SpikingJET: Enhancing Fault Injection for Fully and Convolutional Spiking Neural Networks

SpikingJET: Enhancing Fault Injection for Fully and Convolutional Spiking Neural Networks

Anıl Bayram Gögebakan, Enrico Magliano, Alessio Carpegna, Annachiara Ruospo,
Alessandro Savino, Stefano Di Carlo
Control and Computer Department
Politecnico di Torino
Torino, Italy
Contact: stefano.dicarlo@polito.it

***Abstract*—As artificial neural networks have become increasingly integrated into safety-critical systems such as autonomous vehicles, devices for medical diagnosis, and industrial automation, ensuring their reliability in the face of random hardware faults becomes paramount. This paper introduces SpikingJET, a novel fault injector designed specifically for fully connected and convolutional Spiking Neural Networks (SNNs). Our work underscores the critical need to evaluate the resilience of SNNs to hardware faults, considering their growing prominence in real-world applications. SpikingJET provides a comprehensive platform for assessing the resilience of SNNs by inducing errors and injecting faults into critical components such as synaptic weights, neuron model parameters, internal states, and activation functions. This paper demonstrates the effectiveness of SpikingJET through extensive software-level experiments on various SNN architectures, revealing insights into their vulnerability and resilience to hardware faults. Moreover, highlighting the importance of fault resilience in SNNs contributes to the ongoing effort to enhance the reliability and safety of Neural Network (NN)-powered systems in diverse domains.**

***Index Terms*—Reliability, Fault Injection, Spiking neural networks, AI Resilience**

I. Introduction

Artificial Intelligence (AI), and in particular the field of Deep Learning (DL) and Deep Neural Networks (DNNs), is gradually changing the research and industrial landscape, allowing to solve complex problems with unprecedented precision, reaching human capabilities in areas such as image classification [1]. This makes it particularly attractive in various fields, including safety-critical applications like avionics, automotive, robotics, and healthcare.

Spiking Neural Network (SNN) emerge as an energy-efficient solution for resource and power-constrained applications in this landscape [2, 3]. They take direct inspiration from neuroscience observations, operating on asynchronous events resembling the action potentials, or spikes, used by biological neurons to communicate. Consequently, they excel at the real-time processing of dynamic, time-sensitive data in edge-computing scenarios.

Nonetheless, to deploy SNNs in safety-critical systems, it is crucial to assess their reliability since a wrong decision can harm individuals or the system itself. One of the most commonly used and effective approaches to evaluate the robustness of DNN-based applications is to perform extensive fault-injection campaigns, introducing faults in the system during its inference operation, gathering results, and comparing them with a fault-free execution, to assess the impact of the faults on the system's performance. While there are previous investigations into the overall effects of faults on individual spiking neurons [4], as well as comprehensive studies on full SNNs [5, 6], they tend to investigate the reliability of specific neuron models, network architectures, or fault mechanisms. Furthermore, existing fault injectors [7], although theoretically applicable to SNNs, are primarily evaluated using traditional DNN models. Hence, there is still the need for an extensive and flexible Fault Injector (FI), specifically tailored and validated for use with SNNs, capable of assessing various fault types across different network architectures, neuron models, and applications.

This paper proposes SpikingJET, a simulation-based, non-intrusive FI for SNNs, built on top of the open-source `SnnTorch` [8] framework and developed as an extension of the tool proposed in [9], allowing the user to perform extensive campaigns, injecting stuck-at faults in static model parameters, such as synaptic weights, and neuron's internal parameters, taking into account specific SNN features. Additionally, SpikingJET can inject faults into dynamic state variables and activation functions, ensuring the persistence of stuck-at faults throughout the inference process. An extensive study of fault injection on SNNs through SpikingJET is presented to demonstrate the tool's effectiveness. The injection campaign targets three SNN models, evaluated on as many benchmark datasets: N-MNIST [10], Spiking Heildelberg Digits (SHD) [11] and Dynamic Vision Sensor (DVS) 128 gestures [12], considering Convolutional, Feed Forward (FF) and recurrent Fully Connected (FC) architectures.

The paper is structured as follows: Section III reviews state-of-the-art solutions for fault tolerance in SNNs. Section II briefly overviews SNN principles and fault injection tech-

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101070238. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

979-8-3503-7055-3/24/$31.00 ©2024 IEEE

niques. Section IV outlines methodologies and design choices. Section V details fault-injection campaigns with SpikingJET and discusses outcomes. Finally, Section VI summarizes the study.

