DOC: Detection of On-Line Failures in CNNs

DOC: Detection of On-Line Failures in CNNs

V. Turco, N. Bellarmino, A. Ruospo, R. Cantoro, E. Sanchez
Politecnico di Torino, DAUIN, Torino, Italy

***Abstract*—The integration of Artificial Intelligence (AI) in safety-critical systems raises concerns about reliability, particularly due to the inherent uncertainty of AI algorithms and the complexity of modern hardware, compromising billion of transistors. Existing solutions, such as Algorithm-Based Fault Tolerance often focus on running detection algorithms after every inference, introducing a not negligible overhead in the detection phase. This paper introduces a two-phase fault detection technique for Convolutional Neural Networks (CNNs) with floating-point precision. The first phase identifies easily detectable faults (such as those stemming from a bit-flip on the 30th of a floating-point representation), while the second one targets hard-to-detect critical faults—those producing a wrong prediction but having no visible effect during faults' propagation. Although these faults constitute only 1.4% of all critical faults, their detection is crucial for ensuring system reliability. Validated on the CIFAR-10 dataset with a ResNet-20 model, the proposed method achieves up to 99.67% coverage of critical inferences while maintaining moderate computational overhead. This lightweight, real-time solution enhances the robustness of CNNs in safety-critical applications.**

***Index Terms*—Fault Injection, Reliability, Convolutional Neural Network, Bit-Flip, Online Test.**

I. Introduction

Convolutional Neural Networks (CNNs) have become a cornerstone in various machine learning applications. These models are often deployed in embedded devices and edge computing environments, where they are subjected to unpredictable conditions such as hardware faults. In these scenarios, the reliability of CNNs under fault conditions is paramount. This is especially true when errors arise from faults such as bit flips in memory [1]. These faults can cause misclassifications and undermine the trustworthiness of the system. Current approaches to fault detection in CNNs typically focus on hardware redundancy techniques [2], [3]. However, these methods can be computationally expensive or may require significant changes to the architecture, making them less suitable for resource-constrained environments. Additionally, existing fault detection strategies often lack mechanisms for real-time anomaly detection during inference, limiting their applicability in real-world deployment scenarios. In this paper, we propose an online failure detection technique that is specifically designed to monitor CNNs during inference. The approach detects in a first phase anomalies arising from faults by analyzing the sum of the embeddings (Embedding Sum) after each convolutional layer (easy-to-detect). A key feature of this approach is its use of thresholds computed through fault injection experiments, which are employed to flag potential faults in the network's behavior. Additionally, the proposed method incorporates a second phase involving an additional control system based on the logits of the final layer. This system uses the difference between the top-1 and top-2 logits (Top2Diff, [2]) as a secondary indicator of fault presence, enabling the deployment of test image libraries to validate or invalidate inferences flagged as dangerous (hard-to-detect). This two-stage approach facilitates fault detection in both the early and final layers of the network. To validate the technique, we conduct experiments on the CIFAR-10 dataset using a pre-trained ResNet-20 model. Through fault injection simulations, we demonstrate the effectiveness of the proposed approach in identifying faulty inferences, achieving coverages in excess of 99% with low computational overhead. The results highlight the technique's potential for On-Line failure detection in CNNs, making it a promising solution for enhancing the reliability of deep learning models deployed in fault-prone environments.

The paper is organized as follows: Section II provides a brief background on related works, highlighting the motivations behind this study. Section III presents the proposed approach, followed by Section IV, which details the case study. Section V reports the experimental results, while Section VII concludes by summarizing the main contributions and outlining potential future research directions.

