POLITECNICO DI TORINO
Repository ISTITUZIONALE

DEAR-CNN: Data-Efficient Assessment of Resiliency in Convolutional Neural Networks

Original
DEAR-CNN: Data-Efficient Assessment of Resiliency in Convolutional Neural Networks / Bellarmino, N., Bosio, A., Cantoro, R., Ruospo, A., Sanchez, E.. - ELETTRONICO. - (2025), pp. 13-18. (28th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS) Lyon (FRA) May 5-7, 2025) [10.1109/DDECS63720.2025.11006797].

Availability:
This version is available at: 11583/2999932 since: 2025-05-07T09:52:10Z

Publisher:
IEEE

Published
DOI:10.1109/DDECS63720.2025.11006797

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

11 November 2026

DEAR-CNN: Data-Efficient Assessment of Resiliency in Convolutional Neural Networks

Nicolò Bellarmino*, Alberto Bosio†, Riccardo Cantoro*, Annachiara Ruospo*, Ernesto Sanchez*

*CAD Research Group, Politecnico di Torino, Italy
{name.surname}@polito.it

† Centrale Lyon, INSA Lyon, CNRS
UCBL, CPE Lyon, INL, UMR5270, Ecully, France.
alberto.bosio@ec-lyon.fr

***Abstract*—Convolutional Neural Networks (CNNs) are widely employed in various domains, including safety-critical applications such as autonomous driving. In these scenarios, the reliability of CNNs can be compromised by hardware faults occurring during inference, potentially leading to severe consequences. Evaluating the resilience of CNNs to hardware faults is primarily conducted through Fault Injection (FI) campaigns. However, a significant challenge lies in selecting an appropriate workload. Typically, the entire test set is applied for every injected fault, making the process highly time-consuming and posing difficulties for timely assessments. This paper investigates image selection strategies to rank inputs from the test dataset based on their *difficulty* in being classified by the CNN. The objective is to identify a minimal subset of test data that enables reliable CNN assessment while reducing computational overhead. By prioritizing *challenging* samples, the proposed method focuses on inputs that are more likely to reveal network vulnerabilities under fault conditions, enhancing the efficiency of the reliability evaluation process. Experimental results demonstrate that using such a subset of the test data suffices to estimate the number of critical faults for at least one image. This approach not only accelerates the reliability evaluation but also provides novel insights into CNN reliability, offering a practical framework for continuous assessments.**

***Index Terms*—Fault Injection, Neural Networks, Reliability, Hardware Faults, Rational Sampling, Test Compaction, Fault Tolerance, Machine Learning.**

I. Introduction

Deep Neural Networks (DNNs), particularly Convolutional Neural Networks (CNNs), have become the state-of-the-art models for computer vision tasks in modern computational environments. Their deployment in safety-critical embedded systems demands a comprehensive understanding of their resilience to random hardware faults. However, the increasing complexity of these models poses significant challenges to efficiently assessing their reliability. Fault Injection (FI) campaigns are widely regarded as effective methods for evaluating the reliability of DNNs [1], but exhaustive fault simulations remain computationally prohibitive. While Statistical Fault Injection (SFI) techniques [2] have successfully reduced the sample size of faults under analysis, the computational time required for FI campaigns can still be significantly optimized, particularly by addressing the selection of input stimuli, i.e., the *workload* used during FI. Despite its importance, this aspect has not been adequately explored for image classification tasks with CNNs.

Previous work on image selection has primarily focused on hardware fault detection, often emphasizing the creation of specialized test images to achieve high fault coverage [3]. In contrast, this paper tackles the challenge of optimizing FI campaigns by strategically selecting a representative subset of testing data, focusing on the most *challenging* samples for classification tasks. These samples are identified using metrics computed at the pixel, embedding, or probability vector levels.

Our primary contributions and outcomes include:

- We empirically showed that challenging samples carry critical information about a network's resilience, as they are the most prone to activate critical faults—those that alter the network's predictions.
- By leveraging only challenging images as the workload for FI campaigns, we effectively estimate the proportion of critical faults for the entire dataset without the computational overhead of using the full test set.
- We show that metrics derived from probability vectors are effective in identifying challenging samples suitable for FI campaigns, enabling targeted and efficient reliability assessments.

