss functions L for ANNs, such as MSE and CE, can also be applied in the loss function for SNNs.

Based on the LIF neuron model, the gradient propagation can be governed by

$$\begin{cases} \dfrac{\partial L}{\partial o_i^{t,n}} = \displaystyle\sum_j \frac{\partial L}{\partial u_j^{t,n+1}}\frac{\partial u_j^{t,n+1}}{\partial o_i^{t,n}} + \frac{\partial L}{\partial u_i^{t+1,n}}\frac{\partial u_i^{t+1,n}}{\partial o_i^{t,n}} \\ \dfrac{\partial L}{\partial u_i^{t,n}} = \dfrac{\partial L}{\partial o_i^{t,n}}\dfrac{\partial o_i^{t,n}}{\partial u_i^{t,n}} + \dfrac{\partial L}{\partial u_i^{t+1,n}}\dfrac{\partial u_i^{t+1,n}}{\partial u_i^{t,n}}. \end{cases} \tag{8}$$

However, the firing function is nondifferentiable, i.e., $(\partial o/\partial u)$ does not exist. As mentioned earlier, a Dirac-like function is introduced to approximate its derivative [21]. Specifically, $(\partial o/\partial u)$ can be estimated by

$$\frac{\partial o}{\partial u} \approx \begin{cases} \dfrac{1}{a}, & |u - u_{\text{th}}| \le \dfrac{a}{2} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where a is a hyperparameter to control the gradient width. This approximation indicates that only the neurons whose membrane potential is close to the firing threshold have the chance to let gradients pass through, as shown in Fig. 5. It can be seen that abundant zero gradients are produced, which might lead to the gradient vanishing problem (all input gradients become zero).

C. G2S Converter

There are two goals in the design of G2S converter in each iteration: 1) the final gradients should be compatible with the spiking inputs, i.e., keeping the spike format unchanged after the gradient accumulation and 2) the perturbation magnitude should be imperceptible, i.e., limiting the number of nonzero gradients. To this end, we design three steps: probabilistic sampling, sign extraction, and overflow-aware transformation, which are illustrated in Fig. 6.

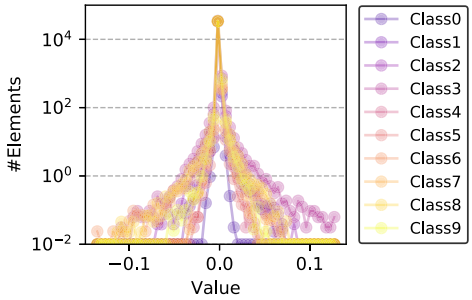


Fig. 5. Distribution of input gradients overall 500 samples from N-MNIST. The model is trained with MSE loss. Most of gradients are zero which might lead to the gradient vanishing problem.

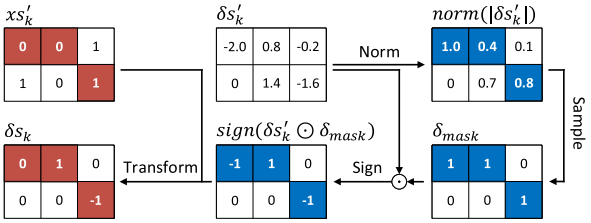


Fig. 6. Illustration of G2S converter with probabilistic sampling reducing the number of modified points, sign extraction ternarizing the continuous gradients for spike compatibility, and overflow-aware transformation clipping the data range in adversarial examples.

1) Probabilistic Sampling: The absolute value of the input gradient $|\delta s'_k|$ obtained by (5) is first normalized to lie in the range of [0, 1]. Then, the normalized gradient map $\text{norm}(|\delta s'_k|)$ is sampled to produce a binary mask, in which the 1s indicate the locations where gradients can pass through. The probabilistic sampling for each gradient element obeys

$$\begin{cases} P(\delta_{\text{mask}} = 1) = \text{norm}\left(\left|\delta s'_k\right|\right) \\ P(\delta_{\text{mask}} = 0) = 1 - \text{norm}\left(\left|\delta s'_k\right|\right). \end{cases} \tag{10}$$

By multiplying the resulting mask with the original gradient map, the number of nonzero elements can be reduced significantly. To evidence this conclusion, we run the attack against the SNN model with a network structure to be provided in Table V over 500 spiking inputs from N-MNIST, and the results are presented in Fig. 7. Given MSE loss and untarget attack scenario, the number of nonzero elements in $\delta s'_k$ could reach 2^{10}. After using the probabilistic sampling, the number of nonzero elements in $\delta s'_k \odot \delta_{\text{mask}}$ can be greatly decreased, masking out $> 96\%$.

2) Sign Extraction: Now, we explain how to generate a ternary gradient map where each element is in $\{-1, 0, 1\}$

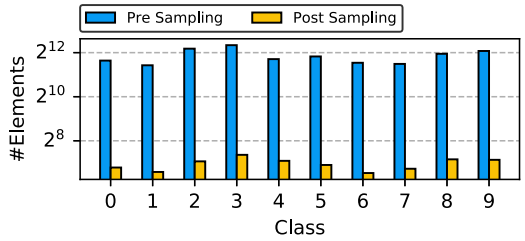


Fig. 7. Number of elements with nonzero input gradients before and after the probabilistic sampling. After the probabilistic sampling step, the number of selected nonzero input elements for modification is reduced a lot for each class.

TABLE II
OVERFLOW-AWARE GRADIENT TRANSFORMATION

	Before Transformation		After Transformation	
xs'_k	$\delta s''_k$	$xs_k + \delta s''_k$	δs_k	$xs'_k + \delta s_k$
0/1	0	0/1	0	0/1
0	1	1	1	1
1	1	2	0	1
0	-1	-1	0	0
1	-1	0	-1	0

to match the spike inputs. This step is simply based on a sign extraction

$$\delta s''_k = \text{sign}\big(\delta s'_k \odot \delta_{\text{mask}}\big) \tag{11}$$

where we define $\text{sign}(x) = 1$ if $x > 0$, $\text{sign}(x) = 0$ if $x = 0$, and $\text{sign}(x) = -1$ otherwise.

3) Overflow-Aware Transformation: Although the above $\delta s''_k$ is able to be ternary, it cannot ensure that the final adversarial example generated by input gradient accumulation is still limited in {0, 1}. For example, an original "0" element in xs_k with a "−1" gradient or an original "1" element with a "1" gradient will yield a "−1" or "2" input that is out of {0, 1}. This overflow breaks the data format of binary spikes. To address this issue, we propose an overflow-aware gradient transformation to constrain the range of the final adversarial example, which is illustrated in Table II.

After introducing the above three steps, now the function of G2S converter can be briefly summarized as follows:

$$\delta s_k = \text{transform}\big[\text{sign}\big(\delta s'_k \odot \delta_{\text{mask}}\big),\ xs'_k\big] \tag{12}$$

where transform(·) denotes the overflow-aware transformation. The G2S converter is able to simultaneously keep the spike compatibility and control the perturbation magnitude.