II. Background

In the field of Artificial Neural Networks (ANNs), SNNs stand out for their closer proximity to neuroscience models. First, neurons exchange information in the form of sequences of asynchronous spikes. The most common approach is to encode information with spike timing without focusing on their specific shape [13]. In the digital realm, spikes are represented as single-bit binary events: either "present" (1) or "not present" (0). This simplifies complexity from spatial to temporal dimensions, reducing computational resources for static data like images. For temporally evolving data, appropriate encoding captures essential information, optimizing computations. This makes SNNs particularly intriguing for resource-limited and power-constrained applications while capturing the temporal dynamics. Among the computational models developed in the last decades, this work targets the Leaky Integrated and Fire (LIF) approach [14], as it balances computational efficiency, temporal information capture, and biological fidelity. The discrete-time characteristic equation of the model is reported in Equation 1.

$$V_m[n] = \begin{cases} \beta \cdot V_m[n-1] + W \cdot s_{in}[n], & \text{if } V_m[n-1] \leq V_{th} \\ V_m[n-1] - V_{th} + W \cdot s_{in}[n], & \text{if } V_m[n-1] > V_{th} \end{cases} \quad (1)$$

The model's core is V_m, which emulates the membrane potential of a biological neuron. To draw a parallel with artificial neurons, two subsequent operations can be identified: (i) a linear combination of the inputs, here expressed as a dot product between the synaptic weights W and the input spikes $s_{in}[n]$ to identify a weighted sum of the inputs. For different models, like Convolutional Spiking Neural Network (CSNN), this can be a more complex operation, like a convolution. The result can be seen as the net input current received by the neuron. (ii) A non-linear function. Unlike traditional activation functions such as ReLU or sigmoid functions, the non-linearity is applied directly to the state variable V_m, which retains previous information ($V_m[n-1]$) and integrates the weighted sum. Finally, a threshold V_{th} separates two working regimes: in sub-threshold, the membrane potential is iteratively decayed ($\beta \cdot V_m[n-1]$, $\beta < 1$), following an exponential trend. If instead V_m exceeds the threshold, it is reset by decreasing it to a quantity equal to V_{th}. When this happens, an output spike s_{out} is generated and sent to the subsequent units, as shown in Equation 2

$$s_{out}[n] = \begin{cases} 1, & \text{if } V_m[n-1] > V_{th} \\ 0, & \text{if } V_m[n-1] \leq V_{th} \end{cases} \quad (2)$$

Looking at the model from a fault-injection perspective, there are 5 possible points of fault: (i) the input linear operation involving synaptic weights W and, if present, a bias value, (ii) the decay factor β, (iii) the threshold value V_{th}, (iv) the state variable V_m and (v) the activation function s_{out}. s_{in} is inherently considered as it represents the inputs originating from a preceding layer, constituting the ensemble of s_{out} from that layer.

When assessing the resilience of a Neural Network (NN) model, the goal is to understand the impact of hardware faults on the model inference. This can be done by artificially injecting faults into the system and then comparing the outcome of the inference with the fault-free correct output, generally called a golden reference. Fault-injection can be performed at different levels of abstraction, namely at the system level, considering the hardware platform on which the NN is executed, or at the application level, targeting a technology-independent model implementation. Faults in electronic devices can be classified based on their temporal nature: in transient bit-flips, an individual memory cell undergoes a local state change, which is correctly restored the first time the cell is written again. Conversely, permanent or stuck-at faults represent irreversible physical damage to the device, in which individual electronic elements are tied to a logical state (0 or 1) [15]. Three primary methods are employed for fault injection [16, 17]. Simulation-based techniques involve modeling various fault mechanisms using software simulations or system emulation. Platform-based FI involves manipulating the target hardware or software directly to induce faults. Lastly, radiation-based methods expose electronic devices to sources like alpha particles, neutrons, or electromagnetic radiation. As techniques approach the target hardware, faults are simulated more accurately, but complexity and control decrease. Therefore, simulation-level injection is ideal for broad, repeatable, and controllable assessment of system resiliency.

To address the increasing costs associated with fault injection procedures, Statistical Fault Injections (SFIs) are today considered a valid solution to estimate a specific characteristic of a population of faults by observing only a reduced portion (i.e., a sample). This statistical measurement will introduce an error (i.e., error margin), meaning that the characteristic of the population being estimated will be close to the estimate by plus or minus the margin of error, which will happen with a specific confidence level. Hence, in a SFI, the Fault List (FL) includes a random sample of faults that is representative of the entire fault universe, and, given a specific error margin (e) and confidence level (t), the number (n) of injected fault is defined as in Equation 3.