The rest of the paper is organized as follows: Section II provides the necessary background information on neural network reliability and fault injection techniques. Section III reviews related works in the field. Section IV outlines the motivations and describes the proposed methodology in detail. Section V details the experimental setup, while Section VI presents the findings and discusses their implications. Finally, Section VII concludes the paper and highlights potential directions for future research.

II. Background

In dependability domains, fault injection is a *de facto* standard and popular method to assess the resilience of a system. It compares the network's behavior under normal conditions to its behavior with simulated faults, ensuring it meets functionality and safety standards. CNNs exhibit inherent fault tolerance due to redundancy and parameterization

979-8-3315-2801-0/25/$31.00 ©2025 IEEE

[4], [5]. However, it has been proved that, in safety-critical applications, even a single fault such as a bit-flip can lead to severe consequences, like misclassification in autonomous driving [6]. Therefore, functional testing techniques, such as FI, play a crucial role in ensuring the reliability and safety of CNNs, especially in safety-critical applications. Emerging standards like *ISO/IEC CD TR 5469* [7] emphasize enhancing the functional safety of AI systems. Recent works in the literature have deeply focused on considering and corrupting bit-flips in NNs' static parameters such as synaptic weights, which are typically stored in 32-bit floating-point formats [1] (but also other data representations have been under scope, e.g. [8]). These faults, categorized as Masked, Non-Critical, or Critical, can lead to varying impacts on the network's output, ranging from no observable changes to severe misclassifications [9]. Among the different fault types, the stuck-at fault model is commonly used for simulating permanent faults in CNN reliability studies. However, exhaustive fault simulation is computationally expensive due to the huge fault universe. To mitigate this, Statistical Fault Injection (SFI) has been proposed, leveraging random sampling to select a statistically significant subset of faults for evaluation, reducing computational demands while maintaining accuracy [2].

Representative Sampling and Metrics

Efficiently testing CNNs with limited labeled data is a significant challenge and an active research area in machine learning [10]. In the ML community, fault detection aims to *identify test samples with a higher likelihood of misclassification*. While this is not directly related to faults in the underlying hardware, it becomes particularly critical when hardware faults disrupt normal behavior during inference. These methods focus on test selection strategies to prioritize samples that provide the most informative insights into the network's performance [11], [12].

Representative sampling involves sorting and selecting test samples based on specific metrics, which can be computed either at the embedding level (intermediate network representations) or using probability vectors (predicted class probabilities). Deterministic algorithms, such as the Kennard-Stone (KS) method [13], select samples iteratively by maximizing their distance, often using measures like the Euclidean or Correlation distance [14]. These techniques aim to create subsets that maintain the diversity and representativeness of the larger dataset. In addition, disagreement-based metrics are used to identify challenging or ambiguous samples. DeepGini [15], an unsupervised metric, measures the spread of class probabilities, with higher values indicating greater uncertainty. EL2N [16], a supervised metric, calculates the Euclidean norm of the error vector between the predicted probability vector and the true label. Cross-Entropy another supervised metric, quantifies the divergence between the predicted and true probability distributions [17].

III. Related Work

Recent research on fault detection and test selection for neural networks has primarily focused on crafting specialized test images or subsets to identify critical hardware faults. Gradient-based adversarial crafting techniques [3] and methods leveraging probability vectors [18], [19] have been extensively studied. The latter includes model-agnostic distance metrics and confidence-based predictions but has been validated only on Multi-Layer Perceptrons (MLPs), achieving limited accuracy (35% on CIFAR-10) and excluding misclassified samples. Other works, such as [20], [21], focus on creating test sets for online fault detection under fault models that flip the most significant bit (MSB) of weights, often corrupting multiple weights or bits simultaneously at rates up to 1%. These approaches differ significantly from our focus on single-bit faults, which are more challenging to detect. Additional studies have explored error resilience in Approximate Computing hardware by reducing test sets based on the Gini Index [22] and applying ranking methods to Spiking Neural Networks using top-1 and top-2 softmax outputs [23]. Unlike these works, our approach targets state-of-the-art CNN architectures, demonstrating empirically that the most challenging samples inherently carry critical information about network vulnerability. By leveraging deterministic metrics, we identify representative samples that highlight fault criticality, bridging gaps left by prior studies focused on simplified architectures or high fault-rate scenarios.