D. RSF

Table III identifies the gradient vanishing issue in SNNs, which is quite severe. Based on the previous study [21], the hyperparameter a in (9) has an influential impact on the gradient approximation of the fire function. Generally, a too small a would prevent gradients from passing through the neurons in the backward pass, i.e., aggravating the gradient vanishing problem. However, a too large a cannot precisely approximate the gradient of the firing function that should be a delta function rather than a wide pulse. Therefore, the gradient vanishing problem cannot be fully resolved by simply increasing a. Therefore, we empirically select proper a values (see Table VI) and propose RSF to address the gradient vanishing problem. Specifically, we design two steps for RSF: element selection and gradient construction, which are illustrated in Fig. 8.

TABLE III
NUMBER OF INPUTS WITH ALL-ZERO GRADIENTS AT THE FIRST ATTACK ITERATION. WE TEST THE UNTARGETED ATTACK WITH OVER 500 INPUTS FOR EACH DATASET

Dataset	N-MNIST	CIFAR10-DVS	MNIST	CIFAR10
#grad.-vanish. inputs (MSE)	130	41	436	103
#grad.-vanish. inputs (CE)	256	32	471	105

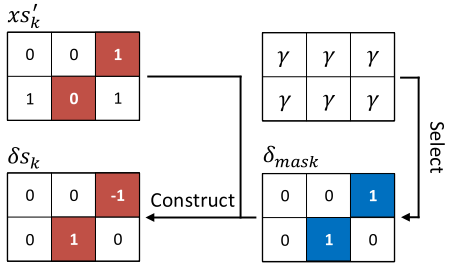


Fig. 8. Illustration of RSF with element selection picking candidate elements through probabilistic sampling and gradient construction creating spike-compatible gradients through spike flipping.

TABLE IV
GRADIENT CONSTRUCTION TO FLIP SPIKING INPUTS

		After Construction	
xs'_k	δ_{mask}	δs_k	$xs'_k + \delta s_k$
0/1	0	0	0/1
0	1	1	1
1	1	-1	0

1) Element Selection: This step is to select the elements to flip the spike event. We provide a gradient initialization that sets all elements to γ as the example provided in Fig. 8. γ is a factor within the range of [0, 1], which controls the number of nonzero gradients after RSF. Now the probabilistic sampling in (10) is still applicable to generate the mask δ_{mask}.

2) Gradient Construction: To maintain the spike format of adversarial examples, we just flip the state of spiking inputs in the selected region. Table IV illustrates the construction of ternary gradients that are able to flip the spiking inputs.

With the above two steps, the spiking inputs can be flipped randomly with a good control of the turnover rate. The overall function of RSF can be expressed as

$$\delta s_k = \text{construct}\big(\delta_{\text{mask}},\ xs'_k\big). \tag{13}$$

E. Overall Attack Algorithm

Based on the explanations of the G2S converter and RSF, Algorithm 1 provides the overall attack algorithm corresponding to the attack flow illustrated in Fig. 4. There are several hyperparameters in our algorithm, such as the maximum attack iteration number (Iter), the norm format (p) to quantify the perturbation magnitude, the perturbation magnitude upper bound (ϵ), the gradient scaling rate (η), and the sampling

factor (γ) in RSF. Notice that we use the average perturbation per point as the metric to evaluate the perturbation magnitude for adversarial example with N pixel points, i.e., $(1/N)\|x'_{k+1} - x'_0\|_p$.

Algorithm 1 Overall SNN Attack Algorithm

Input: x, $Iter$, p, ϵ, η, γ;
if *image input* **then** $xi_0 = x$; **end**
else $xs'_0 = x$; **end**
for $k = 1$ *to* $Iter$ **do**
  if *image input* **then**
    $xs'_k \leftarrow$ Bernoulli sampling on xi'_k;
  end
  Get $\delta s'_k$ through Equation (5);
  if *gradient vanishing occurs in* $\delta s'_k$ **then**
    // RSF
    $\delta_{mask} \leftarrow$ Probabilistic sampling on γ;
    $\delta s_k = construct(\delta_{mask},\ xs'_k)$;
  end
  else
    // G2S converter
    $\delta_{mask} \leftarrow$ Probabilistic sampling on $norm(|\delta s'_k|)$;
    $\delta s_k = transform[sign(\delta s'_k \odot \delta_{mask}),\ xs'_k]$;
  end
  if *image input* **then**
    $\delta i_k \leftarrow \frac{1}{T}\sum_{t=1}^{T} \delta s^t_k$; // Temporal aggregation
    $xi'_{k+1} = xi'_k + \delta i_k$;
    if $\frac{1}{N}\|xi'_{k+1} - xi'_0\|_p \geqslant \epsilon$ **then**
      break; // Attack failed
    end
    if *attack succeeds* **then**
      return xi'_{k+1}; // Attack successful
    end
  end
  else
    $xs'_{k+1} = xs'_k + \delta s_k$;
    if $\frac{1}{N}\|xs'_{k+1} - xs'_0\|_p \geqslant \epsilon$ **then**
      break; // Attack failed
    end
    if *attack succeeds* **then**
      return xs'_{k+1}; // Attack successful
    end
  end
end

V. Loss Function and Firing Threshold

In this work, we consider two design knobs that affect the SNN attack effectiveness: the loss function during training and the firing threshold of the penultimate layer during the attack.

A. MSE and CE Loss Functions

We compare two widely used loss functions, MSE loss and CE loss. We observe that gradient vanishing occurs more often when the model is trained by CE loss. It seems that there is a "trap" region in this case. Specifically, the output neurons cannot change the response anymore no matter how RSF modifies the input. As shown in Fig. 9(a), when we use CE loss during training, the gradient is usually vanished between the decision boundaries (i.e., the shaded area); while this phenomenon seldom happens if MSE loss is used.

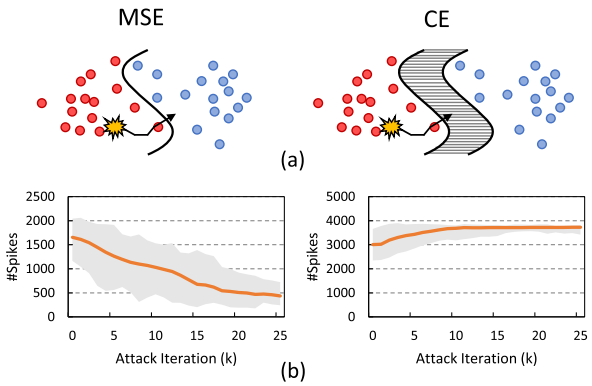


Fig. 9. Loss function analysis. (a) Decision boundary comparison. (b) Number of output spikes in the penultimate layer at different attack iterations. The shaded area in (a) represents a "trap" area that receives zero gradient; the larger number of spikes in the penultimate layer under CE probably introduces the "trap" effect.