$$n = \frac{N}{1 + e^2 \cdot \frac{N-1}{t^2 \cdot p * (1-p)}} \quad (3)$$

In Equation 3, N is the size of the total population of faults (this number depends on the adopted fault model), and p is the probability of success. Adopting an equal probability of success or failure is a conservative solution. Hence, p is equal to 0.5. It is worth underlining that SFI techniques have been widely applied in the literature for estimating the vulnerability of DNNs at different granularity (e.g., to find

out the most critical layer, kernel, weight, or bit position). In [18], the effectiveness of SFIs in estimating failure rates (e.g., the number of wrong predictions caused by faults) has been experimentally validated with Convolutional Neural Networks (CNNs). The statistical methodology shows that by injecting only the 1.21% (ResNet-20) or 0.55% (MobileNetV2) of the entire possible experiments, it is possible to achieve an estimate of the CNN reliability close to the exhaustive result with an error always lower than 1%.

III. Related works

Fault injection in SNNs can be approached in different ways [19]. One common strategy is to model faults starting from typical hardware implementations of spiking neurons [20], trying to understand the possible faults that can incur and simulating them [4]. Functionally, these faults are typically classified into three categories: (i) dead neurons, (ii) saturated neurons (where the output spike consistently remains at 0 or 1, respectively, regardless of input), and (iii) timing variations (resulting from alterations in internal parameters affecting spike timing). The first two categories often stem from stuck-at faults, where the output spike or threshold becomes fixed at specific logical values, or faults in the membrane perpetually hold the neuron above or below the threshold, causing continuous firing or complete inhibition. Conversely, the third category encompasses other fault mechanisms, including milder stuck-at faults in the internal parameters or deviations in membrane behavior. Despite existing efforts, there remains a gap in addressing diverse fault types comprehensively within SNNs. Current injection methods often target specific hardware implementations or components, such as power supply [5, 6], or hardware memory [21], with limited focus on emerging hardware designs [22]). Although some studies consider general resiliency to noise in spiking layers [23], software-based FIs, while allegedly compatible with SNNs, primarily focus their analysis on more traditional models like DNNs [7], leaving gaps in comprehensive fault assessment and resiliency testing across various architectures, neuron models, and applications.

IV. Proposed Approach

Figure 1 shows the main building blocks of the proposed SpikingJET framework. The tool consists of two primary components: the Fault List Generator (FLG) and the Fault Injection Manager (FIM), each playing a distinct role in the fault injection process. Specifically, the FLG compiles the list of faults to be injected, enumerating them within the FL. Subsequently, the FL is passed to the FIM, which utilizes it to execute the actual fault injection campaign. Finally, the results of the faulty runs are compared to the golden reference to assess the impact of each fault on the output accuracy. Let's delve deeper into the details of the process. It is crucial to note that the fault injection campaign occurs during the inference phase, necessitating the user's provision of a fully pre-trained model. Initially, this model (Golden Ref. in Figure 1) generates a fault-free reference, termed the golden reference, encompassing predictions for all input data.

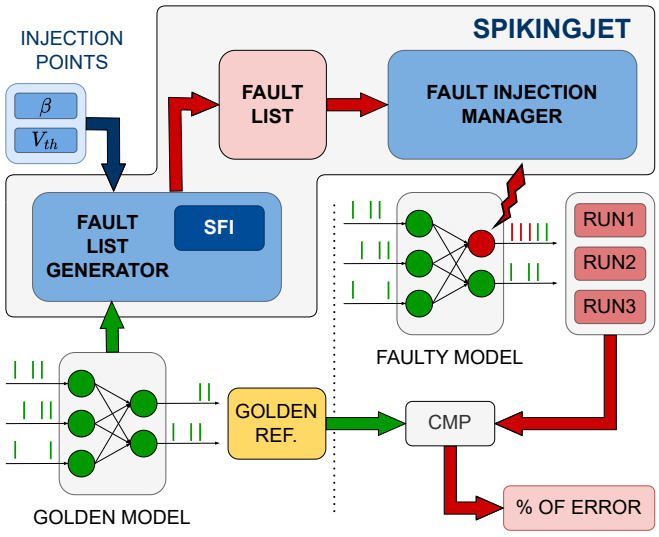

Fig. 1. SpikingJET framework

The initial step involves FL generation: the FLG reads user requests (injection points in Figure 1) and automatically retrieves all potential parameters available for injection within the network and verifies the compatibility of user requests with them. Subsequently, it constructs the complete FL, containing the exhaustive list of single-bit faults to be injected within the SNN. Here, the FIM must interpret each entry of the FL to ascertain the precise bit position for fault injection. Each entry specifies (i) the parameter for injection, which could be synaptic weights (W), neuron parameters (V_{th}, β), evolving state (V_m), or activation function (s_{out}), (ii) the spatial coordinates of the selected parameter within the network, indicating the specific layer, neuron, and connection for injection, and (iii) the index of the bit for injection. Since an exhaustive fault injection in all the bits of all the requested parameters is generally unfeasible due to the size of the considered architectures, the FLG adopts a stochastic approach based on the SFI formula to ensure a thorough exploration when generating thee FL, as explained in Section II.