IV. Proposed Approach

Performing an exhaustive FI campaign on NNs weights is computationally impractical due to the vast dimensionality of the fault space. While SFI techniques have reduced the number of injected faults required for analysis, the choice of input stimuli, or *workload*, remains a significant challenge. Current methodologies typically rely on the entire test set as the workload [1], which is suboptimal for two reasons: (1) many faults are critical for multiple images simultaneously, leading to redundancy, and (2) a significant portion of faults are masked, resulting in no output differences during inference. This observation suggests that a smaller, carefully selected subset of images could effectively uncover most critical faults. In this work, we propose to optimize FI on NNs leveraging *challenging* samples only: images that are inherently difficult for the network to classify and are more likely to activate critical faults. Challenging samples represent extreme cases that stress the network under worst-case conditions, revealing latent vulnerabilities in its reliability. The proposed framework is based on adopting a metric to identify challenging samples and then perform the following steps:

1) **Forward Pass**: Compute the chosen metric for each image in the test set through a single forward pass;
2) **Ranking**: Rank the images based on the computed metric to prioritize those with the highest likelihood of uncovering critical faults;
3) **Subset Selection**: Select the top n ranked images as the workload for the FI campaign.

This approach significantly reduces the computational overhead associated with FI campaigns. Metric computation involves only a forward pass through the network, making the image selection process nearly instantaneous. Moreover, this

method is highly flexible and can be seamlessly integrated into any FI framework (e.g., CUDA, CPU, TPU) with minimal code adjustments.

To evaluate the reliability of a network, we cast the problem as a fault detection task, categorizing each fault based on its most severe impact across the dataset. For each fault, the faulty and fault-free outputs are compared across all images in the workload, with faults cumulatively classified based on their severity. By logging the cumulative number of critical faults as a function of the workload size, we generate *fault detection curves*, which validate the effectiveness of challenging sample selection and reveal correlations between image characteristics and fault activation.

This approach enables rapid reliability assessment by prioritizing challenging samples that provide a representative estimation of network criticality. By focusing on a subset of images, our method accelerates FI campaigns, reducing computational costs while maintaining high fault coverage under worst-case scenarios.

Algorithm 1 Fault Detection and Categorization

Require: $sorted_image_list, fault_list, DNN$
Ensure: Counts of masked, not critical, and critical faults

```
severity_list ← [0, 0, ..., 0]
for image ∈ sorted_image_list do
    for fault ∈ faults_list do
        severity ← labelFault(DNN, fault, image)
        if severity > severity_list[fault] then
            severity_list[fault] = severity
        end if
    end for
end for
masked_f, not_critical_f, critical_f = count(severity_list)
Output: masked_f, not_critical_f, critical_f
```

V. Experimental Setup

We investigated six distinct models spanning four prominent deep learning architectures, each contributing uniquely to the evolution of neural network design and efficiency. We employed two Residual Neural Networks (ResNets) [24] architectures with 9 and 18 layers, a LeNet [25] model, two VGG [26] architecture with 11 and 16 layers, and a MobileNetV2 [27], [28]. These networks are particularly well-suited for deployment in resource-constrained environments, such as embedded systems, edge devices, or low-power applications, where computational efficiency and memory footprint are critical considerations. We validate our experiments on two popular datasets for image classification: CIFAR-10 [29] and MNIST [30]. The datasets are divided into predefined training and test splits. The test set for both datasets is composed of 10,000 images. A fault injection campaign was conducted using a SFI process [2], [31]. Permanent hardware faults have been considered by means of stuck-at fault injections in the weights of CNNs. An injected fault involves flipping a random bit of a random parameter of the network, according to the SFI process. About 16,640 stuck-at faults have been injected to get an estimate with an error of $e = 0.01$ and a 0.99 confidence. Selected bits are flipped, under the single fault assumption (i.e., flipping one bit at a time). Results of the SFI campaigns are presented in Table I. The *Clean Acc* indicates the accuracy of the network in the fault-free condition, while the *Faulty Acc* column provides the accuracy when a fault is introduced. The *Not Critical Inferences* column represents the percentage of test samples whose final classification outcomes are unaffected by the introduced faults (it thus compromised both *masked* and *not-critical* inferences, as described in Section II and Reference [9]), while the *Critical Inferences* column indicates the percentage of samples for which faults result in a misclassification. Finally, the *Critical Faults* column reports the total number of unique faults identified as critical for at least one inference, highlighting the network's vulnerability to single-fault scenarios. All experiments were performed in Python using PyTorch tools for the DL models. Experiments run on a server equipped with an Intel® Core™ i9-9900K CPU @3.60GHz x 16, 32GB of RAM, and an Nvidia® 2080 TI GPU.