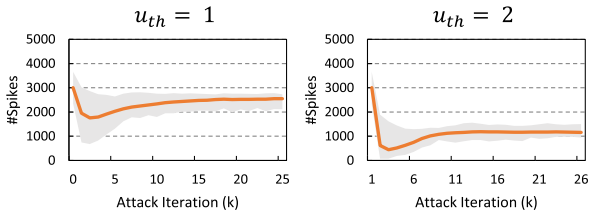


Fig. 10. Number of output spikes in the penultimate layer with different firing threshold in that layer. The increase of the firing threshold in the penultimate layer is able to reduce the number of spikes.

For a deeper understanding, we examine the output pattern of the penultimate layer (during untargeted attack) since it directly interacts with the output layer, as depicted in Fig. 9(b). Here the network structure will be provided in Table V and the 500 test inputs are randomly selected from the N-MNIST dataset. When the training loss is MSE, the number of output spikes in the penultimate layer gradually decreases as the attack process evolves. On the contrary, the spike number first increases and then stays unchanged for the CE trained model. Based on this observation, one possible hypothesis is that more output spikes in the penultimate layer might increase the distance between decision boundaries, thus introducing the mentioned "trap" region with gradient vanishing.

B. Firing Threshold of the Penultimate Layer

As introduced in Section V-A, the models trained by CE loss are prone to output more spikes in the penultimate layer, leading to the "trap" region that makes the attack difficult. To address this issue, we increase the firing threshold of the penultimate layer during the attack to reduce the number of spikes there. Notice that we only modify the firing threshold in the FP stage during the generation of adversarial examples. With the threshold tuning, we present the number of spikes

TABLE V

NETWORK STRUCTURE ON DIFFERENT DATASETS. "C," "AP," AND "FC" DENOTE CONVOLUTIONAL LAYER, AVERAGE POOLING LAYER, AND FC LAYER, RESPECTIVELY

Dataset	Network Structure
Spike	Input-128C3-128C3-AP2-384C3-384C3-AP2-1024FC-512FC-10FC
Image	Input-128C3-256C3-AP2-512C3-AP2-1024C3-512C3-1024FC-512FC-10FC
Gesture-DVS	Input-64C3-128C3-AP2-128C3-AP2-256FC-11FC

TABLE VI

HYPERPARAMETER SETTINGS AND MODEL ACCURACY DURING TRAINING

Datasets	Gesture-DVS	N-MNIST	CIFAR10-DVS	MNIST	CIFAR10
Input Size	$32 \times 32 \times 2$	$34 \times 34 \times 2$	$42 \times 42 \times 2$	$28 \times 28 \times 1$	$32 \times 32 \times 3$
u_{th}	0.3	0.3	0.3	0.3	0.3
$e^{-\frac{dt}{\tau}}$	0.3	0.3	0.3	0.25	0.25
a	0.5	0.5	0.5	1	1
T	60	15	10	15	15
Time Bin	1ms	5ms	5ms	-	-
Acc (MSE)	91.32%	99.49%	64.60%	99.27%	76.37%
Acc (CE)	-	99.42%	64.50%	99.52%	77.27%

again in Fig. 10, where the CE loss is used and other settings are the same with those in Fig. 9(b). Compared to the original threshold setting ($u_{\text{th}} = 0.3$) in the previous experiments, the number of output spikes in the penultimate layer can be decreased significantly on average. Later experiments in Section VI-D will evidence that this tuning of firing threshold is able to improve the adversarial attack effectiveness.

VI. EXPERIMENT RESULTS

A. Experiment Setup

We design our experiments on both spiking and image datasets. The spiking datasets include N-MNIST [26] and CIFAR10-DVS [27] which are captured by dynamic vision sensors [31]; while the image datasets include MNIST [28] and CIFAR10 [29]. For these two kinds of dataset, we use different network structure, as listed in Table V. For each dataset, the detailed hyperparameter setting during training and the trained accuracy are shown in Table VI. For each model, we train it for 50 epochs, and the learning rate decays by 0.1 at epoch 35. The default loss function is MSE. Since we focus on the attack methodology in this work, we do not use the optimization techniques such as input encoding layer, neuron normalization, and voting-based classification [5].

We set the maximum iteration number of adversarial attack, i.e., Iter in Algorithm 1, to 25. We randomly select 50 inputs in each of the ten classes for untargeted attack and ten inputs in each class for a targeted attack. In a targeted attack, we set the target to all classes except the ground-truth one. We use attack success rate and average perturbation per point (i.e., $\|\delta\|_p$) as two metrics to evaluate the attack effectiveness. Specifically, the attack success rate is calculated in the same way as the prior work do [32], [47]: for an untargeted attack, it is the percentage of the cases that adversarial examples fool the model to output a different label from the ground-truth one; for a targeted attack, it is the percentage of the cases that adversarial examples manipulate the model to output the target label. Noted, during the calculation of attack success rate, we only consider the original images that can be correctly classified to eliminate the impact of intrinsic model prediction errors. The reason that we use the same perturbation metric of point-to-point distance for both image-based and spike-based data sources is to simplify the comparison. In the perturbation calculation, we adopt L2 norm, i.e., $p = 2$. For image-based datasets, we normalize each input value into $[0, 1]$.

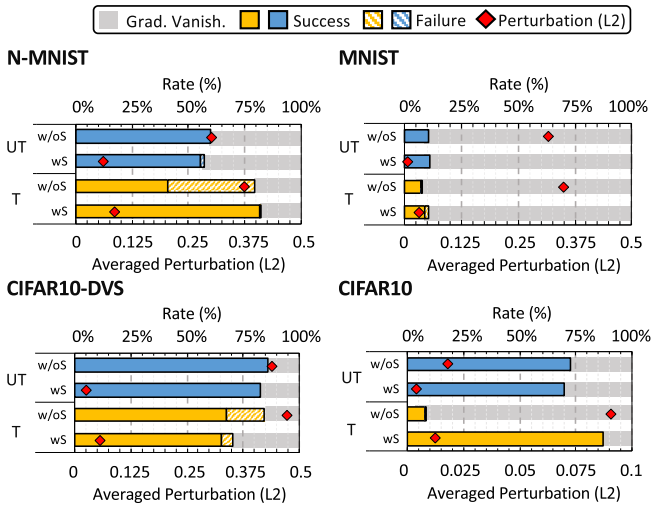


Fig. 11. Comparison of attack success rate and average perturbation over different datasets with and without probabilistic sampling in G2S converter. "T," "UT," "w/oS," and "wS" refer to targeted attack, untargeted attack, G2S without probabilistic sampling, and G2S with probabilistic sampling, respectively.

B. Influence of G2S Converter

We first validate the effectiveness of the G2S converter. Among the three steps in the G2S converter (i.e., probabilistic sampling, sign extraction, and overflow-aware transformation) as introduced in Section IV-C, the last two are needed in addressing the spike compatibility while the first one is used to control the perturbation amplitude. Therefore, we examine how does probabilistic sampling affect attack effectiveness. Note that we do not use RSF to solve the gradient vanishing here.