The second step consists of the effective fault injection campaign. The FIM sequentially reads each entry of the FL and executes a complete inference cycle for each, introducing the corresponding fault and recording the network's output predictions for subsequent comparison with the golden reference. The persistence of the fault within the network throughout the entire inference process is ensured via two distinct methods, contingent on the target injection point: if the target parameter is static, meaning it is read-only, it is corrupted before starting the inference process. Once corrupted, no further actions are necessary as the altered value remains unchanged during inference. This applies to synaptic weights, thresholds, and exponential decay constants. Conversely, the permanent fault must be refreshed for dynamic parameters each time the value

is rewritten. This is obtained by precisely instrumenting the code to access the target variable during the forward pass and to be able to actively modify it at runtime throughout the inference cycle. To simplify matters, the permanent fault is refreshed at every iteration, regardless of the actual rewriting of the target parameter. This approach is adopted for both membrane potentials and output spikes.

SpikingJET supports the injection of stuck-at faults across all five points outlined in Section II, encompassing three primary fault models:

1) Stuck-at faults affecting synaptic connections
2) Byzantine neurons, indicating timing variations in the firing activity of neurons' outputs
3) Crashed neurons, specifically dead and saturated neurons

The analysis aims to evaluate the sensitivity of any targeted location to faults, discerning the impact each has across various layers. Consequently, injection results are averaged across different bit positions of the fault occurring within the same parameter and neurons within the same layer.

V. Experimental Results

A. Experimental setup

SpikingJET is evaluated on three distinct models, targeting three different datasets, obtained through specialized sensors or conversion chains, specifically developed to be used with SNNs:

1) N-MNIST [10]: spiking version of the original frame-based MNIST[24] dataset, obtained moving an ATIS [25] event camera in front of the 28x28 8-bit greyscale images of handwritten digits. It counts 10 classes, with numbers from 0 to 9. A fully connected network with one hidden layer of 500 neurons and one output layer of 10 neurons is used to classify it, reaching a golden accuracy of 83.58%.
2) SHD [11]: a dataset of spoken digits with 20 classes corresponding to numbers from 0 to 9 pronounced in English and German. Data were converted into trains of spikes using a biologically plausible model of the human cochlea. In this case, a fully connected recurrent model features a hidden layer with 200 neurons and an output layer of 20, with fully connected feedback connections between neurons in the hidden layer. The golden accuracy is 65.86%.
3) DVS128 [12]: dataset of 11 gestures recorder through a DVS camera[26]. In this case, a convolutional model is used with a 12C5-32C5-800-11 architecture, where xCy identifies a convolutional layer with x channels and a kernel of size $y \cdot y$. The golden accuracy is 72.26%.

In all the models, there are coupled one fully connected layer followed by one LIF layer, where the weights of the fully connected are used for the synapses connection of the LIF layer. The selected models don't aim at reaching State-of-Art (SoA) accuracy on the three benchmarks, which attest around 99.6% for the N-MNIST [27], 98.0% for the DVS128 gesture [28] and 95.1% for the SHD[29], and their optimization is out of the scope of this work. The goal was to obtain reference models to evaluate the impact of the fault injection through SpikingJET.

TABLE I
Breakdown of all potential target parameters and variables, aggregate per layer typology.

Layer	Framework	Injectable Parameters
Fully Connected	Torch	weight, bias
Convolutional	Torch	weight, bias
LIF	snnTorch	beta, threshold, potential, spike

A first run is used by SpikingJET to extract the injectable parameters needed to build the FL. Table I reports all the parameters and state variables targeted by the FI per layer type, in line with what seen in Section II, and they represent the Injection Point as depicted in Figure 1. It is worth underlining that the adopted neural networks are all based on *Float32* torch tensors. Fault injections have been performed injecting stuck-at permanent faults with an error margin of $e = 1\%$ and a confidence level $t = 99\%$ (see SFI in Equation 3), following [30]. Different metrics are considered to classify all types of Silent Data Corruption (SDC). In particular, the top-ranked element in the output scores vector, indicating the SNN prediction. As such, the impact of the fault can be classified as:

1) Masked: top-ranked elements predicted scores are the same between faulty and golden runs.
2) SDC-1: The top-ranked element predicted differs from that predicted by its error-free execution. This type of SDC is particularly critical since it changes the network prediction.
3) SDC min-max%: The confidence score of the top-ranked element deviates between min% and +/-max% from its value in the error-free execution.