TABLE I
Accuracy metrics, fault-type distributions, and critical faults for an SFI process involving 16,640 injected faults.

Model	Clean Acc	Faulty Acc	Not Critical Inferences	Critical Inferences	Critical Faults
CIFAR10					
ResNet18	93.52%	91.24%	97.52%	2.46%	826
ResNet9	92.17%	89.64%	97.97%	2.77%	1508
LeNet	79.36%	79.07%	99.64%	0.34%	1158
VGG11	90.10%	88.15%	97.82%	2.18%	819
VGG16	91.61%	89.53%	97.71%	2.29%	894
MobileNetv2	91.72%	89.47%	97.51%	2.49%	1614
MNIST					
ResNet18	99.63%	97.08%	97.45%	2.55%	581
ResNet9	99.65%	96.87%	97.22%	2.78%	527
LeNet	99.25%	99.09%	99.85%	0.15%	484
VGG11	99.68%	97.36%	97.67%	2.33%	661
VGG16	99.58%	97.22%	97.64%	2.36%	555

VI. Results

A. Metric Selection

Several metrics can be used to identify "challenging" samples, calculated either at the probability vector level or the embedding level. In initial experiments, we compared EL2N, DeepGini, and Cross Entropy loss, evaluating the *Spearman* [32] correlation coefficient. These metrics were computed on a ResNet18 model trained on CIFAR-10, with inferences performed under fault-free conditions, using the entire test set.

TABLE II
Correlation Matrix of Selected Metrics (ResNet 18, CIFAR-10)

Metric	Cross Entropy	DeepGini	EL2N
Cross Entropy	1.00	0.99	0.86
DeepGini	0.99	1.00	0.85
EL2N	0.86	0.85	1.00

The metrics are highly correlated (the closer the values are to 1, the higher the correlation): results in Table II suggests

that using Cross Entropy or DeepGini is not particularly relevant in terms of distinguishing between sample rankings or difficulty. The moderately lower correlation of EL2N with both Cross Entropy may implies that EL2N captures some unique aspects of the data. In Fig. 1, it is possible to observe how image samples are likely to activate critical faults based on their metric values. This plot corresponds to a ResNet9 model trained on CIFAR-10, with similar plots generated for other networks. Each scatter point represents an image. The x-axis shows the metric value associated with the image, while the y-axis indicates the number of critical faults activated by that specific image. Critical faults are activated uniformly among images. However, it is evident that some images lead to more frequent fault occurrences, often corresponding to specific metric values. This observation suggests that samples with certain metric values may act as "hotspots" for faults or are inherently more sensitive to triggering faults compared to others.

This finding could be instrumental in developing fault-detection mechanisms based on functional input stimuli: in principle, it is possible to select a subset of testing images based on these metrics, creating a *worst-case* testing set to evaluate the performance and the reliability of a CNN. Alternatively, *challenging* samples can be selected and used as a starting working set for a step of crafting testing images for fault detection. While previous works have taken this fact as established without providing any proof or empirical evidence, we have investigated the phenomenon, showing that it is the challenging samples that carry most of the information about the network's criticality. As an example, for the EL2N metric it is possible to select images within a specified zone around the central metric value using a threshold ϵ. This allows us to effectively choose images with metric values in the range of $central - \epsilon$ to $central + \epsilon$, identifying images that reside close to the decision boundary between classes.