II. Related Work

Related works on Fault Detection (such as Algorithm-Based Fault Tolerance, ABFT [4], [5]) are based on running detection algorithms after every inference. Fault detection scheme are mainly based on crafting ad-hoc images able to detect critical faults. Gradient-based adversarial image crafting [6] has been extensively investigated, in addition to methods based on considering the probability vector of the DNN [7]. Some methods have aimed to enhance the reliability of neural networks starting since training phase [8]. Methods based on Embedding Sum and logits analysis techniques have been investigated in anomaly detection and outlier detection within machine learning models. For example, work [9] demonstrates how various tensor-related metrics, applied to output feature maps (OFMs), can provide valuable insights for fault detection. The Embedding Sum is computed by summing the elements of the OFMs generated during the processing of an input. Conversely, the term logit refers to the numerical values produced by the final layer of a neural network, immediately prior to the application of an activation function (e.g., Softmax), which transforms these values into probabilities. For instance, in [2], a similar approach is applied to detect abnormal behaviors

978-1-6654-7763-5/25/$31.00 ©2025 IEEE

in deep learning models by analyzing their embeddings and output, with a particular focus on improving the detection of faulty or anomalous behavior. In detail, they introduce in the first stage a *fmap level resilience* technique (FLR), and at a later time an *inference level resilience* technique (ILR). The FLR system enhances robustness by duplicating filters associated with the most vulnerable OFMs [2], introducing redundancy to produce two copies. Similarly, the ILR system leverages redundancy to protect against errors in vulnerable inferences. The combination of these two techniques (FILR) is evaluated across various models and datasets, achieving critical inference coverage of up to 99.78%. The associated overhead varies depending on the neural model utilized, ranging from 20% to 90%. In this work, we propose a similar approach, leveraging OFMs and Top2Diff computed on the model's logits to achieve high detection coverage with minimal overhead, without introducing redundancy into the system. Specifically, we design a two-tier control system: the first targets *easy-to-detect* critical inferences by utilizing a tensor-related metric for the early detection of permanent faults, as discussed in [9]. The second control system employs Top2Diff to determine whether to deploy test images [10] for identifying hard-to-detect critical inferences.

III. Proposed Approach

The proposed mechanism is designed to monitor in the field CNN applications during inference (with weights stored in FP32 format) and identify hazardous anomalies caused by faults, such as bit flips occurring in memories. The main goal of the solution is to prioritize the in-field detection of faults leading to failures, to avoid wrong CNN predictions. In this work, we categorize critical faults (also known as Silent Data Corruption SDC-1) in two classes: **easy-to-detect** and **hard-to-detect**. The first class covers all those faults affecting the most significant bit of the exponent part (MSB), that are easy to recognize as their occurrence produces out-of-range values (e.g., NaN). The second class is represented by all the remaining faults producing failures (i.e., wrong predictions) that are hard to detect, as their effect is not easily discernible in the CNN computation. So, the proposed methodology works in two phases: first it discards CNN inferences corrupted by **easy-to-detect** random-hardware faults with a low-cost metric applied to intermediate layers; then it exploits test images to recognize and classify the **hard-to-detect** faults

The technique is developed to operate in real-time, with as little as possible computational overhead, making it suitable for deployment in resource-constrained environments where CNNs might be susceptible to random-hardware faults. The design of the approach consists of two different steps:

- **Pre-deployment Analysis**: A structured analysis is conducted using metrics and FI campaigns to derive specific parameters and thresholds. These specifications are utilized to monitor the neural network during the online fault detection:
  - **Internal Representation Analysis**: it introduces the concept of Embedding Sum between model layers as a metric to capture the internal representations of the CNN. This metric is used to compute an initial threshold t_1 that serves as a basis for performing fault detection of **easy-to-detect** faults.
  - **Top2Diff Logits Analysis**: this step involves analyzing the probability distributions of faulty logits to design a specific threshold t_2 for the model's output, based on the difference between the top-1 and top-2 predictions, (Top2Diff). This second threshold will be useful to detect **hard-to-detect** faults.
  - **Image test libraries (ITLs) Selection**: based on $t2$, carefully selected test images are run to enhance the percentage of faulty inferences detected.
- **In-Field Detection Methodology**: This step is performed online and represents the in-field part of the methodology. It is further divided into two additional phases:
  - **Phase 1**: In this phase, Embedding sum it compared with t_1 threshold to identify *easy-to-detect* faults.
  - **Phase 2**: The objective of this phase is to mitigate faults not detected during the first phase. An additional verification step based on comparing the Top2Diff metric of propagated inferences and the threshold t_2 is done. This analysis aims to identify particularly sensitive images that, in the event of a critical fault, may result in inferences previously misclassified as false positives. For these identified images, a critical checking system utilizing predesigned ITLs is activated.

