Adversarial attacks on spiking convolutional neural networks for event-based vision

Julian Büchel[1]*, Gregor Lenz[2], Yalun Hu[3], Sadique Sheik[2] and Martino Sorbaro[4,5]*

[1]IBM Research, Zurich, Switzerland, [2]SynSense, Zurich, Switzerland, [3]SynSense, Chengdu, China, [4]AI Center, ETH Zürich, Switzerland, [5]Institute of Neuroinformatics, University of Zürich, ETH, Zurich, Switzerland

Event-based dynamic vision sensors provide very sparse output in the form of spikes, which makes them suitable for low-power applications. Convolutional spiking neural networks model such event-based data and develop their full energy-saving potential when deployed on asynchronous neuromorphic hardware. Event-based vision being a nascent field, the sensitivity of spiking neural networks to potentially malicious adversarial attacks has received little attention so far. We show how white-box adversarial attack algorithms can be adapted to the discrete and sparse nature of event-based visual data, and demonstrate smaller perturbation magnitudes at higher success rates than the current state-of-the-art algorithms. For the first time, we also verify the effectiveness of these perturbations directly on neuromorphic hardware. Finally, we discuss the properties of the resulting perturbations, the effect of adversarial training as a defense strategy, and future directions.

KEYWORDS

spiking convolutional neural networks, adversarial examples, neuromorphic engineering, robust AI, dynamic vision sensors

1. Introduction

Compared to the neural networks commonly used in deep learning, Spiking Neural Network resemble the animal brain more closely in at least two main aspects: the way their neurons communicate through spikes, and their dynamics, which evolve in continuous time. Aside from offering more biologically plausible neuron models for computational neuroscience, research in the applications of Spiking Neural Network is currently blooming because of the rise of neuromorphic technology. Neuromorphic hardware is directly compatible with Spiking Neural Network and enables the design of low-power models for use in battery-operated, always-on devices.

Adversarial examples are an "intriguing property of neural networks" (Szegedy et al., 2013) by which the network is easily fooled into misclassifying an input which has been altered in an almost imperceptible way by the attacker. This property is usually undesirable in applications: it was proven, for example, that an adversarial attack may pose a threat to self-driving cars (Eykholt et al., 2018). Because of their relevance to real-world applications, a large amount of work has been published on this subject, typically following a pattern where new attacks are discovered, followed by new defense strategies, in turn followed by proof of other strategies that can still break through them (see Akhtar and Mian, 2018 for a review).

With the advent of real-world applications of spiking networks in neuromorphic devices, it is essential to make sure they work securely and reliably in a variety of contexts. In particular, there is a significant need for research on the possibility of adversarial attacks on neuromorphic hardware used for computer vision tasks. In this paper, we make an attempt at modifying event-based data, by adding and removing events, to generate adversarial examples that fool a spiking network deployed on a convolutional neuromorphic chip. This offers important insight into the reliability and security of neuromorphic vision devices, with important implications for commercial applications.

1.1. What is event-based sensing?

Event-based Dynamic Vision Sensor share characteristics with the mammalian retina and have several advantages over conventional, frame-based cameras:

- Camera output in the form of events and thus power consumption are directly driven by changes in the visual scene, omitting output completely in the case of a static scene.
- Pixels fire independently of each other which results in a stream of events at microsecond resolution instead of frames at fixed intervals. This enables very low latency and high dynamic range.

The sparse, asynchronous Dynamic Vision Sensor output does not suit current high-throughput, synchronous accelerators such as GPUs. To process event-based data efficiently, neuromorphic hardware is being developed, where neurons are only updated whenever they receive an event. Spiking neuromorphic systems include large-scale simulation of neuronal networks for neuroscience research (Furber et al., 2012) and low-power real-world deployments of machine learning algorithms. Spiking Convolutional Neural Network as well as conventional Convolutional Neural Network have been run on neuromorphic chips such as IBM's TrueNorth and HERMES (Esser et al., 2016; Khaddam-Aljameh et al., 2021), Intel's Loihi (Davies et al., 2018) and SynSense's Speck and Dynap-CNN (Liu et al., 2019) for low-power inference. The full pipeline of event-based sensors, stateful spiking neural networks, and asynchronous hardware—which is present in SynSense's Speck—allows for large gains in power efficiency compared to conventional systems.

1.2. Adversarial attacks on discrete data

The history of attack strategies against various kinds of machine learning algorithms pre-dates the advent of deep
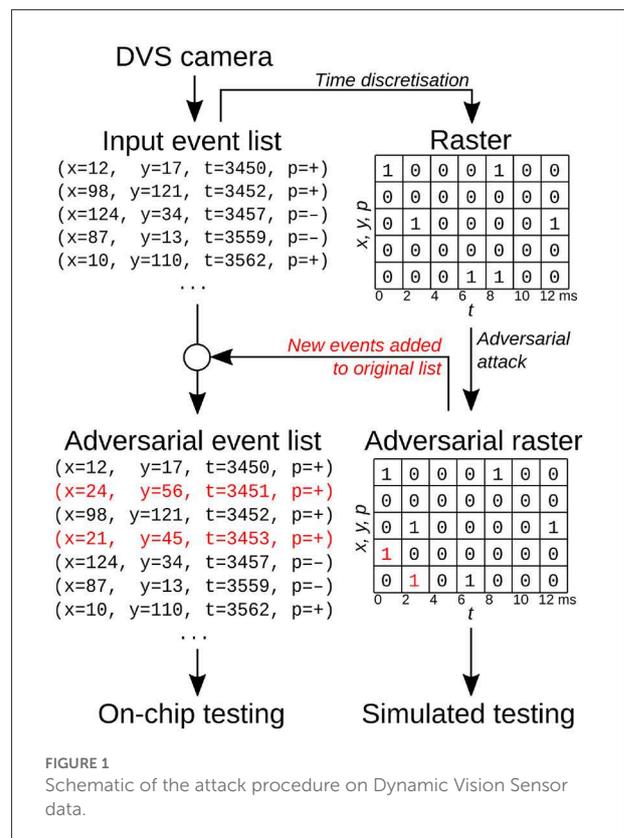
FIGURE 1
Schematic of the attack procedure on Dynamic Vision Sensor data.

learning (Biggio and Roli, 2018), but the phenomenon received widespread interest when adversarial examples were first found for deep convolutional networks (Szegedy et al., 2013). In general, given a neural network classifier C and an input x which is correctly classified, finding an adversarial perturbation means finding the smallest δ such that $C(x+\delta) \neq C(x)$. Here, "smallest" refers to minimizing $\|\delta\|$, where the norm is chosen arbitrarily depending on the requirements of the experiment. For example, using the L^∞ norm (maximum norm) will generally make the perturbation less noticeable to a human eye, while the use of the L^1 norm will encourage sparsity, i.e., a smaller number of perturbed pixels.