Fig. 11 presents the comparison of attack results over four datasets with or without the probabilistic sampling. In this section, we estimate only the attack success/failure rate for the input samples that do not encounter the gradient vanishing problem during the attack. Thus, three parts add up to 100%, i.e., the success rate, the failure rate, and the percentage of the input samples that encounter the gradient vanishing problem. We provide the following observations. First, the required perturbation amplitude of a targeted attack is higher than that of an untargeted attack, and the success rate of a targeted attack is usually lower than that of an untargeted attack. These results reflect the difficulty of a targeted attack that needs to move the output to an expected class accurately. Second, probabilistic sampling can significantly reduce the perturbation amplitude in all cases because it removes many small gradients. Third, the probabilistic sampling can maintain the attack success rate in most cases under targeted attack. Specifically, on N-MNIST and CIFAR10 datasets, the probabilistic

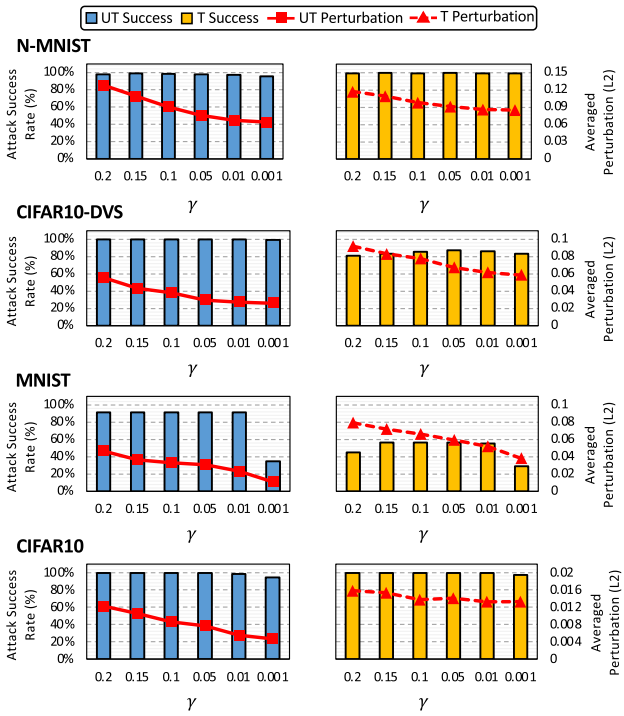


Fig. 12. Attack success rate and average perturbation with different γ settings. "T" and "UT" refer to targeted attack and untargeted attack, respectively.

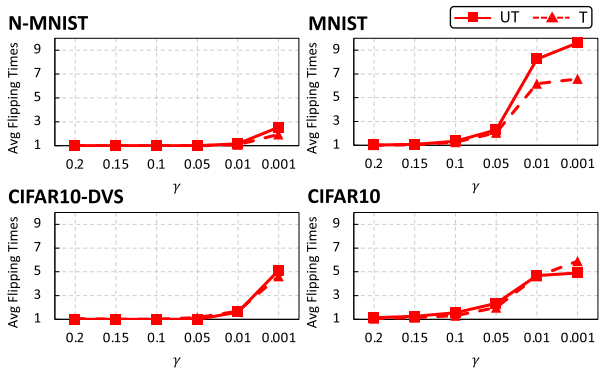


Fig. 13. Flipping times with different γ settings in RSF. "T" and "UT" refer to targeted attack and untargeted attack, respectively. A smaller γ increases the flipping times since the perturbation is not strong enough to push the model out of the gradient vanishing region.

TABLE VII

IMPACT OF THE LOSS FUNCTION ON THE ATTACK SUCCESS RATE (WITHOUT FIRING THRESHOLD OPTIMIZATION). "T" AND "UT" REFER TO TARGETED ATTACK AND UNTARGETED ATTACK, RESPECTIVELY

	MSE Loss		CE Loss	
Dataset	UT	T	UT	T
N-MNIST	97.38%	99.44%	90.12%	16.78%
CIFAR10-DVS	100%	86.35%	100%	82.95%
MNIST	91.31%	55.33%	93.16%	47.81%
CIFAR10	98.68%	99.72%	98.48%	40.51%

sampling can improve the targeted attack success rate a lot (e.g., >80% on CIFAR10). Although the targeted attack success rate is slightly lowered after applying the probabilistic sampling on CIFAR10-DVS, it is not mainstream and might be caused by the restriction on the number of attack iterations. With the probabilistic sampling, the attack failure rate could be reduced to almost zero if the gradient does not vanish.

C. Influence of RSF

Then, we validate the effectiveness of RSF. In RSF, the hyper-parameter γ controls the number of selected elements, thus affecting the perturbation amplitude. Keep in mind that a larger γ indicates a larger perturbation via flipping the state of more elements in the spiking input.

We first analyze the impact of γ on the attack success rate and perturbation amplitude, as shown in Fig. 12. A similar conclusion as observed in Section VI-B also holds, that the target attack is more difficult than the untargeted attack. As γ decreases, the number of elements with the flipped state is reduced, leading to smaller perturbation. Whereas, the impact of γ on the attack success rate depends heavily on the attack scenario and the dataset. For the easier untargeted attack, it seems that a slightly large γ is already helpful. The attack success rate will be saturated close to 100% even if at $\gamma = 0.01$. For the targeted attack with higher difficulty, it seems that there exists an obvious peak success rate on these datasets where the γ value equals 0.05. The results are reasonable since the impact of γ is twofold: i) a too large γ will result in a large perturbation amplitude and might cause a non-convergent attack; ii) a too small γ cannot move the model out of the region with gradient vanishing.

We also record the number of flipping times under different γ setting, as shown in Fig. 13. Here the "flipping times" means the number of iterations during the attack process where the gradient vanishing occurs and the spike flipping is needed. When γ is large, the number of flipping times can be only one since the perturbation is large enough to push the model out of the gradient vanishing region. As γ becomes smaller, the required number of flipping times becomes larger. In order to balance the attack success rate (see Fig. 12) and the flipping time (see Fig. 13), we finally recommend the setting of $\gamma = 0.05$ in RSF on the datasets we tested.

D. Influence of Loss Function and Firing Threshold

Additionally, we evaluate the influence of different training loss functions on the attack success rate. The comparison is summarized in Table VII. Here, the G2S converter and RSF are switched on. The model trained by CE loss leads to a lower attack success rate compared to the one trained by MSE loss, and the gap is especially large in the targeted attack scenario. As explained in Section V, this reflects the "trap" region of the models trained by CE loss due to the increasing spike activities in the penultimate layer during the attack.