B. Results

Table II summarizes the injection campaigns for the three datasets, showing details about the amount of injected permanent faults and the total time required for the injection.

TABLE II
Total amount of injected fault and time required for a complete fault-injection campaign

Network Model	Injected faults	FI time [h:m:s]
DVS128	16,307	1:42:25
N-MNIST	15,944	4:59:06
SHD	16,578	1:09:44

The first thing to notice is that SpikingJET can perform a complete injection campaign involving all the available parameters within the network, with an overall time requirement between 1 and 5 hours, in line with the results reported in [9]. This confirms that the original tool was extended with the capability to inject in SNN typical parameters, maintaining similar time constraints and enabling a fast exploration of different fault mechanisms in SNNs.

The outcomes of the fault injection campaign, utilizing the specified parameters across the entire network for the three datasets, are depicted in Figure 2. Notably, most faults belong to the masked category, corroborating the inherent resilience of NNs as reported in prior studies [18]. Furthermore, it appears that simpler networks exhibit greater resilience. However, the analysis conducted may not be exhaustive enough to establish this observation as a general rule: the FF-FC network used to classify the N-MNIST dataset exhibits only minor susceptibility to the injections, whereas the convolutional and recurrent models employed for the DVS128 and SHD datasets display a higher degree of vulnerability. Moreover, among the unmasked faults, only a small subset results in complete mispredictions ($1\% \leq SDC - 1 \leq 6.5\%$), and generally, the proportion of faults leading to an SDC greater than 5% is quite restricted.

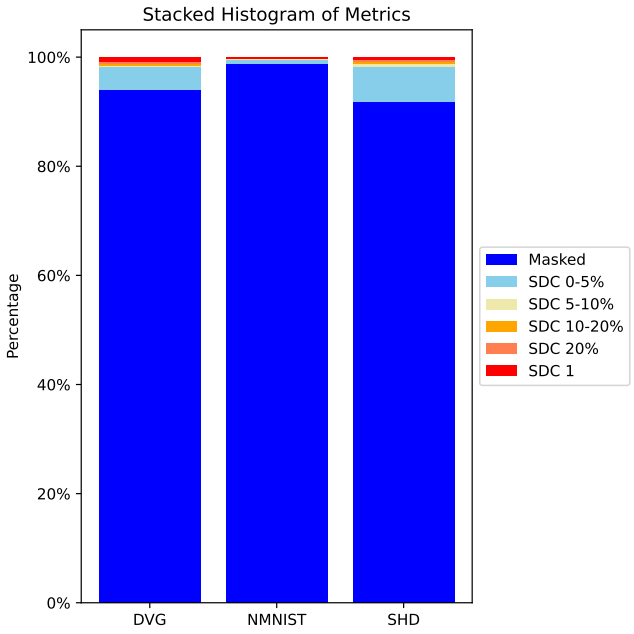

Fig. 2. Overall classification results on the different benchmarks

For completeness, a more in-depth analysis is undertaken, examining vulnerability data at the granularity of individual layers. The findings are presented in Table III. The percentage of faults injected relative to the total injectable parameters for each layer is reported as $n\%$. For instance, taking the second FC layer, denoted as fc2 in the N-MNIST section of Table III, it connects the 500 neurons of the hidden layer with the 10 neurons of the output layer, resulting in a total of $500 \cdot 10 = 5000$ weights. In this scenario, with $n\% = 3.75\%$, an average of $0.0375 \cdot 5000 = 188$ weights were injected with stuck-at faults.