A. *Pre-deployment Analysis*

1) Internal Representation Analysis:

After each convolutional layer i, Embeddings Sum (Σ_i) is computed. These sums encapsulate the overall response of the layer and can serve as a valuable metric for detecting faults. The rationale behind using embedding sums is that faults, such as bit flips, will typically cause significant deviations in the layer's activations, altering the distribution of embedding sums. To set a baseline for normal operation, we first compute the distribution of embedding sums under fault-free conditions using a representative dataset such as in Figure 1. The distribution is modeled using standard statistical techniques, and the layer embedding sums distribution with the highest mean is selected as the reference. For a reference distribution, a threshold is calculated based on the mean (μ) and standard deviation (σ) to define the boundary between normal and anomalous behavior. The threshold is computed as follows:

$$t_1 = \mu + k \cdot \sigma \tag{1}$$

where:

- μ: The mean of the embedding sum distribution for the chosen layer, representing the average value of the data points.
- σ: The standard deviation of the embedding sum distribution for the layer, which indicates the spread or variability of the data around the mean.

- k: This parameter determines how far the threshold (t_1) is set from the mean. Specifically, k is a scaling factor for σ, and its choice influences the threshold's position relative to the mean:
  - A larger value of k places the threshold farther from the mean, making it less strict and reducing the likelihood of false positives (i.e., classifying normal instances as anomalies).
  - A smaller value of k places the threshold closer to the mean, increasing sensitivity to anomalies but also raising the risk of false positives.

The value of k is chosen based on the desired balance between sensitivity and specificity in detecting anomalies.

Data points with Embedding sum above t_1 are classified as outliers, and the the inference is flagged as critical. The threshold is empirically determined through the golden model experiments, with careful consideration to achieve a balance between detection sensitivity and false positive rates.

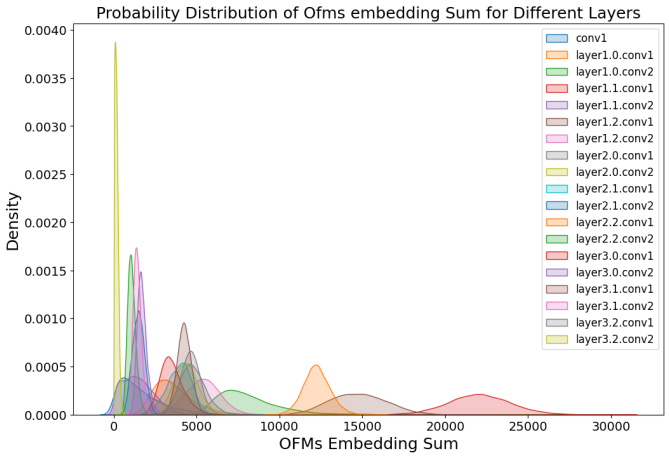

Fig. 1. Layers Embedding Sum distribution.

*2) **Top2Diff Logit Analysis**:*

While Embedding Sum analysis is an effective early detection mechanism for the easy-to-detect faults, certain faults may escape and not being noticed.

For example, Fig. 2 illustrates the distribution of faulty embedding sums that closely resemble those of the golden inferences, resulting in critical outputs without significantly deviating from the expected metric values. In the figure, red dots represent embedding sums for faulty inferences leading to CNN wrong predictions, while blue dots correspond to faulty embeddings that do not impact the final prediction. To detect red dots (the so-called *hard-to-detect* faults), the proposed method incorporates a secondary check based on the output logits of the final CNN layer, combined with the execution of ITLs for enhanced fault detection of hard-to-detect faults.