There are two main challenges in transferring existing adversarial algorithms to event-based vision:

- The presence of a continuous time dimension, as opposed to frames taken at fixed intervals;
- The binary discretization of input data and SNN activations, as opposed to traditional image data (at least 8 bit) and floating point network activations.

Event-based sensors encode information in the form of events that have a timestamp, location (x, y) and polarity (lighting increased or decreased). Because at any point in time an

event can either be triggered or not, one can simply view event-based inputs as binary data by discretizing time (Figure 1). In this view, the network's input is a three-dimensional array whose entries describe the number of events at a location (x, y) and in time bin t; an additional dimension, of size 2, is added due to the polarity of events. If the time discretization is sufficiently precise, and no more than one event appears in each voxel, the data can be treated as binary.

In this work, we present new algorithms that adapt the adversarial attacks SparseFool (Modas et al., 2018), and adversarial patches (Brown et al., 2017), to work with the time dynamics of spiking neural networks, and with the discrete nature of event-based data. We focus on the case of white box attacks, where the attacker has full access to the network and can backpropagate gradients through it. We test our attacks on the Neuromorphic MNIST (Orchard et al., 2015) and IBM Gestures (Amir et al., 2017) datasets, which are the most common benchmark datasets within the neuromorphic community. Importantly, for the first time, we also test the validity of our methods by deploying the attacks on neuromorphic hardware.

Our contributions can be summarized as follows:

- We contribute algorithms that adapt several adversarial attacks strategies to event-based data and Spiking Neural Network, with detailed results to quantify their effectiveness and scalability.
- We show that these adapted algorithms outperform current state-of-the-art algorithms in the domain of Spiking Convolutional Neural Network.
- We show targeted universal attacks on event-based data in the form of adversarial patches, which do not require prior knowledge of the input.
- We validate the resulting adversarial examples on an Spiking Neural Network deployed on a convolutional neuromorphic chip. To the best of our knowledge, this is the first time the effectiveness of adversarial examples is demonstrated directly on neuromorphic hardware.

2. Related work

Despite the growing number of Spiking Neural Network deployed on digital (Davies et al., 2018) and analog (Moradi et al., 2018) neuromorphic hardware, robustness to adversarial perturbations has received comparatively little attention by the research community. Some methods proposed for attacking binary inputs have focused on brute-force searches with heuristics to reduce the search space (Bagheri et al., 2018; Balkanski et al., 2020). Algorithms of this family do not scale well to large input sizes, as the number of queries made to the network grows exponentially. In particular, this becomes

a serious problem when the time dimension is added, greatly increasing the dimensionality of the input.

In Marchisio et al. (2021), the authors demonstrate various algorithms for attacking Dynamic Vision Sensor data that serves as input to an Spiking Neural Network. While some attacks are brute force ("Dash Attack," "Corner Attack") and therefore do not scale, "Frame Attack" simply adds events in hard-coded locations and produces perturbations that are orders of magnitudes larger than all of the perturbations of the algorithms presented here. The only algorithm that exploits gradient information is "Sparse Attack," against which we compare[1]. Unfortunately, the authors do not report quantitative results on the magnitudes of their perturbations (i.e., how many spikes are added or removed), which is an important metric for demonstrating efficiency of the attack.

Sharmin et al. (2020) demonstrate interesting properties of the inherent robustness of Spiking Neural Network to adversarial attacks, with variations depending on the training method. Their work, however, only uses static image data with continuous pixel values converted to Poisson spike input frequencies, which is rather different from working with actual event-based camera input.

As Liang et al. (2020) note, white-box attacks can exploit "surrogate" gradients in order to calculate gradients of a loss function with respect to the input in Spiking Neural Network. Not only do these gradients reflect the true dependency of the output to the input, but they also capture temporal information *via* Backpropagation Through Time. In their work, the authors use a Dirac function for their surrogate gradient, which causes, as they report, considerable "vanishing gradient" problems. While they show good results, they were forced to use additional tricks such as probabilistic sampling and an artificial construction of gradients termed "Restricted Spike Flipper" in their manuscript. In our work we solved the vanishing gradient issue by resorting to a more effective surrogate function (Section 3.3).

Outside of the domain of Spiking Neural Network, various studies explore the adversarial robustness of conventional neural networks deployed on digital accelerators or analog in-memory computing based accelerators. For example, Stutz et al. (2020) demonstrate that standard networks are susceptible to bit errors in the SRAM array storing the quantized weights in deep learning accelerators. The authors further show that one can mitigate this susceptibility *via* random bit flipping during training. While this work performs experiments using noise models obtained from various SRAM arrays, it still lacks a full-fledged hardware demonstration. Cherupally et al. (2021) obtain a noise model of a RRAM-based crossbar often found in analog in-memory computing architectures. The noise model is then

---

1 We followed the implementation at https://github.com/albertomarchisio/DVS-Attacks.

used in simulation to study the adversarial robustness of neural networks deployed on such a crossbar.

Besides attacking neural networks *via* input perturbation, a large body of work on directly attacking the physical hardware exists. Hardware fault injection attacks (see Giraud and Thiebeauld, 2004 for a survey) are mostly based on voltage glitching, electromagnetic pulses, or rapid row activation in DRAM (Kim et al., 2014).

In this work, we focus on input-based attacks, which we verify in simulations and on commercially available spiking neuromorphic hardware. This step is important to determine real-world effectiveness, as the exact simulation of asynchronous, event-based processing with microsecond resolution is prohibitively expensive on conventional hardware.