B. Fault Detection

We can face the reliability assessment problem as Fault Detection task (Algorithm 1), as explained in Section IV. We first sort images on the basis of specific metric values (in descending order), and then we run the fault detection algorithm. For each sample in the sorted image set, we plot the cumulative sum of detected faults across the defined fault categories. Plots show on the y-axis of each subplot the number of faults (masked, critical or not critical), while on x-axis the number of image samples used (Section VI-B). In our visualization, blue curves/points denote masked faults, red curves/points represent non-critical faults, and green curves/points the critical faults. Red-brick curves represent the values for the metric, normalized in the range [0,1]. Dotted horizontal green lines are the final values, while the dotted black values are placed at ± 0.1 of the steady state. By using a random approach, nearly all images are required to detect all the critical faults (Section VI-B). We run this random analysis for different random seeds (curves are the average of 5 runs of different random image sets). If, instead, we sort images and run the detection algorithm, we can see that after a transitory phase, sorted metric values quickly approach (and *plateau*) to zero. Considering the *EL2N* (Section VI-B), it is evident how all the images that have activated critical values lie in the transitory phase of the curve: when the critical faults stabilize (and thus, all of them are detected) suddenly the metric values approach zero, and the plateau is reached. This observation holds true for all metrics computed at the probability vector level; however, the results presented here focus on the EL2N metric for the sake of summarization. Not all samples are essential for detecting critical faults, but it is enough to analyze *challenging* samples to determine how many faults will be critical for at least an image in the dataset, giving insights about network reliability. In a first approximation, we can label as *challenging* every sample outside what we call the *plateau* zone. We defined the *plateau* zone as the region in which the absolute value of the first derivative of the metric functions approaches zero, or is identically below a certain threshold (we can take as an example $\epsilon = 5 \times 10^{-5}$). The samples before the plateau are considered *of interest*. Considering images for which the metric differs from zero can give us an upper bound on the images on which to run the FI. Alternatively, it is possible to use the hotspot identified in Section VI-A to select images on the basis of specific metric values. For example, we can choose an ϵ of 0.3 around the central value. With these images, we can catch almost all the faults that are critical for at least one image in the dataset with a reduced test set. Results of these are summarized in Table IV. The random detection is a mean over 5 different random samples of the same size of the EL2N central-mode set. In the last columns, we evaluated the time saving by using challenging samples only to estimate the number of critical faults/performing the FI campaign. We considered a time of 1 millisecond per inference, independent of the network being used. The total time required to test all samples (100%) is given by: Total Time $=$ $1\,\text{ms} \times 10,000\,\text{images} \times 16,640\,\text{faults} = 1.93\,\text{days}$

The time needed to perform the reliability assessments scales with the sample size. Specifically, for any given sample size, the time is calculated as:

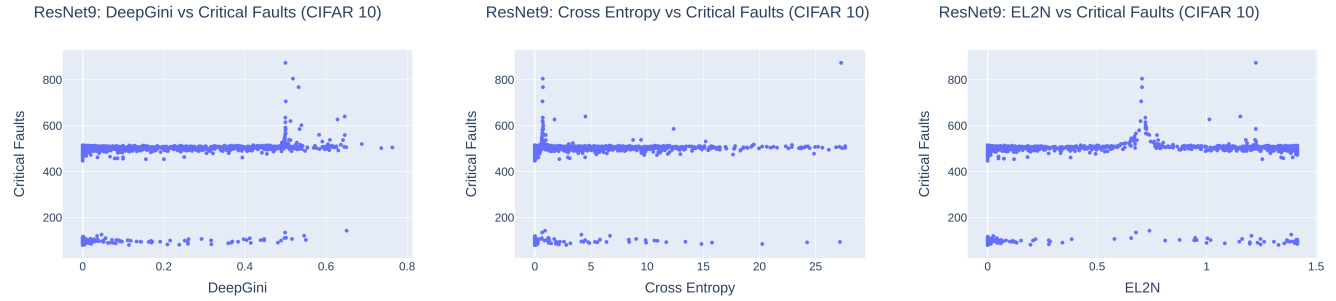
$$\text{Time} = \text{Total Time} \times \left(\frac{\%\,\text{of Samples}}{100} \right)$$

To validate the approach, we performed FI on ResNet18 trained on CIFAR10 using 5 random seeds for the fault list generation. As shown in Table III, EL2N-based selection identified more critical faults than random sampling with the same sample size.