Fig. 3 illustrates the impact of an FI campaign on a CNN, focusing on Top2Diff. The analysis reveals that only faulty inferences with a low Top2Diff are prone to deviating from the golden (fault-free) prediction. Accordingly, to identify faults that affect the model's decision-making, even when embedding

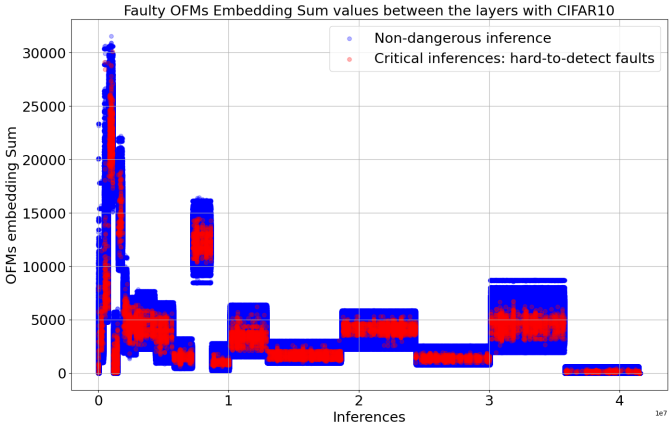

Fig. 2. Faulty OFMs Sum metric values between the layers with CIFAR10.

sum analysis fails to indicate anomalies, the goal of this approach is to trigger an alert when Top2Diff does not exceed a designed threshold, t_2. This alert initiates further diagnostic procedures, such as additional tests on the input image or more comprehensive fault detection routines, to ensure the reliability of the system. t_2 can be derived from a FI campaign conducted on the selected CNN model using a representative dataset, and it is specifically designed to identify sensitive inferences by analyzing the distribution of Top2Diff values for faulty inferences. To find the best t_2, an optimization is applied that employs an empirical approach using a greedy algorithm based on exhaustive search. The optimization aims to maximize accuracy, defined as the proportion of correct classifications between the critical (red) and non-critical (blue) sets shown in Fig. 3. Ultimately, in real-world scenarios, we expect that if the Top2Diff metric falls below the predefined threshold t_2, it signals a high degree of uncertainty in the model's prediction, often indicative of fault sensitivity.

*3) **ITLs selection**:*

The ITL is generated by a set-covering algorithm applied to the set U of test images that triggered at least one critical faults, resulting in a critical inference (Algorithm 1). This task that can be efficiently performed using the data collected on U during a FI campaign.

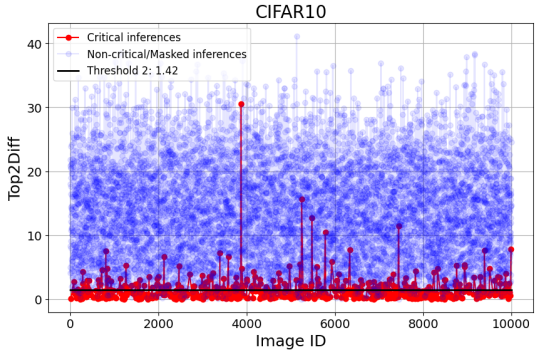

Fig. 3. Top2Diff metric measured after fault injection campaigns.

Algorithm 1 Greedy Set Cover for Fault Coverage Using Images

Input: U (Set of images), S (Collection of subsets, $S_i \subseteq U$ where each S_i covers a set of faults)
Output: C (Subset of S that covers all faults)
$C \leftarrow \emptyset$ ▷ Initialize the cover set C for selected images
while there are still uncovered faults **do**
  $S_j \leftarrow \arg\max_{S_i \in S} |S_i \cap U|$ ▷ Select the image subset S_j that covers the most uncovered faults
  $C \leftarrow C \cup \{S_j\}$ ▷ Add selected subset S_j to C
  $U \leftarrow U \setminus S_j$ ▷ Remove the faults covered by S_j from U
end while
return C

B. *In-field detection methodology*

The overall workflow of the fault detection methodology is as follows:

*1) **Phase 1 (Easy-to-detect faults detection):***

During inference, after each convolutional layer i, Σ_i is computed in real-time and compared against the predefined threshold, t_1. If the sum exceeds the threshold, the inference is flagged as faulty. This early detection mechanism permits identifies faults before they propagate through the network, thereby preventing incorrect classifications in subsequent layers. Additionally, it enable saving energy, time, and computational resources by halting further processing of faulty inferences. This first control in depicted in Fig. 4.