3. Methods

3.1. Attack strategies

3.1.1. SparseFool on discrete data

To operate on event-based data efficiently, the ideal adversarial algorithm requires two main properties: sparsity and scalability. *Scalability* is needed because of the increased dimensionality given by the additional time dimension. *Sparsity* ensures that the number of events added or removed is kept to a minimum. One approach that combines the above is SparseFool (Modas et al., 2018), which iteratively finds the closest point in L^2 on the decision boundary of the network using the DeepFool algorithm (Moosavi-Dezfooli et al., 2015) as a subroutine, followed by a linear solver that enforces sparsity and boundary constraints on the perturbation. DeepFool finds the smallest perturbation in L^2 by iteratively moving the input in the direction orthogonal to the linearized decision boundary around the current input. Since decision boundaries of neural networks are non-linear, this process has to be repeated until a misclassification is triggered. Because Spiking Neural Network have discrete outputs (the number of spikes over time for each output neuron), it is easier to suffer from vanishing gradients as the perturbation approaches the decision boundary. This occurs because DeepFool calculates the perturbation that just reaches the decision boundary, which is very small when the input is already close to the decision boundary. Therefore, adding this small perturbation to the input might not reflect in the number of emitted spikes as the membrane potential must cross the spiking threshold. We made the following changes to overcome these issues. Firstly, we clamped the perturbation at every iteration of DeepFool so that it was no smaller than a value η, in order to protect against vanishing gradients. η was treated as a hyperparameter that should be kept as small as possible without incurring vanishing gradients. Without η, SparseFool yields a success rate of 12.9% on 100 samples of the Neuromorphic MNIST dataset, where the success rate is the

percentage of attacked samples that led to a misclassification out of the samples that were originally classified correctly by the network. Secondly, to account for the discreteness of event-based data, we rounded the output of SparseFool to the nearest integer at each iteration. Finally, SparseFool normally involves upper and lower bounds l and u on pixel values (normally set, for images, to $l = 0; u = 255$). We exploit these to enforce the binary constraint on the data ($l = 0; u = 1$), or, in the on-chip experiments, to fix a maximum firing rate in each time bin, which is the same as that of the original input [$l = 0; u = \max(\text{input})$]. From now on, we will refer to this variant as SpikeFool.

3.1.2. Adversarial patches

As the name suggests, adversarial patches are perturbations that are accumulated in a certain region (patch) of the image. The idea is that these patches are generated in a way that enables the adversary to place them anywhere in the image. This attack is *targeted* to a desired label, and *universal*, i.e., not input-specific. To test a more realistic scenario where an adversary could potentially perform an attack without previous knowledge of the input, we apply these patches to the IBM hand gesture dataset. We note that the prediction of the Convolutional Neural Network trained on this dataset is mostly determined by spatial location of the input. For example, the original input of "Right Hand Wave" is not recognized as such if it is shifted or rotated by a substantial amount. In order to simulate effective realistic attacks, we choose to limit both computed and random attack patches to the area of where the actual gesture is performed. As in Brown et al. (2017), we generate the patches using Projected Gradient Descent on the log softmax value of the target output neuron. Projected Gradient Descent is performed iteratively on different images of the training set and the position of the patch is randomized after each sample. For each item in the training data, the algorithm updates the patch until the target label confidence has reached a pre-defined threshold. The algorithm skips the point if the original label equals the target label. This process is repeated for every training sample and for multiple epochs. To measure the effectiveness of our computed patches, we also generate random patches of the same size, and measure the target success rates. In a random patch, every pixel has a 50% chance of emitting a spike at each time step.

3.2. Datasets

Neuromorphic MNIST consists of 300 ms-long recordings of MNIST digits that are captured using a three-fold saccadic motion of a Dynamic Vision Sensor sensor (Orchard et al., 2015). We bin the events of each recording into 60 time steps, capping the maximum number of events to 1 per pixel.

TABLE 1 Result comparison between SpikeFool and related works.

	Attack strategy	Hardware	Success rate (%)	Median elapsed time (s/sample)	Median no. queries	Median L^0
N-MNIST	SpikeFool ($\eta = 0.2, \lambda = 2$)	GPU	99.53	24.87 ± 0.0815	45	**256**
	SpikeFool ($\eta = 0.5, \lambda = 2$)	GPU	99.88	12.00 ± 0.072	27	272
	Liang et al. (2020)	GPU	$\mathbf{100.00 \pm 0.00}$	$0.69 \pm 1 \cdot 10^{-4}$	**2**	865.3 ± 6.37
	Marchisio et al. (2021)	GPU	74.41 ± 0.33	$\mathbf{0.64 \pm 0.001}$	3	$3.23 \cdot 10^4 \pm 360.32$
IBM	SpikeFool ($\eta = 0.1, \lambda = 3$)	GPU	**100.00**	2.56 ± 0.003	11	311
	SpikeFool ($\eta = 0.1, \lambda = 2$)	GPU	99.87	2.4 ± 0.002	11	200
	SpikeFool ($\eta = 0.1, \lambda = 1$)	GPU	97.61	2.85 ± 0.02	17	**116**
	SpikeFool	Speck2b	88.05			
	Liang et al. (2020)	GPU	99.77 ± 0.126	0.53 ± 0.004	9	1345.5 ± 11.48
	Marchisio et al. (2021)	GPU	92.44 ± 0.05	$\mathbf{0.14 \pm 2 \cdot 10^{-4}}$	**2**	227343 ± 54.56

η indicates the minimum step size for updates to the perturbation: higher values of η find less precise perturbations (larger L^0 values), but are sometimes needed in order to prevent zero-gradient issues within the algorithm. λ is the sparsity parameter: lower λ (with a minimum of 1) yields sparser results, but gives a slightly lower success rate. The success rate is defined as the fraction of samples that were initially correctly classified, for which the attack algorithm converged to an adversarial example that the network classifies incorrectly. Attacks were repeated 5 times on 1000 samples. We report the mean and standard deviation for non-deterministic metrics. The median number of queries did not show any variation over 5 repetitions. Bold font indicates the best result for that column.