TABLE III
EL2N-BASED DETECTION VS RANDOM, RESNET18 (CIFAR10) FOR 211 IMAGES

Seed	Critical Faults	EL2N Detection	Random Detection
1	785	784	507
2	798	797	472
3	820	814	496
4	820	815	499

Fig. 1. Metric values vs fault identified by each image in the test set (DeepGini, Cross Entropy and EL2N).



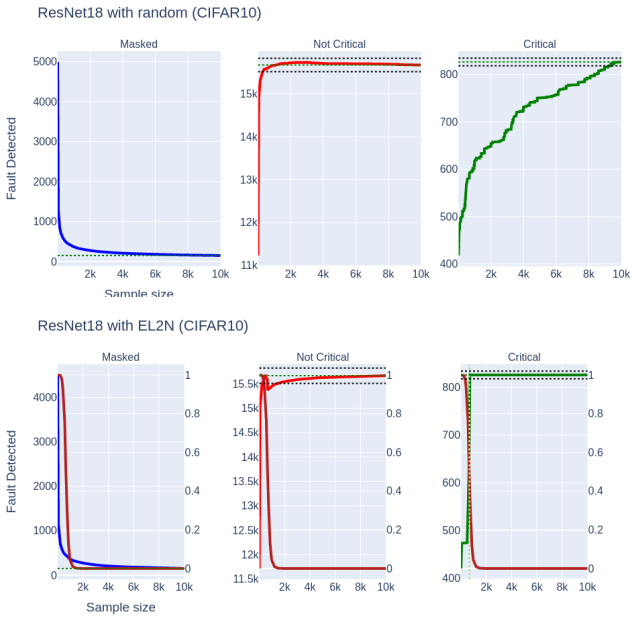


Fig. 2. *EL2N* Fault Detection. For the EL2N approach, images are sorted in descending order of the EL2N value. Red-Brick curve represent the EL2N value, for each image, normalized in the range $[0, 1]$

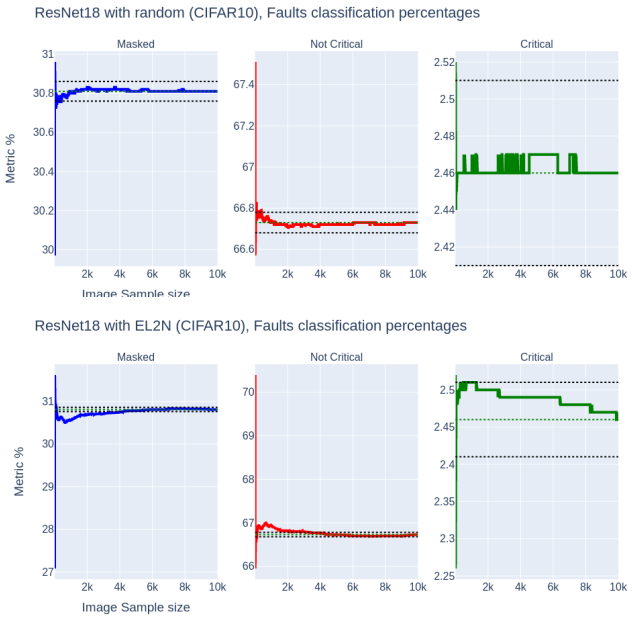


Fig. 3. Random vs *EL2N* Fault Classification (%)

C. Fault Classification

Fault classification analysis, derived from classical literature on NNs reliability assessment [1], aims to estimate the probability that, in the presence of a fault, the network's output will change or remain unaffected. This can result in the fault being classified as critical, non-critical, or masked when an image is passed through the network. Random sampling methods provide an accurate estimate of these probabilities, even when only a fraction of the test set is used. Section VI-B shows the results for ResNet18, but similar plots were obtained for other networks. If we sort images based on their difficulty for the network, an overshoot in the number of faults identified as critical is observed. By prioritizing *challenging* samples, we can quickly estimate the upper bound for the network's criticality. As shown in Fig. 1, faults are evenly distributed across the images, except for those with specific metric values. This suggests that by focusing only on these challenging samples—those with particular metric values—we can effectively estimate the upper bound of critical faults in the network, speeding up the process of assessing criticality.