Analyzing the results obtained using the simple FC architecture applied to the N-MNIST dataset, one noticeable trend is that as the injection is directed towards layers closer to the output, the impact becomes increasingly pronounced. This observation is evident in the diminishing number of masked faults with increasing layer index and the different categories of SDC, which exhibit an opposite trend. Zooming in on individual parameters, the impact on the bias in linear layers is more significant than that on synaptic weights. This is reasonable because, typically, a synaptic weight represents the contribution of a single input to the total synaptic current. In contrast, the bias influences the overall value of the current itself, often having a larger magnitude and, consequently, a stronger impact. Looking at the spiking layers then, central to this work, the impact of a fault in each of the model parameters can be seen. In particular, faults affecting the threshold or the output spike have the highest impact. This was again expected since they directly affect the neuron activity, leading in extreme cases to the aforementioned dead and saturated neuron situations, which are the harsher ones. Results on the recurrent FC model used on the SHD are in some ways comparable to the N-MNIST ones. The impact of the faults, in general, increases moving towards the outer layers. Interestingly, faults in the feedback connections cause more errors than the FF ones. Results on the last LIF layer, particularly the membrane potential and spike, are probably outliers and need further exploration, targeting them with specific fault injection campaigns. Finally, the CSNN applied to the DVS128 dataset has a quite different behavior: faults in the first convolutional layers have a larger impact from the point of view of SDCs, indicating that the initial feature extraction phase is likely the most crucial aspect in such architectures. On the other hand, LIF layers behave in the usual way: faults impact more in outer layers. This is even more evident concerning the previous models, suggesting that the phenomenon is even stronger with deeper architecture. Additionally, the model time constant is the second most relevant quantity, immediately after the threshold. Given the temporal relevance of features in gesture recognition, this underlines the importance of temporal information in this specific model, meaning that timing variations in the neurons' activity, i.e., byzantine neuron behaviors, are more impactful in this case. However, it has to be noticed that the presented results are preliminary, and a more thorough and comprehensive analysis should be conducted, focusing on injecting faults into individual layers. Indeed, the network-wise fault injection may cause the error margin at the layer level to be greater than the predefined value ($e = 1\%$). Consequently, results can hint at the general reliability of different network parameters. Nonetheless, this initial analysis already showcases the capabilities of SpikingJET in conducting comprehensive fault injection campaigns within reasonable timeframes. Moreover, the results of the analysis highlight the resilience of different families of SNN models against stuck-at faults.

VI. Conclusions

In conclusion, this paper introduces SpikingJET, a FI tool to assess the resilience of FC and convolutional SNNs to hardware faults. SpikingJET provides a comprehensive platform for assessing the resilience of SNNs by inducing errors and injecting transient and permanent faults into critical components

TABLE III
ANALYSIS OF THE LAYER-WISE IMPACT OF FAULTS ON THE THREE BENCHMARK DATASETS

NMNIST									
Layer Name	Layer Type	Parameter	n%	SDC 1	SDC 0-5%	SDC 5-10%	SDC 10-20%	SDC 20%	Masked
fc1	Fully Connected	Bias	4.84	0.58	5.58	0.26	0.30	0.50	92.75
		Weight	4.18	0.15	0.67	0.03	0.03	0.05	99.04
lif1	Leaky	Beta	3.90	0.69	7.16	0.30	0.43	0.67	90.72
		Potential	4.37	1.99	7.03	0.27	0.43	0.73	89.51
		Spike	4.21	3.65	10.89	0.36	0.69	1.19	83.18
		Threshold	6.40	2.51	7.41	0.30	0.42	0.73	88.59
fc2	Fully Connected	Bias	3.75	2.70	8.45	0.09	0.39	0.62	87.71
		Weight	3.75	3.10	3.95	0.06	0.18	0.28	92.40
lif2	Leaky	Beta	5.62	3.08	10.11	0.08	0.46	0.71	85.54
		Potential	2.81	2.44	12.30	0.04	0.06	0.09	85.05
		Spike	3.75	8.78	10.54	0.13	0.21	0.36	79.03
		Threshold	4.68	22.91	5.55	0.09	0.33	0.55	70.54
SHD									
Layer Name	Layer Type	Parameter	n%	SDC 1	SDC 0-5%	SDC 5-10%	SDC 10-20%	SDC 20%	Masked
fc1	Fully Connected	Bias	0.31	2.76	16.84	1.66	1.77	1.74	75.21
		Weight	4.46	0.40	5.47	0.36	0.31	0.31	93.12
fb1	Fully Connected	Bias	0.35	1.87	20.26	2.25	2.16	1.96	71.47
		Weight	0.27	0.55	8.28	0.59	0.50	0.47	89.58
lif1	Leaky	Beta	0.37	2.06	32.58	3.04	2.91	2.51	56.88
		Potential	0.31	3.17	40.59	2.31	1.96	2.18	49.76
		Spike	0.28	13.59	37.97	2.82	2.61	3.14	39.56
		Threshold	0.35	3.60	24.32	2.28	2.16	2.95	64.66
fc2	Fully Connected	Bias	0.31	1.26	53.56	1.14	0.90	1.34	41.77
		Weight	0.28	1.90	16.74	0.22	0.18	0.23	80.67
lif2	Leaky	Beta	0.15	0	14.79	0	0	0	85.20
		Potential	0.15	0	0	0	0	0	100
		Spike	0.31	0	0	0	0	0	100
		Threshold	0.62	1.07	26.39	0.50	0.43	0.97	70.61
DVS128									
Layer Name	Layer Type	Parameter	n%	SDC 1	SDC 0-5%	SDC 5-10%	SDC 10-20%	SDC 20%	Masked
conv1	Convolutional	Bias	2.86	1.46	13.46	1.38	1.49	1.67	80.54
		Weight	1.98	1.70	23.04	1.75	1.61	2.12	69.78
lif1	Leaky	Beta	1.56	1.95	25.95	1.79	2.06	2.12	66.13
		Potential	1.94	1.44	7.33	0.52	0.50	0.71	89.50
		Spike	2.00	1.59	7.06	0.51	0.48	0.69	89.66
		Threshold	2.01	4.90	22.05	1.82	2.73	4.38	64.11
conv2	Convolutional	Bias	1.27	5.41	4.21	0.33	0.66	0.72	88.67
		Weight	2.02	0.73	3.37	0.25	0.25	0.36	95.05
lif2	Leaky	Beta	3.13	1.48	15.16	1.25	1.72	1.80	78.59
		Potential	1.91	0.88	5.05	0.56	0.57	0.87	92.06
		Spike	2.05	0.56	3.66	0.36	0.40	0.53	94.48
		Threshold	2.50	15.82	22.27	2.54	2.05	5.57	51.76
fc1	Fully Connected	Bias	1.42	0.00	0.55	0.00	0.08	0.08	99.30
		Weight	2.01	0.44	1.99	0.07	0.07	0.10	97.32
lif3	Leaky	Beta	0.85	6.12	52.60	2.99	2.60	2.86	32.81
		Potential	2.27	14.26	62.94	0.78	0.98	1.32	18.41
		Spike	2.27	4.20	62.79	0.29	1.22	0.59	30.91
		Threshold	1.14	90.53	4.10	1.37	1.86	2.15	0.00