IBM Gestures dataset consists of recordings of 11 classes of human gestures, captured under three different lighting conditions (Amir et al., 2017). For this dataset, the model must have the ability to process temporal features to distinguish between clockwise and counterclockwise versions of the same gesture. We bin 200 ms slices of recordings into 20 frames each, again capping the frames to 1 per pixel. For experiments on the chip we choose a higher temporal resolution and bin the same 200 ms slices into 100 frames each.

3.3. Network models

The spiking networks used for the Neuromorphic MNIST and IBM Gestures tasks are trained using the publicly available PyTorch-based Spiking Neural Network library Sinabs[2] which supports the same non-leaky integrate-and-fire neurons available on the neuromorphic chip.

Models used for Neuromorphic MNIST were trained using *weight transfer*, whereby an equivalent Convolutional Neural Network is trained on accumulated frames (i.e., summing the data over the time dimension), and the Convolutional Neural Network weights are transferred to the spiking network (Rueckauer et al., 2017; Sorbaro et al., 2020). The model we used is a LeNet-5 architecture with 20, 32, 128, 500, and 10 channels, respectively. The network achieves 85.05%

classification test accuracy with full precision weights and 85.2% with 8-bit quantized weights.

For the IBM Gestures task, training is done using Backpropagation Through Time since for this dataset we have to learn temporal features as well. We make use of a surrogate gradient in the backwards pass to enable learning despite the discontinuous nature of spikes (Neftci et al., 2019). The model has a LeNet-5 architecture plus batchnorm layers with 8, 8, 8, 64, and 11 channels, respectively. This network achieves a classification test accuracy of 84.2%. The accuracy is retained if the network weights are quantized to 8 bits. The network used for the on-chip experiments does not have batch-normalization layers as they introduce biases which are not supported on the hardware.

3.4. Experiments on the neuromorphic chip

We used a multi-core spiking neural network processing chip prototype from SynSense, called "Speck2b." It uses asynchronous logic design to keep dynamic power that is consumed whenever spikes are routed to a minimum. That means that neurons update and spike only when an input is received, but are not limited to time bins of clock cycles, which is very different than conventional von Neumann hardware. The chip and its 327,000 neurons are fully configurable, supporting convolutional, linear, as well as pooling connectivity. Event input

---

2 http://sinabs.ai
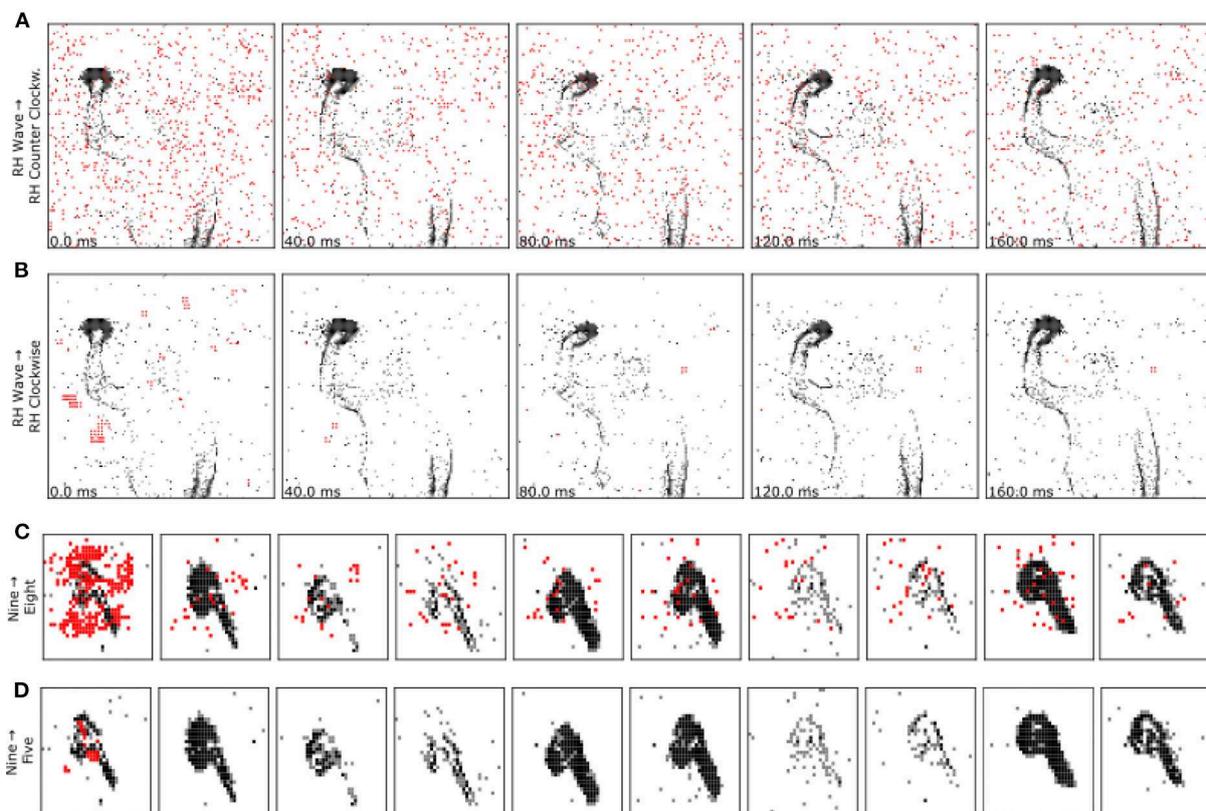
FIGURE 2
Our method **(B, D)** significantly reduces the magnitude of the perturbations introduced by the adversary. The runner-up method proposed in Liang et al. (2020) **(A, C)** adds visibly more events and is thus less stealthy.

(from datasets such as Neuromorphic MNIST and IBM gestures) is sent to the FPGA as a block of events in (t, p, x, y) format. The FPGA then handles the feeding to the chip at the correct timestamps. This is important as the chip itself, using fully asynchronous logic, does not understand the notion of time steps. Every neuron will compute its input on demand and fire as soon as it is ready. Since events have to be handled in a sequential manner, this can sometimes lead to slight changes in the order of input events, which then has consequences for neuron activation further downstream (in later layers). If many events arrive in a short time span, they are buffered depending on the throughput limitations of each core. Those limitations together with weight quantization on the neuromorphic hardware lead to slight differences in output when compared to simulation.