To improve the attack effectiveness, we increase the firing threshold of the penultimate layer during the attack to reduce the spiking activities. Note that we only modify the penultimate layer's firing threshold in the forward pass during the generation of adversarial examples. The experimental results are provided in Fig. 14. For untargeted attacks, the increase of the firing threshold can improve the attack success rate to almost 100% on all datasets. For targeted attacks, the cases present different behaviors. Specifically, on image datasets

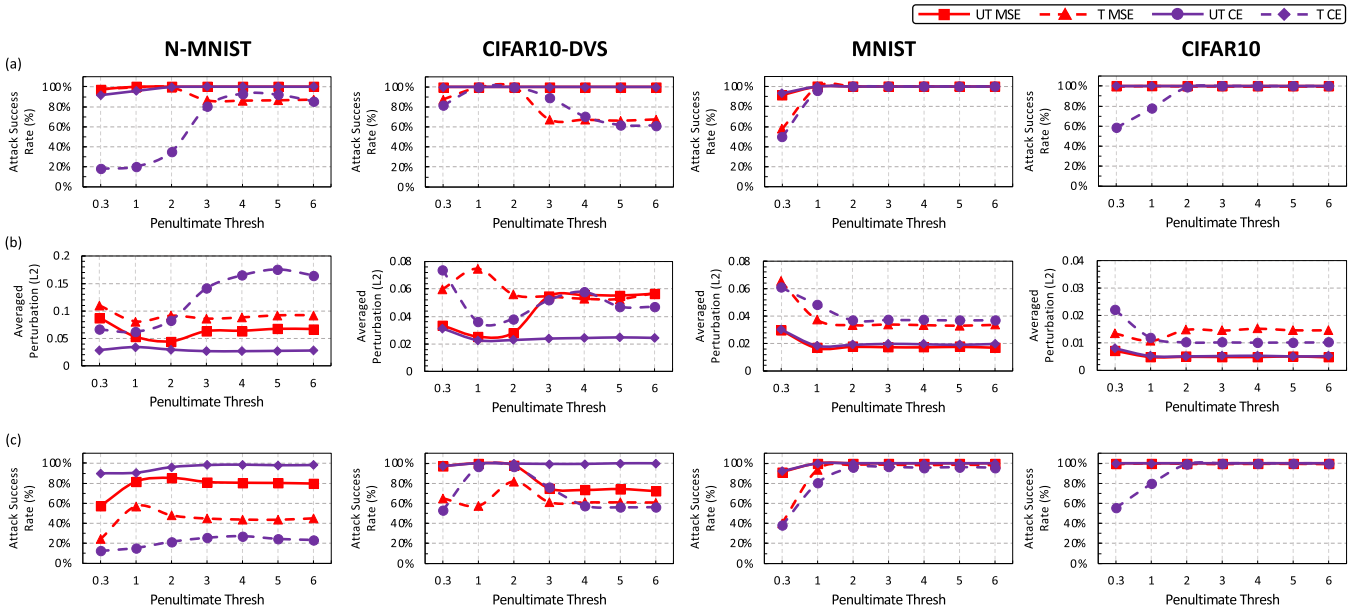


Fig. 14. Attack effectiveness with different firing threshold. (a) Attack success rate without strict ϵ bound. (b) Average perturbation without strict ϵ bound. (c) Attack success rate under strict perturbation bound ($\epsilon = 0.08$). "T" and "UT" refer to targeted attack and untargeted attack, respectively. In most cases, we can achieve a high attack success rate and acceptable perturbation with a slightly larger firing threshold at the penultimate layer, even if with strict perturbation bound ($\epsilon = 0.08$).

(i.e., MNIST and CIFAR10), the attack success rate can be quickly improved and remained at about 100%; while on spiking datasets (i.e., N-MNIST and CIFAR10-DVS), the attack success rate initially goes higher and then decreases, in other words, there exists the best threshold setting. This might be due to the sparse-event nature of the neuromorphic datasets, on which the number of spikes injected into the last layer will be decreased severely if the firing threshold becomes large enough, leading to a fixed loss value and thus a degraded attack success rate. Moreover, from the perturbation distribution, it can be seen that the increase of the firing threshold does not introduce much extra perturbation in most cases. All the above results indicate that appropriately increasing the firing threshold of the penultimate layer is able to improve the attack effectiveness significantly without enlarging the perturbation.

In Fig. 14(a), we do not strictly bound ϵ, in order to avoid disturbing the analysis of the firing threshold. The average perturbation magnitude values are shown in Fig. 14(b), which are relatively small (within 0.08 in most cases). We further analyze the attack success rate under the limitation of strict perturbation bounds. Specifically, during attack iterations, if the average perturbation per point is greater than a predefined value ϵ, the attack is considered a failure. As shown in Fig. 14(c), our attack method can still achieve a considerable attack success rate with $\epsilon = 0.08$ when compared to the results in Fig. 14(a) for most cases. While there is a degradation for targeted attack over the N-MNIST dataset, which may be caused by the high sparsity of the spike inputs in that dataset.

E. Effectiveness Comparison With Existing SNN Attack

As discussed in Section III-B, our attack is quite different from previous work using trial-and-error input perturbation [15], [16] or SNN/ANN model conversion [17]. Beyond the methodology difference, here we coarsely discuss the attack effectiveness. Due to the high complexity of the trial-and-error manner, the testing dataset is quite small (e.g., USPS dataset [15]) or even with only one single example [16]. In contrast, we demonstrate the effective adversarial attack on much larger datasets. For the SNN/ANN model conversion method [17], the authors show results on the CIFAR10 dataset. In that work, the authors used the accuracy loss of the model, which is caused by substituting the original inputs with the adversarial examples, for the evaluation of the attack effectiveness. We compare our attack results with theirs (inferred from the figure data in [17]) on CIFAR10 under different ϵ configurations, as shown in Table VIII. It can be seen that our attack method can incur more model accuracy loss in most cases, which indicates our better attack effectiveness.

TABLE VIII

COMPARISON OF THE ACCURACY LOSS BETWEEN OUR WORK AND PRIOR WORK [17] UNDER DIFFERENT PERTURBATION BOUNDS

ϵ	8/255	16/255	32/255	64/255
Untargeted [17]	37.50%	62.50%	75.00%	77.00%
Untargeted (ours)	50.47%	72.46%	76.67%	76.86%
Targeted [17]	20.00%	37.50%	52.50%	63.00%
Targeted (ours)	19.16%	42.36%	65.58%	71.48%

F. SNNs Versus ANNs Against Adversarial Attack

In this section, we further compare SNNs and ANNs against adversarial attacks. In essence, we make the comparison from two perspectives: the perturbation distance demanded for a successful attack; the transferability between the adversarial examples of ANNs and SNNs. Here, the transferability of model A's adversarial examples on model B represents the attack success rate of attacking model B with the adversarial examples generated by model A. In our evaluation, a larger