such as synaptic weights, neuron model parameters, internal states, and activation functions. A first analysis conducted with SpikingJET shows the overall robustness of SNNs to stuck-at faults. Simultaneously, it enables the identification of the network's weakest components, shedding light on areas where focusing fault tolerance techniques would be beneficial. SpikingJET establishes a foundation for in-depth examination of SNN resilience, starting with a comprehensive analysis of the network and subsequently concentrating on its most vulnerable components. With its rapid experimental pace, it facilitates agile and adaptable evaluation of SNN resilience. To encourage research in this field, we release the code related to our experiments as open-source: https://github.com/smilies-polito/SpikingJET

REFERENCES

[1] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *Int. J. Comput. Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

[2] Alessio Carpegna, Alessandro Savino, and Stefano Di Carlo. "Spiker: an FPGA-optimized Hardware accelerator for Spiking Neural Networks". In: *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2022, pp. 14–19. DOI: 10.1109/ISVLSI54635.2022.00016.

[3] Alessio Carpegna, Stefano Di Carlo, and Alessandro Savino. "Artificial resilience in neuromorphic systems". In: *Proceedings of the 12th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*. 2022, pp. 112–114.

[4] Theofilos Spyrou et al. "Neuron Fault Tolerance in Spiking Neural Networks". In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. ISSN: 1558-1101. Feb. 2021, pp. 743–748. DOI: 10.23919/DATE51398.2021.9474081.

[5] Karthikeyan Nagarajan et al. "Analysis of Power-Oriented Fault Injection Attacks on Spiking Neural Networks". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. ISSN: 1558-1101. Mar. 2022, pp. 861–866. DOI: 10.23919/DATE54114.2022.9774577.

[6] Karthikeyan Nagarajan et al. "Fault Injection Attacks in Spiking Neural Networks and Countermeasures". English. In: *Frontiers in Nanotechnology* 3 (Jan. 2022). ISSN: 2673-3013. DOI: 10.3389/fnano.2021.801999.

[7] Alessio Colucci, Andreas Steininger, and Muhammad Shafique. "enpheeph: A Fault Injection Framework for Spiking and Compressed Deep Neural Networks". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2022, pp. 5155–5162. DOI: 10.1109/IROS47612.2022.9982181.

[8] Jason K Eshraghian et al. "Training spiking neural networks using lessons from deep learning". In: *Proceedings of the IEEE* 111.9 (2023), pp. 1016–1054.

[9] G. Gavarini, A. Ruospo, and E. Sanchez. "SCI-FI: a Smart, aCcurate and unIntrusive Fault-Injector for Deep Neural Networks". In: *2023 IEEE European Test Symposium (ETS)*. 2023, pp. 1–6. DOI: 10.1109/ETS56758.2023.10173957.

[10] Garrick Orchard et al. "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades". English. In: *Frontiers in Neuroscience* 9 (Nov. 2015). ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00437. (Visited on 03/11/2024).