The multi-core, single-chip neuromorphic processor we used for the experiments features an integrated DVS sensor for real-time, fully integrated vision and is commercially available for programming as part of a development kit, which includes USB connectivity, power management and an FPGA for communication. Since the chip features an on-chip DVS sensor, its main application is in the vision domain, making it even more relevant for our study.

On the chip, weight precision, number of computations per second and throughput are reduced, as the hardware is optimized for very low power consumption. The networks detailed in the previous sections have to be modified in order to make them suitable for on-chip inference: their weights are rescaled and discretized as required by the chip's 8-bit weight precision. This is one factor that can lead to a degradation in prediction accuracy when compared to simulations. A second and more important factor why simulations do not mimic our chip's behavior exactly is the need for time discretization when training our networks off-chip (see Figure 1).

As this work focuses on white-box attacks, we first compute the adversarial examples using the network simulation on the computer, and then test both original and attacked spiketrains in simulation and on the chip. However, the simulation and the attack work in discrete time, while the chip receives events in continuous time. In order to convert the discrete-time attacked raster back to a list of events for the chip, we compare the original and attacked rasters, identifying new events added by the attack and adding them to the original list (Figure 1). We empirically found very few events were removed by SparseFool; for example, in the Neuromorphic MNIST experiment, there

FIGURE 3
Examples of adversarial inputs on the, Neuromorphic MNIST **(top)** and IBM Gestures **(bottom)** datasets, as obtained by the SpikeFool method. The captions show the original (true) label, correctly identified, and the class later identified by the model. The data was re-framed in time for convenience of visualization. Red indicates added spikes. See the Supplementary Video for more examples and motion visualization.

were 28 additions per each removal (see Supplementary material for more information). For simplicity, we therefore chose to ignore removals for on-chip experiments.

4. Results

4.1. Adversarial attack performance

Table 1 shows the comparison of SpikeFool with current state-of-the-art attack algorithms in the spiking domain on two widely used benchmarks. We compare various metrics such as success rate, median elapsed time per sample, and, most importantly, the median perturbation size. Results from Marchisio et al. (2021) and Liang et al. (2020) are drawn from our own implementations, closely following published code (if available) from the authors.

We find that the attack presented in Marchisio et al. (2021) largely fails to achieve reasonable perturbation magnitudes on both benchmarks. This is likely because of the lack of bounds and rounding in their method, which result in large deviations. In contrast, the attack presented in Liang et al. (2020) achieves near perfect success rates in very short time at relatively low perturbation sizes. However, this speed comes at the cost of perturbation size: our method yields perturbations that are up to $11\times$ smaller than the ones generated by Liang et al. (2020). To put these numbers into perspective, Figure 2 compares the adversarial samples generated by both methods. One can observe the clear difference between the two methods and could argue that attacks generated by Liang et al. (2020) are more visible. Interestingly, we observe that SpikeFool often resorts to inserting small patches limited to key areas. We believe this is due to the fact that the network is inherently robust to salt-and-pepper noise and that introducing localized patches is by far more effective. Figure 3 and the Supplementary Video show

additional examples of successful attacks. Further information about how many spikes are added and removed during training can be found in the Supplementary material.

We also run a random subset of 777 DVS Gesture samples that were attacked using SpikeFool ($\lambda = 2.0, \eta = 0.3$) on our neuromorphic hardware and observe a success rate of 88.05% misclassified samples, in comparison to 93.28% in simulation. This performance is lower compared to the one reported in Table 1 due to the much higher time resolution needed on chip, which makes it harder to find an attack. A possible reason for this discrepancy lies in how the chip is limited in computing capacity by weight precision and restricted throughput per time unit, which causes some of the input events to be dropped. Furthermore, the conversion of binned data back into lists of spikes is necessarily lossy. In terms of attack efficiency, we observe a median difference in number of spikes of 903 among the attacks that were successful on chip, corresponding to a median 9.3% increase in the number of spikes per sample.

4.2. Adversarial patches

Although we have demonstrated that one can achieve high success rates on custom spiking hardware that operates with microsecond precision, the applicability of this method is still limited, as the adversary needs to suppress and add events at high spatial and temporal resolution, thus making the assumption that the adversary can modify the event-stream coming from the Dynamic Vision Sensor camera. Furthermore, SpikeFool assumes knowledge of the model and requires computing the perturbation offline, which is not feasible in a timely manner. In a more realistic setting, the adversary is assumed to generate perturbations by changing the input the Dynamic Vision Sensor camera receives on the fly, by e.g., adding a light-emitting device to the visual input of the Dynamic Vision Sensor camera.

Using the training data from the IBM Gestures dataset, we generate an adversarial patch for each target class with high temporal precision (event samples of 200 ms are binned using 0.5 ms-wide bins) and evaluate the effectiveness in triggering a targeted misclassification both in simulation and on-chip using the test data. To simulate spatial imprecision during deployment, each test sample is perturbed by a patch that was randomly placed within the area of the original gesture. Table 2 summarizes our findings on target success rates for generated and random patches. Simulated results show high success rates, and on-chip performance shows a slight degradation, which can be expected due to weight quantization on the tested specialized hardware. We also find that the chip has trouble processing inputs because most of the added patch events occur concentrated in the beginning of recordings in a large transient peak. In one case, the targeted attack for label "Arm Roll" mostly fails on chip as not all events are processed, which makes it harder to differentiate from "Hand Clap," a similar gesture

TABLE 2 Adversarial patches for different target labels were evaluated on—and off—chip.