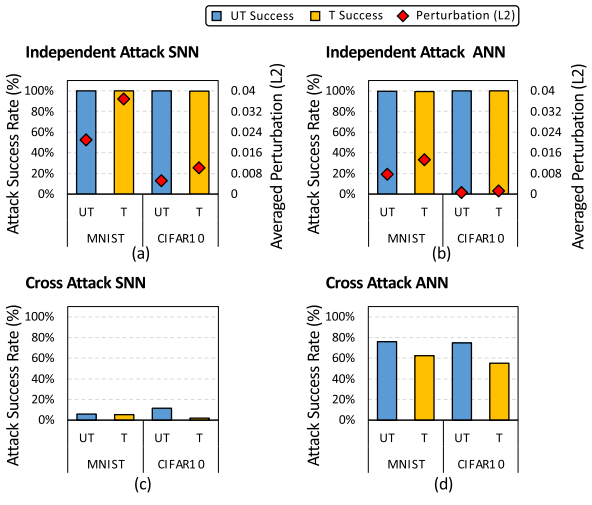


Fig. 15. Attack success rate comparison between ANNs and SNNs under gradient-based attack. "T" and "UT" refer to targeted attack and untargeted attack, respectively. (a) and (b) Independent attack. (c) and (d) Cross attack. In this scenario, attacking SNNs requires larger perturbation than attacking ANNs and the adversarial examples generated by attacking the ANN models fail to attack the SNN models.

perturbation distance indicates more challenges to attack a model; a lower transferability implies that the adversarial examples generated by one model are lower possible to attack other models successfully.

In this section, we select image-based datasets, MNIST and CIFAR10s. For ANN models, we use the same network structure as SNN models given in Table V. The training loss function is CE here. We test two attack scenarios: independent attack and cross attack. For the independent attack, the ANN models are attacked using the BIM method in (4); while the SNN models are attacked using the proposed gradient-based method and also a gradient-free method. Note that the firing threshold of the penultimate layer of SNN models during the attack is set to 2 in this section as suggested in Fig. 14. For the cross attack, we use the adversarial examples generated by attacking the SNN models to mislead the ANN models or vice versa.

From Fig. 15(a) and (b), we can easily observe that all attack success rates are quite high in the independent attack scenario. While attacking the SNN models requires larger perturbation than attacking the ANN models in the above experiment. From the results of the cross attack in Fig. 15(c) and (d), we find that using the adversarial examples generated by attacking ANN models to fool the SNN models is very difficult, with only $<12\%$ success rate, indicating the lower transferability of the ANN adversarial examples. The observation in this scenario reflects that SNN adversarial examples are easier to transfer to an ANN model with the same network structure.

Besides the gradient-based attack method, we also exam the gradient-free boundary attack method [37]. Since the computational complexity of boundary attack is much higher than the gradient-based attack, we only evaluate the untargeted attack. The results are depicted in Fig. 16. We find that the required perturbation to attack SNNs is still higher. However, under cross attack, the ANN adversarial examples present better transferability than the SNN adversarial examples,

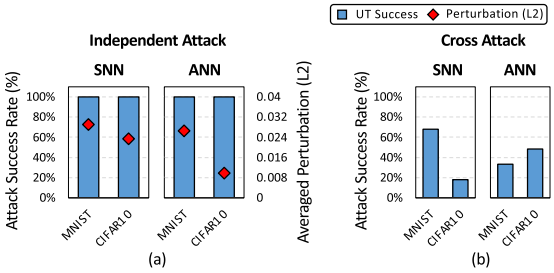


Fig. 16. Attack success rate comparison between ANNs and SNNs under untargeted boundary attack. (a) Independent attack. (b) Cross attack. In this scenario, attacking SNNs still requires larger perturbation than attacking ANNs; however, the ANN adversarial examples on MNIST present better transferability than the SNN adversarial examples.

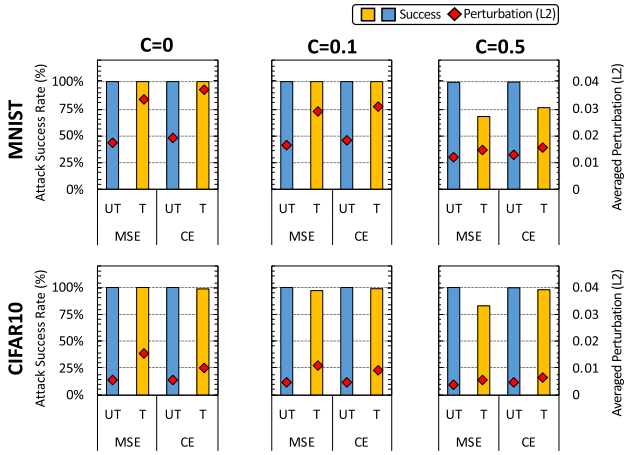


Fig. 17. Attack effectiveness with CWL2 under different settings of c. A larger c indicates a smaller perturbation but may compromise the attack success rate.

which indicates the SNN model in this scenario is easier to attack.

G. Other Gradient-Based Attack Methods and Datasets

In this subsection, we validate the effectiveness of the proposed attack methodology using more experiments with advanced attack methods (e.g., CWL2 [36]) and dynamic datasets (e.g., Gesture-DVS [48]).

CWL2 is an advanced adversarial attack method that is widely applied in ANN attack, which integrates a regularization item to restrict the magnitude of the perturbation. We tailor Algorithm 1 to perform CWL2 attack against SNN models over image-based inputs. The adversarial example generation follows:

$$xi'_{k+1} = xi'_k + \delta i_k - c \times \nabla_{xi'_k} \left\| xi'_k - xi'_0 \right\|_2^2 \tag{14}$$

where xi'_0 and xi'_k represent the original input and the adversarial example generated at the kth attack iteration. c is a parameter that determines the impact of regularization item. A larger c indicates smaller perturbation at the cost of possibly lower attack success rate. The CWL2 attack would degrade to the classic BIM attack when $c = 0$.

We tested the tailored SNN-oriented CWL2 attack on MNIST and CIFAR10 datasets with different configurations of c. As illustrated in Fig. 17, a slight increase of c ($c = 0.1$) helps reduce the perturbation magnitude without sacrificing

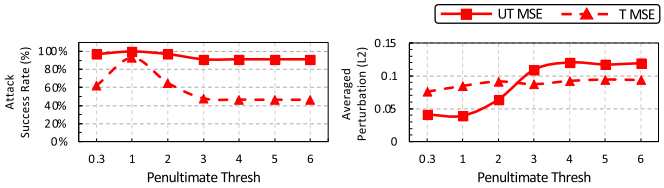


Fig. 18. Attack effectiveness on the Gesture-DVS dataset.

the attack success rate, compared to the results at ($c = 0$). However, when c is too large ($c = 0.5$), the attack success rate decreases. For example, the targeted attack success rate on MNIST dataset is reduced by up to 32.45% when $c = 0.5$.