[11] Benjamin Cramer et al. "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.7 (July 2022), pp. 2744–2757. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2020.3044364.

[12] Arnon Amir et al. "A Low Power, Fully Event-Based Gesture Recognition System". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 7388–7397. DOI: 10.1109/CVPR.2017.781.

[13] Daniel Auge et al. "A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks". en. In: *Neural Processing Letters* 53.6 (Dec. 2021), pp. 4693–4710. DOI: 10.1007/s11063-021-10562-2. (Visited on 12/08/2023).

[14] A. N. Burkitt. "A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input". en. In: *Biological Cybernetics* 95.1 (July 2006), pp. 1–19. DOI: 10.1007/s00422-006-0068-6.

[15] M. Bushnell and Vishwani Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer Publishing Company, Incorporated, 2013. ISBN: 1475781423.

[16] Dario Mamone et al. "On the Analysis of Real-time Operating System Reliability in Embedded Systems". In: *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. 2020, pp. 1–6. DOI: 10.1109/DFT50435.2020.9250861.

[17] Michele Portolan et al. "Alternatives to fault injections for early safety/security evaluations". In: *2019 IEEE European Test Symposium (ETS)*. IEEE. 2019, pp. 1–10.

[18] A. Ruospo et al. "Assessing Convolutional Neural Networks Reliability through Statistical Fault Injections". In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2023, pp. 1–6. DOI: 10.23919/DATE56975.2023.10136998.

[19] Haralampos-G. Stratigopoulos, Theofilos Spyrou, and Spyridon Raptis. "Testing and Reliability of Spiking Neural Networks: A Review of the State-of-the-Art". In: IEEE Computer Society, Oct. 2023, pp. 1–8. DOI: 10.1109/DFT59622.2023.10313541.

[20] Sarah A. El-Sayed et al. "Spiking Neuron Hardware-Level Fault Modeling". In: *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. ISSN: 1942-9401. July 2020, pp. 1–4. DOI: 10.1109/IOLTS50870.2020.9159745.

[21] Rachmad Vidya Wicaksana Putra, Muhammad Abdullah Hanif, and Muhammad Shafique. "ReSpawn: Energy-Efficient Fault-Tolerance for Spiking Neural Networks considering Unreliable Memories". In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. Munich, Germany: IEEE Press, Nov. 2021, pp. 1–9. DOI: 10.1109/ICCAD51958.2021.9643524.

[22] Elena-Ioana Vatajelu, Giorgio Di Natale, and Lorena Anghel. "Special Session: Reliability of Hardware-Implemented Spiking Neural Networks (SNN)". In: *2019 IEEE 37th VLSI Test Symposium (VTS)*. Monterey, CA, USA: IEEE, Apr. 2019, pp. 1–8. DOI: 10.1109/VTS.2019.8758653. URL: https://ieeexplore.ieee.org/document/8758653/.

[23] Jingying Li, Bo Sun, and Xiaoyan Xie. "A Reliability Assessment Approach for A LIF Neurons Based Spiking Neural Network Circuit". In: *2023 24th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*. Graz, Austria: IEEE, Apr. 2023, pp. 1–7. DOI: 10.1109/EuroSimE56861.2023.10100785. (Visited on 03/11/2024).

[24] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998). Conference Name: Proceedings of the IEEE, pp. 2278–2324. DOI: 10.1109/5.726791.

[25] Christoph Posch et al. "Live demonstration: Asynchronous time-based image sensor (ATIS) camera with full-custom AE processor". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010, pp. 1392–1392. DOI: 10.1109/ISCAS.2010.5537265.

[26] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[27] Ali Samadzadeh et al. "Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction". In: *Neural Processing Letters* 55.6 (Dec. 2023), pp. 6979–6995. DOI: 10.1007/s11063-023-11247-8.

[28] Xueyuan She, Saurabh Dash, and Saibal Mukhopadhyay. "Sequence Approximation using Feedforward Spiking Neural Network for Spatiotemporal Learning: Theory and Optimization Methods". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=bp-LJ4y_XC.

[29] Ilyass Hammouamri, Ismail Khalfaoui-Hassani, and Timothée Masquelier. "Learning Delays in Spiking Neural Networks using Dilated Convolutions with Learnable Spacings". In: *The Twelfth International Conference on Learning Representations*. 2024.

[30] R. Leveugle et al. "Statistical fault injection: Quantified error and confidence". In: *2009 Design, Automation & Test in Europe Conference & Exhibition*. 2009, pp. 502–506. DOI: 10.1109/DATE.2009.5090716.