VII. Conclusions

This preliminary study demonstrates that the selection of test images significantly influences fault estimation in DNNs. Our analysis establishes that while random sampling methods provide accurate and unbiased fault classification estimates, prioritizing *challenging* samples enables the construction of a *worst-case* test set for CNN reliability assessments. Metrics computed at the probability vector level, such as Cross-Entropy and especially EL2N, are particularly effective in identifying *challenging* samples. This approach facilitates a faster estimation of the number of faults critical for at least one image, providing deeper insights into NNs reliability beyond the conventional metrics of masked, critical, and non-critical faults used in the literature. Consequently, we can avoid testing the entire dataset by focusing on the most informative and critical samples. Furthermore, our findings reveal that specific metric values are correlated with the detection of critical faults. Since faults are uniformly distributed across images except for critical samples, these samples can be prioritized as

TABLE IV
RANDOM VS EL2N-BASED DETECTION

Network	Critical Faults	Method	% of Samples	Faults Detected	Time (days)
CIFAR10					
LeNet	1158	Central Mode	25.32%	1151	0.49
		Random Detection	25.32%	742	0.49
		Pre-Plateau Detection	79.45%	1158	1.55
		Random Detection (100%)	100%	1158	1.93
ResNet9	1508	Central Mode	2.20%	1338	0.04
		Random Detection	2.20%	672	0.04
		Pre-Plateau Detection	16.27%	1508	0.31
		Random Detection (100%)	100%	1508	1.93
ResNet18	826	Central Mode	2.11%	822	0.04
		Random Detection	2.11%	496	0.04
		Pre-Plateau Detection	14.47%	826	0.27
		Random Detection (100%)	100%	826	1.93
VGG11	819	Central Mode	1.77%	795	0.03
		Random Detection	1.77%	518	0.03
		Pre-Plateau Detection	16.41%	819	0.32
		Random Detection (100%)	100%	819	1.93
VGG16	894	Central Mode	1.23%	891	0.02
		Random Detection	1.23%	510	0.02
		Pre-Plateau Detection	13.33%	894	0.26
		Random Detection (100%)	100%	894	1.93
MobileNet	1614	Central Mode	14.21%	1122	0.27
		Random Detection	14.21%	907	0.27
		Pre-Plateau Detection	53.39%	1614	1.03
		Random Detection (100%)	100%	1614	1.93
MNIST					
ResNet9	527	Central Mode	0.36%	527	0.01
		Random Detection	0.36%	518	0.01
		Pre-Plateau Detection	3.11%	527	0.05
		Random Detection (100%)	100%	527	1.93
ResNet18	581	Central Mode	0.29%	581	0.01
		Random Detection	0.29%	483	0.01
		Pre-Plateau Detection	3.64%	581	0.07
		Random Detection (100%)	100%	581	1.93
VGG11	661	Central Mode	0.22%	659	0.01
		Random Detection	0.22%	511	0.01
		Pre-Plateau Detection	3.61%	661	0.06
		Random Detection (100%)	100%	661	1.93
VGG16	555	Central Mode	0.27%	551	0.01
		Random Detection	0.27%	515	0.01
		Pre-Plateau Detection	3.28%	555	0.05
		Random Detection (100%)	100%	555	1.93
LeNet	484	Central Mode	0.84%	346	0.02
		Random Detection	0.84%	297	0.02
		Pre-Plateau Detection	9.01%	436	0.18
		Random Detection (100%)	100%	484	1.93

workloads for FI campaigns. Future work should aim to refine image selection techniques to further enhance the accuracy and efficiency of DNN fault assessments and explore their applicability in crafting new test cases for advanced fault detection methodologies.

ACKNOWLEDGE

This work has been funded by the ANR France 2030 AdaptING project "ANR-23-PEIA-0009".

REFERENCES

[1] A. Ruospo *et al.*, "A survey on deep learning resilience assessment methodologies," *Computer*, 2023.