In addition, we also applied our attack method on Gesture-DVS. The model configurations have been shown in Table VI. We only test the cases with the MSE training loss function for simplicity, and the attack results are shown in Fig. 18. Our methodology can still achieve a high attack success rate with acceptable perturbation even on this dynamic dataset. The trend of attack success rate variation under different penultimate layer threshold setting is similar to that on other spike-based datasets we have tested earlier.

VII. Conclusion and Discussion

SNNs have attracted broad attention and have been widely deployed in neuromorphic devices due to the importance of brain-inspired computing. Naturally, the security problem of SNNs should be considered. In this work, we first identify the challenges in attacking an SNN model with spatiotemporal-gradient-based methods, including the incompatibility between the spiking inputs and the continuous gradients, and the gradient vanishing problem. Second, we design a G2S converter and an RSF to address the mentioned two challenges, respectively. Our methodology can control the perturbation amplitude well and is applicable to both spiking and image data formats. Interestingly, we find that there is a "trap" region in SNN models trained by CE loss, which can be overcome by adjusting the firing threshold of the penultimate layer. We conduct extensive experiments on various datasets and show a 99%+ attack success rate in most cases, which is the best result on SNN attack. Furthermore, we compare the attack of SNNs and ANNs. From our empirical results, the adversarial examples for SNNs require a larger perturbation distance, but it still remains open whether SNNs can be more robust than ANNs against adversarial attacks.

For future work, we recommend several interesting topics. Although we only study the white-box adversarial attack to avoid shifting the focus of presenting our methodology, the black-box adversarial attack should be investigated because it is more practical. Fortunately, the proposed methods in this work can be transferred into the black-box attack scenario. Second, we only analyze the influence of loss function and firing threshold due to the page limit. It still remains an open question that whether other factors can affect the attack effectiveness, such as the gradient approximation form of the firing activities, the time window length for rate coding or the coding scheme itself, the network structure, and other solutions that can substitute G2S and RSF. Third, more appropriate evaluation metrics should be designed to evaluate the perturbation for spike data. Fourth, a more comprehensive research to compare the robustness between ANNs and SNNs against adversarial attack is an interesting topic. Fifth, the attack against physical neuromorphic devices rather than just theoretical models is more attractive. At last, compared to the attack methods, the defense techniques are highly expected for the construction of large-scale neuromorphic systems.

References

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.
[2] L. Deng *et al.*, "Rethinking the performance comparison between SNNS and ANNS," *Neural Netw.*, vol. 121, pp. 294–307, Jan. 2020.
[3] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.
[4] G. Haessig, A. Cassidy, R. Alvarez, R. Benosman, and G. Orchard, "Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 4, pp. 860–870, Aug. 2018.
[5] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1311–1318.
[6] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 994–1001, Apr. 2018.
[7] Z. Jonke, S. Habenschuss, and W. Maass, "Solving constraint satisfaction problems with networks of spiking neurons," *Frontiers Neurosci.*, vol. 10, p. 118, Mar. 2016.
[8] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.
[9] G. Shi, Z. Liu, X. Wang, C. T. Li, and X. Gu, "Object-dependent sparse representation for extracellular spike detection," *Neurocomputing*, vol. 266, pp. 674–686, Nov. 2017.
[10] T. Hwu, J. Isbell, N. Oros, and J. Krichmar, "A self-driving robot using deep convolutional neural networks on neuromorphic hardware," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 635–641.
[11] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.
[12] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.
[13] J. Pei *et al.*, "Towards artificial general intelligence with hybrid tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, Aug. 2019.
[14] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199
[15] A. Marchisio, G. Nanfa, F. Khalid, M. Abdullah Hanif, M. Martina, and M. Shafique, "Is spiking secure? A comparative study on the security vulnerabilities of spiking and deep neural networks," 2019, *arXiv:1902.01147*. [Online]. Available: http://arxiv.org/abs/1902.01147
[16] A. Bagheri, O. Simeone, and B. Rajendran, "Adversarial training for probabilistic spiking neural networks," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2018, pp. 1–5.
[17] S. Sharmin, P. Panda, S. Shakib Sarwar, C. Lee, W. Ponghiran, and K. Roy, "A comprehensive analysis on adversarial robustness of spiking neural networks," 2019, *arXiv:1905.02704*. [Online]. Available: http://arxiv.org/abs/1905.02704
[18] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.
[19] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, Mar. 2017.
[20] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.
[21] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal back propagation for training high-performance spiking neural networks," *Frontiers Neurosci.*, vol. 12, p. 331, May 2018.
[22] Y. Jin, W. Zhang, and P. Li, "Hybrid macro/micro level back propagation for training deep spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7005–7015.

[23] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 787–797.
[24] P. Gu, R. Xiao, G. Pan, and H. Tang, "STCA: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1366–1372.
[25] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.
[26] G. Orchard, A. Jayawant, G. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers Neurosci.*, vol. 9, p. 437, Nov. 2015.
[27] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "CIFAR10-DVS: An event-stream dataset for object classification," *Frontiers Neurosci.*, vol. 11, p. 309, May 2017.
[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[29] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009, Rep. No. 7.
[30] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networking Models Cognition*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[31] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Jan. 2008.
[32] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: http://arxiv.org/abs/1412.6572
[33] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*. [Online]. Available: http://arxiv.org/abs/1607.02533
[34] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.
[35] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.
[36] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.
[37] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017, *arXiv:1712.04248*. [Online]. Available: http://arxiv.org/abs/1712.04248
[38] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2017, *arXiv:1712.09665*. [Online]. Available: http://arxiv.org/abs/1712.09665
[39] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning models," 2017, *arXiv:1707.08945*. [Online]. Available: http://arxiv.org/abs/1707.08945
[40] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, Feb. 2018, pp. 1–17.
[41] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Oper. Syst. Princ.*, Oct. 2017, pp. 1–18.
[42] Y. Xiang Marcus Tan, Y. Elovici, and A. Binder, "Exploring the back alleys: Analysing the robustness of alternative neural network architectures against adversarial attacks," 2019, *arXiv:1912.03609*. [Online]. Available: http://arxiv.org/abs/1912.03609
[43] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers Neurosci.*, vol. 13, p. 95, 2019.
[44] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8529–8533.
[45] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2018.
[46] S. Reza Kheradpisheh and T. Masquelier, "S4NN: Temporal backpropagation for spiking neural networks with one spike per neuron," 2019, *arXiv:1910.09495*. [Online]. Available: http://arxiv.org/abs/1910.09495
[47] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?—A comprehensive study on the robustness of 18 deep image classification models," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 631–648.
[48] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7243–7252.

Ling Liang received the B.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2015, and the M.S. degree from the University of Southern California, Los Angeles, CA, USA, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA.

His current research interests include machine learning security, tensor computing, spiking neural networks, and computer architecture.

Xing Hu (Member, IEEE) received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2009, and the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2009 and 2014, respectively.

She is currently an Associate Professor with the State Key Laboratory of Computer Architecture, Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing. She has authored or coauthored over 40 refereed publications. Her current research interest includes deep learning system security and efficiency.