Target label	Hand clap	RH wave	LH wave	RH clockwise	RH counter clockwise	LH clockwise	LH counter clockwise	Arm roll	Air drum	Air guitar	Other
Adversarial patch	90.3	99.0	89.8	87.3	79.7	49.7	51.5	63.6	79.1	92.3	64.7
Adv. patch (on-chip)	94.0	89.0	94.1	81.3	65.1	35.9	43.8	5.0	82.7	87.3	66.8
Random patch	18.8	80.7	77.0	0	0	3.6	0.6	0	0	12.6	16.6
Rand. patch (on-chip)	43	76.8	72.2	0	0	9.0	2.4	0	0	0	17.7

Shown here are the success rates in percent for each target label. An attack is considered successful if the original label is not the target label and the network predicts the target label when the patch is applied.

that occurs in the same central spatial location. This could be mitigated by limiting the number of events in a patch to ensure that they could all be correctly processed on the chip.

We compare this result with a baseline of randomly generated patches, and we observe that two labels, namely "Left" and "Right Hand Wave" subsume all other attacked labels in this case. This hints that randomly injecting events in various locations is not enough to perturb network prediction to a desired label and that our algorithm succeeds in finding a meaningful patch. We find that generating patches with lower temporal resolution heavily degrades performance on-chip, as the chip operates using microsecond precision. We also perform ablation studies where we move the patch far from the position where the gesture is expected and observe that the patch mostly triggers a different misclassification, mostly the gesture that is expected at the new position. We find that this observation originates from the fact that the network heavily relies on the spatial location of the gesture and one should, if the hardware allows, consider larger networks that are invariant to the position of the gesture. To summarize, adversarial patches are effective in triggering a targeted misclassification both on– and off–chip compared to randomly generated ones. Figure 4 and the Supplementary Video show examples of successful patch attacks. Importantly, these attacks are universal, meaning that they can be applied to any input and do not need to be generated for each sample.

4.3. Defense *via* adversarial training

Once it is known that a model or system is sensitive to a certain type of adversarial attack, it is natural to investigate whether there is a way to build a network that is more resistant to these attacks. We therefore experimented with adversarial training using the TRadeoff-inspired Adversarial Defense *via* Surrogate-loss minimization (TRADES) method (Zhang et al., 2019). The method adds a new term to the loss function, which minimizes the Kullback-Leibler divergence between the output of the network on the original input and the output when the adversarial example is presented:

$$\mathcal{L}_{\text{rob}} = \mathcal{L} + \frac{\beta_{\text{rob}}}{B} \, \text{D}_{\text{KL}}(f(\mathbf{x}_{\text{adv}}); f(\mathbf{x}_0)).$$

Here, B is the batch size, β_{rob} is the parameter that defines the trade-off between robustness and accuracy, f is the network and $\mathbf{x}_{\text{adv}}$ is the adversarial input. To find $\mathbf{x}_{\text{adv}}$ at training time, we use a Projected Gradient Descent-based attack, since it can be batched—but we attack the resulting networks using SpikeFool at test time.

More specifically, we use Projected Gradient Descent in the L^{∞} domain and choose $\epsilon = 0.5$ as the maximum perturbation, with $N_{\text{pgd}} = 5$ attack steps. We use Projected Gradient Descent in the spiking domain by accumulating

TABLE 3 Training with a small robustness term ($\beta_{\text{rob}} = 0.01$) increases generalization and therefore improves the test accuracy of the network.

	Network	Test accuracy (%)	Success rate (%)	Median L^0
	$\beta_{\text{rob}} = 0.0$ (normal)	79.23	98.19	438.0
	$\beta_{\text{rob}} = 0.01$	**81.36**	**98.78**	511.0
IBM	$\beta_{\text{rob}} = 0.05$	75.27	97.65	**1434.0**
	$\beta_{\text{rob}} = 0.1$	73.16	95.46	1316.0

For increasing values of β_{rob}, the amount of changed events needed to trigger a misclassification increases dramatically (more than 3×), while the performance of the network decreases (hence the name "tradeoff-inspired"). Bold font indicates the best result for that column.

the gradients in full-precision using a straight-through estimator (Bengio et al., 2013). The details of our spiking-adapted implementation of Projected Gradient Descent are described in the Supplementary material, including extensions, and a full comparison with SpikeFool.

Although we use a much simpler attack strategy at training time, we found that it produced perturbations of reasonable sizes while being extremely efficient and sufficiently effective: for the choices of β_{rob} we considered, we see that the adversarially-trained network requires stronger and less stealthy attacks before it is fooled (Figure 5). As expected, SparseFool's success rate is still high, since it aims to find a solution no matter the costs; but there is an increase in the number of added spikes required, which is indeed a sign of robustness (Table 3). Further work is required for a comprehensive investigation of other defense strategies.

5. Discussion

We studied the possibility of fooling Spiking Neural Network through adversarial perturbations to Dynamic Vision Sensor data, and verified these perturbations on a spiking convolutional neuromorphic chip. There were two main challenges to this endeavor: the discrete nature of event-based data, and their dependence on time. Dynamic Vision Sensor attacks also have different sparsity requirements, because the magnitude of the perturbation is measured in terms of number of events added or removed. For this purpose, we adopted a surrogate-gradient method and backpropagation-through-time to perform white-box attacks on spiking networks. We presented SpikeFool, and adapted version of SparseFool, which we compared to current state-of-the-art methods on well-known benchmarks. We find that SpikeFool achieves near perfect success rates at lowest perturbation magnitudes on time-discretized samples of the Neuromorphic MNIST and IBM Gestures datasets. In the best cases, the attack requires the addition of less than a hundred events over 200 ms. To

FIGURE 4

Examples of adversarial patches successfully applied to a single "right hand clockwise" data sample, with different target classes. See also the Supplementary Video for motion visualization and more examples.