[2] A. Ruospo *et al.*, "Assessing convolutional neural networks reliability through statistical fault injections," in *2023 Design, Automation and Test in Europe Conference I& Exhibition (DATE)*, 2023.

[3] W. Li *et al.*, "Adversarial testing: A novel on-line testing method for deep learning processors," in *2023 IEEE 32nd Asian Test Symposium (ATS)*, 2023.

[4] M. H. Ahmadilivani *et al.*, "Special session: Approximation and fault resiliency of dnn accelerators," in *2023 IEEE 41st VLSI Test Symposium (VTS)*, 2023.

[5] A. Bosio *et al.*, "A reliability analysis of a deep neural network," in *2019 IEEE Latin American Test Symposium (LATS)*, 2019.

[6] C. Qian *et al.*, "A survey of bit-flip attacks on deep neural network and corresponding defense methods," *Electronics*, 2023.

[7] ISO and IEC, "Artificial intelligence — functional safety and ai systems," ISO/IEC JTC 1/SC 42, Geneva, CH, Technical Report, Jan. 2024.

[8] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018.

[9] S. Pappalardo *et al.*, "A fault injection framework for ai hardware accelerators," in *2023 IEEE 24th Latin American Test Symposium (LATS)*, 2023.

[10] Q. Hu *et al.*, *Evaluating the robustness of test selection methods for deep neural networks*, 2023.

[11] C. Zhao *et al.*, "Can test input selection methods for deep neural network guarantee test diversity? a large-scale empirical study," *Information and Software Technology*, 2022.

[12] T. Borovicka *et al.*, "Selecting representative data sets," in *Advances in Data Mining Knowledge Discovery and Applications*, A. Karahoca, Ed., Rijeka: IntechOpen, 2012, ch. 2.

[13] R. Kennard *et al.*, "Computer aided design of experiments," *Technometrics*, 1969.

[14] P. Schober *et al.*, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia I& Analgesia*, 2018.

[15] M. Weiss *et al.*, "Simple techniques work surprisingly well for neural network test prioritization and active learning (replicability study)," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA '22, ACM, Jul. 2022.

[16] M. Paul *et al.*, "Deep learning on a data diet: Finding important examples early in training," in *Advances in Neural Information Processing Systems*, M. Ranzato *et al.*, Eds., Curran Associates, Inc., 2021.

[17] A. Mao *et al.*, *Cross-entropy loss functions: Theoretical analysis and applications*, 2023.

[18] S. Kundu *et al.*, "Special session: Effective in-field testing of deep neural network hardware accelerators," in *2022 IEEE 40th VLSI Test Symposium (VTS)*, 2022.

[19] S. Kundu *et al.*, "Toward functional safety of systolic array-based deep learning hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2021.

[20] F. Meng *et al.*, "A self-test framework for detecting fault-induced accuracy drop in neural network accelerators," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021.

[21] F. Meng *et al.*, "Exploring image selection for self-testing in neural network accelerators," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022.

[22] M. Barbareschi *et al.*, "Investigating the resilience source of classification systems for approximate computing techniques," *IEEE Transactions on Emerging Topics in Computing*, 2024.

[23] S. A. El-Sayed *et al.*, "Compact functional testing for neuromorphic computing circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[24] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[25] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[26] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[27] A. G. Howard *et al.*, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017.

[28] M. Sandler *et al.*, *Mobilenetv2: Inverted residuals and linear bottlenecks*, 2019.

[29] A. Krizhevsky, *Cifar-10 dataset*, https://www.cs.toronto.edu/~kriz/cifar.html, Accessed on April 11, 2024, 2009.

[30] Y. LeCun *et al.*, *The mnist database of handwritten digits*, http://yann.lecun.com/exdb/mnist/, Accessed on April 11, 2024, 1998.

[31] G. Gavarini *et al.*, "Sci-fi: A smart, accurate and unintrusive fault-injector for deep neural networks," in *2023 IEEE European Test Symposium (ETS)*, 2023.

[32] "Spearman Rank Correlation Coefficient," en, in *The Concise Encyclopedia of Statistics*, New York, NY: Springer, 2008.