Dr. Hu was a PC Member of the IEEE/ACM International Symposium on Microarchitecture (MICRO) 2021 and the Design Automation Conference (DAC), and served as a Guest Editor for *Frontiers in Neuroscience*.

Lei Deng (Member, IEEE) received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2012, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2017.

He was a Post-Doctoral Fellow at the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA, from 2017 to 2021. He is currently an Assistant Professor at the Center for Brain Inspired Computing Research (CBICR), Tsinghua University. He has authored or coauthored over 70 refereed publications. His research interests span the areas of brain-inspired computing, machine learning, neuromorphic chip, and computer architecture.

Dr. Deng was a recipient of MIT Technology Review Innovators Under 35 China 2019. He was a PC Member of the International Joint Conference on Neural Networks (IJCNN) 2021 and the International Symposium on Neural Networks (ISNN) 2019, and served as a Guest Associate Editor for *Frontiers in Neuroscience* and *Frontiers in Computational Neuroscience*.

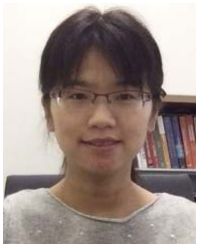
Yujie Wu received the B.E. degree from Lanzhou University, Lanzhou, China, in 2016, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2021.

He is currently a Post-Doctoral Fellow at the Institute of Theoretical Computer Science, Graz University of Technology, Graz, Austria. He has authored or coauthored over 20 refereed publications. His research interests include spiking neural networks, neuromorphic device, and brain-inspired computing.

Guoqi Li (Member, IEEE) received the B.E. degree from Xi'an University of Technology, Xi'an, China, in 2004, the M.E. degree from Xi'an Jiaotong University, Xi'an, in 2007, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2011.

He was a Scientist with the Data Storage Institute and the Institute of High Performance Computing, Agency for Science, Technology and Research (ASTAR), Singapore, from 2011 to 2014. He is currently an Associate Professor with the Center for Brain Inspired Computing Research (CBICR), Tsinghua University, Beijing, China. His current research interests include machine learning, brain-inspired computing, neuromorphic chip, complex systems, and system identification.

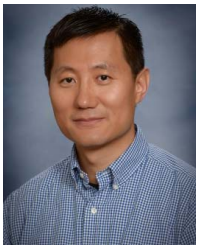
Dr. Li is an Editorial-Board Member of *Control and Decision* and a Guest Associate Editor of *Frontiers in Neuroscience* and Neuromorphic Engineering. He was a recipient of the 2018 First Class Prize in Science and Technology of the Chinese Institute of Command and Control, Best Paper Awards (*EAIS* 2012 and *NVMTS* 2015), and the 2018 Excellent Young Talent Award of the Beijing Natural Science Foundation.



Yufei Ding (Associate Member, IEEE) received the B.S. degree in physics from the University of Science and Technology of China, Hefei, China, in 2009, the M.S. degree from the College of William and Mary, Williamsburg, VA, USA, in 2011, and the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA, in 2017.

Since 2017, she has been with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, USA, as an Assistant Professor. Her research interest resides at the intersection of compiler technology and (big) data analytics, with a focus on enabling high-level program optimizations for data analytics and other data-intensive applications.

Dr. Ding was a receipt of the NCSU Computer Science Outstanding Research Award in 2016 and the Computer Science Outstanding Dissertation Award in 2018.



Peng Li (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2003.

He was a Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA, from 2004 to 2019. He is presently a Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. His research interests include integrated circuits and systems, computer-aided design, brain-inspired computing, and computational brain modeling.

Dr. Li's work has been recognized by various distinctions, including the International Conference on Computer Aided Design (ICCAD) Ten Year Retrospective Most Influential Paper Award, four IEEE/ACM Design Automation Conference Best Paper Awards, the IEEE/ACM William J. McCalla ICCAD Best Paper Award, the ISCAS Honorary Mention Best Paper Award from the Neural Systems and Applications Technical Committee of the IEEE Circuits and Systems Society, the U.S. National Science Foundation CAREER Award, two Inventor Recognition Awards from Microelectronics Advanced Research Corporation, two Semiconductor Research Corporation Inventor Recognition Awards, the William and Montine P. Head Fellow Award, and the TEES Fellow Award from the College of Engineering, Texas A&M University. He was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 2008 to 2013 and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS from 2008 to 2016, and he is currently a Guest Associate Editor of *Frontiers in Neuroscience*. He was the Vice President for the Technical Activities of the IEEE Council on Electronic Design Automation from 2016 to 2017.

Yuan Xie (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 1999 and 2002, respectively.

He was an Advisory Engineer with the IBM Microelectronic Division, Burlington, NJ, USA from 2002 to 2003. He was a Full Professor with Pennsylvania State University, Pennsylvania, PA, USA, from 2003 to 2014. He was a Visiting Researcher with the Interuniversity Microelectronics Centre (IMEC), Leuven, Belgium, from 2005 to 2007 and in 2010. He was a Senior Manager and Principal Researcher with the AMD Research China Lab, Beijing, from 2012 to 2013. He is currently a Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. His interests include VLSI design, electronics' design automation (EDA), computer architecture, and embedded systems.

Dr. Xie is an expert in computer architecture who has been inducted to the International Symposium on Computer Architecture (ISCA)/International Symposium on Microarchitecture (MICRO)/High-Performance Computer Architecture (HPCA) Hall of Fame and the AAAS/ACM Fellow. He was a recipient of best paper awards (HPCA 2015, the International Conference on Computer Aided Design (ICCAD) 2014, the Great Lakes Symposium on VLSI (GLSVLSI) 2014, the IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2012, the International Symposium on Low Power Electronics and Design (ISLPED) 2011, Asia and South Pacific Design Automation Conference (ASPDAC) 2008, and the Association of Surgeons of India (ASICON) 2001) and best paper nominations (ASPDAC 2014, MICRO 2013, the Design, Automation and Test in Europe Conference (DATE) 2013, ASPDAC 2009–2010, and ICCAD 2006), the 2016 IEEE Micro Top Picks Award, the 2008 IBM Faculty Award, and the 2006 NSF CAREER Award. He served as the TPC Chair for ICCAD 2019, HPCA 2018, ASPDAC 2013, ISLPED 2013, and the International Forum on Embedded MPSOC and Multicore (MPSOC) 2011, a Committee Member in the IEEE Design Automation Technical Committee (DATC), the Editor-in-Chief for the *ACM Journal on Emerging Technologies in Computing Systems*, and an Associate Editor for the *ACM Transactions on Design Automations for Electronics Systems*, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE DESIGN AND TEST OF COMPUTERS, and the *IET Computers and Design Techniques*. Through extensive collaboration with industry partners (e.g., AMD, HP, Honda, IBM, Intel, Google, Samsung, IMEC, Qualcomm, Alibaba, Seagate, and Toyota), he has helped the transition of research ideas to industry.