FIGURE 5

Adversarial training visibly increases the number of events that need to be added/removed in order to trigger a misclassification.
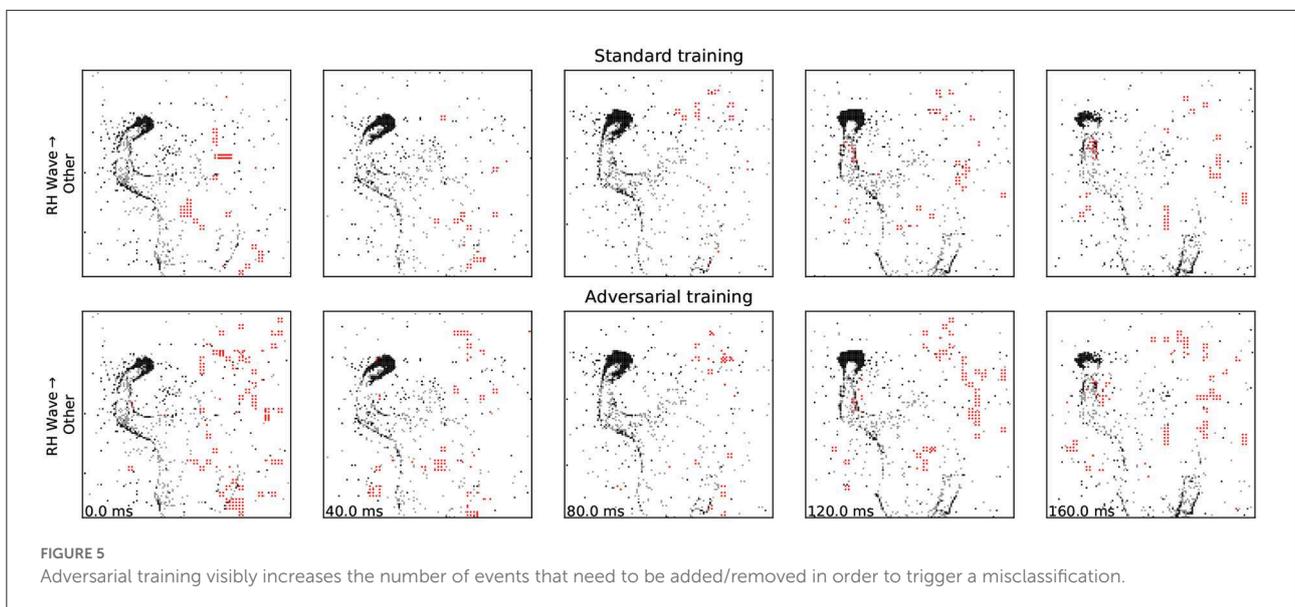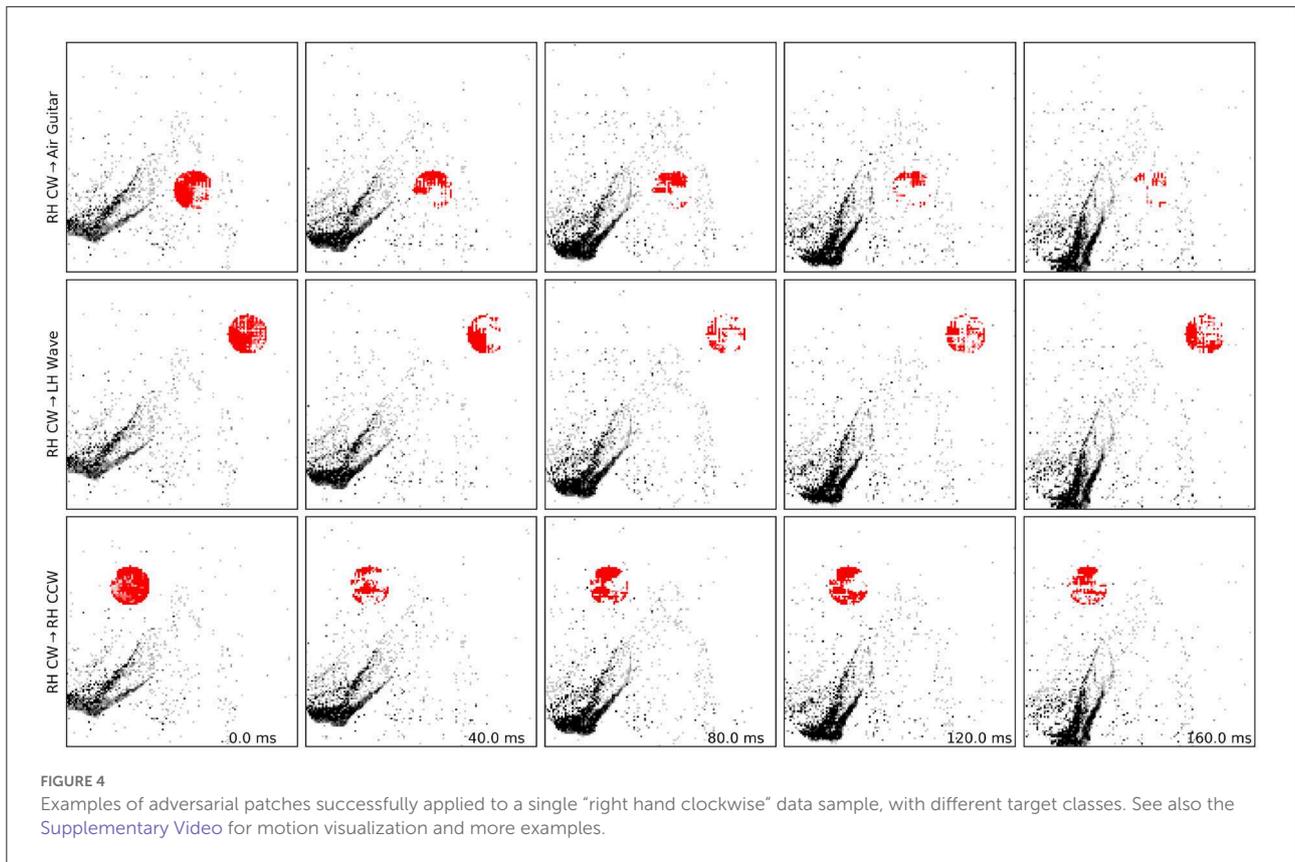
the best of our knowledge, we were also the first to show that the perturbation is effective on a network deployed on a neuromorphic chip, implying that the method is resilient to the small but non-trivial mismatch between simulated and deployed networks.

Additionally, since SpikeFool computes perturbations offline and not on a live stream of Dynamic Vision Sensor events, we also investigated a more realistic setting, where an adversary can inject spurious events in the form of a patch inserted into the visual field of the Dynamic Vision Sensor

camera. We demonstrated that we can generate patches for different target labels. Although these patches require a much higher amount of added events, they do not require prior knowledge of the input sample and therefore offer a realistic way of fooling deployed neuromorphic systems. A natural next step would be to understand whether it is possible to build real-world patches that can fool the system from a variety of distances and orientations, as Eykholt et al. (2018) did for photographs. Moreover, it will be interesting to see how important knowledge about the architecture is and if one can generate patches by having access to a network that differs from the one deployed.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: https://research.ibm.com/interactive/dvsgesture/ and https://www.garrickorchard.com/datasets/n-mnist/.

Author contributions

JB and MS conceived and designed the research. JB, MS, GL, and YH performed experiments, collected data, analyzed, and interpreted the data. JB, MS, and GL drafted the manuscript. All authors developed software and simulations, performed critical revision of the manuscript, and approved the final version for publication.

Funding

Acknowledgments

Conflict of interest

Author JB is employed by IBM. Authors GL, YH, and SS are employed by SynSense.

The remaining author declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2022.1068193/full#supplementary-material

References

Akhtar, N., and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: a survey. *IEEE Access* 6, 14410–14430. doi: 10.1109/ACCESS.2018.2807385

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., et al. (2017). "A low power, fully event-based gesture recognition system," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 7388–7397.

Bagheri, A., Simeone, O., and Rajendran, B. (2018). "Adversarial training for probabilistic spiking neural networks," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)* (Kalamata: IEEE).

Balkanski, E., Chase, H., Oshiba, K., Rilee, A., Singer, Y., and Wang, R. (2020). Adversarial attacks on binary image recognition systems. *CoRR*, abs/2010.11782. doi: 10.48550/arXiv.2010.11782

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432 [cs.LG]*. doi: 10.48550/arXiv.1308.3432

Biggio, B., and Roli, F. (2018). Wild patterns: ten years after the rise of adversarial machine learning. *Pattern Recognit.* 84, 317–331. doi: 10.1016/j.patcog.2018.07.023

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. (2017). Adversarial patch. *CoRR*, abs/1712.09665. doi: 10.48550/arXiv.1712.09665

Cherupally, S. K., Meng, J., Rakin, A. S., Yin, S., Yeo, I., Yu, S., et al. (2021). Improving the accuracy and robustness of rram-based in-memory computing against rram hardware noise and adversarial attacks. *Semiconduct. Sci. Technol.* 37, ac461f. doi: 10.1088/1361-6641/ac461f

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., et al. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc. Natl. Acad. Sci. U.S.A.* 113, 11441–11446. doi: 10.1073/pnas.1604850113

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., et al. (2018). "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 1625–1634.

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., et al. (2012). Overview of the spinnaker system architecture. *IEEE Trans. Comput.* 62, 2454–2467. doi: 10.1109/TC.2012.142

Giraud, C., and Thiebeauld, H. (2004). "A survey on fault attacks," in *Smart Card Research and Advanced Applications VI*, eds J.-J. Quisquater, P. Paradinas, Y. Deswarte, and A. A. El Kalam (Boston, MA: Springer U.S.), 159–176.

Khaddam-Aljameh, R., Stanisavljevic, M., Mas, J. F., Karunaratne, G., Braendli, M., Liu, F., et al. (2021). "Hermes core-a 14nm cmos and pcm-based in-memory compute core using an array of 300ps/lsb linearized cco-based adcs and local digital processing," in *2021 Symposium on VLSI Technology*, 1–2.

Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J. H., Lee, D., et al. (2014). Flipping bits in memory without accessing them: an experimental study of dram disturbance errors. *SIGARCH Comput. Archit. News* 42, 361–372. doi: 10.1145/2678373.2665726

Liang, L., Hu, X., Deng, L., Wu, Y., Li, G., Ding, Y., et al. (2020). Exploring adversarial attack in spiking neural networks with spike-compatible gradient. *CoRR*, abs/2001.01587. doi: 10.1109/TNNLS.2021.3106961

Liu, Q., Richter, O., Nielsen, C., Sheik, S., Indiveri, G., and Qiao, N. (2019). "Live demonstration: face recognition on an ultra-low power event-driven convolutional neural network asic," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (Long Beach, CA: IEEE).

Marchisio, A., Pira, G., Martina, M., Masera, G., and Shafique, M. (2021). "DVS-attacks: adversarial attacks on dynamic vision sensors for spiking neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)* (Shenzhen: IEEE), 1–9.

Modas, A., Moosavi-Dezfooli, S., and Frossard, P. (2018). Sparsefool: a few pixels make a big difference. *CoRR*, abs/1811.02248. doi: 10.1109/CVPR.2019.00930

Moosavi-Dezfooli, S., Fawzi, A., and Frossard, P. (2015). Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599. doi: 10.1109/CVPR.2016.282

Moradi, S., Qiao, N., Stefanini, F., and Indiveri, G. (2018). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Trans. Biomed. Circ. Syst.* 12, 106–122. doi: 10.1109/TBCAS.2017.2759700

Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* 36, 51–63. doi: 10.1109/MSP.2019.2931595

Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* 9, 437. doi: 10.3389/fnins.2015.00437

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11, 682. doi: 10.3389/fnins.2017.00682

Sharmin, S., Rathi, N., Panda, P., and Roy, K. (2020). "Inherent adversarial robustness of deep spiking neural networks: effects of discrete input encoding and non-linear activations," in *European Conference on Computer Vision* (Springer), 399–414.

Sorbaro, M., Liu, Q., Bortone, M., and Sheik, S. (2020). Optimizing the energy consumption of spiking neural networks for neuromorphic applications. *Front. Neurosci.* 14, 662. doi: 10.3389/fnins.2020.00662

Stutz, D., Chandramoorthy, N., Hein, M., and Schiele, B. (2020). Bit error robustness for energy-efficient DNN accelerators. *arXiv:2006.13977 [cs.LG]*. doi: 10.48550/arXiv.2006.13977

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., et al. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*. doi: 10.48550/arXiv.1312.6199

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. (2019). "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning* (PMLR), 7472–7482.