*2) **Phase 2 (Hard-to-detect faults detection):***

For every inference, the Top2Diff is calculated. If the metric does not exceed t_2, an alert is generated, and the ITL is executed. If even one of these test images results in a prediction differing from the golden (fault-free) reference, the inference is flagged as critical. This second control is depicted in Fig. 5.

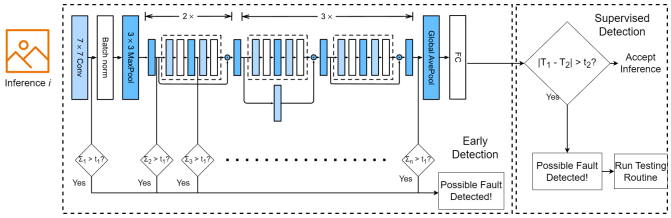

Fig. 4. Phase 1: workflow for the easy-to-detect fault detection based on t_1.

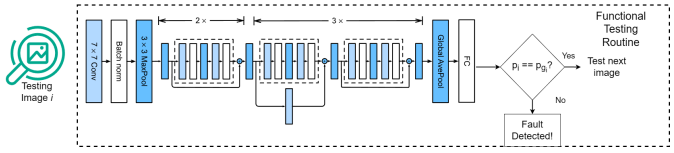

Fig. 5. Phase 2: workflow for hard-to-detect fault detection through the ITLs.

IV. Case of Study

Experiments were conducted using a pre-trained ResNet-20 architecture [11], validated on CIFAR-10 [12]. To validate the methodology, a fault injection campaign was performed using a Statistical Fault Injection (SFI) process [13]. Permanent random-hardware faults were modeled as stuck-at faults, where a fault was injected by flipping a random bit in a randomly selected network synaptic weights. A total of 4,160 stuck-at faults were injected, achieving an estimation with an error margin of $e = 2\%$ and a confidence level of 99%.

To validate the proposed approach described in Section III, we utilized the 10,000 images available in the CIFAR-10 test set. To ensure a robust analysis with a limited dataset, a randomized cross-validation technique was employed.

The dataset was randomly partitioned 10 times into:

- **Training Subsets**: A subset of 8,000 images, used to calculate and analyze the thresholds t_1 and t_2, as well as to define the ITL.
- **Validation Subsets**: A subset of 2,000 images, used to validate the methodology.

Experimental results were calculated by averaging the outcomes across the 10 case histories. All experiments were performed in Python using PyTorch tools for the DL models. Experiments run on a server equipped with an Intel® Core™ i9-13900K CPU @3.0GHz x 24, 64GB of RAM, and an Nvidia® RTX A4000 GPU.

V. Experimental Results

Phase 1

The initial experimental results focus on the application of the threshold t_1, which is computed using the training subset as detailed in Section III-A1. The computation of t_1 follows the formula provided in Eq. 1. In this study, $k = 6$ was empirically chosen, with the aim of prioritizing the accurate detection of critical faults while allowing for a small, controlled rate of false positives to ensure system reliability.

Evaluation is performed by constructing a matrix that quantifies:

- True Positives (TPs): Critical inferences successfully identified.
- False Negatives (FNs): Faulty inferences that are not classified as dangerous.
- True Negatives (TNs): Non-critical inferences correctly identified.
- False Positives (FPs): Non-critical inferences incorrectly flagged as critical.

Fig. 6 illustrates the confusion matrix summarizing the performance of the initial check using t_1. This was evaluated considering only the **faulty inferences generated during the FI campaign**.

The first step successfully identifies and interrupts 2.15% of faulty inferences, which represent approximately 98.6% of all critical faulty inferences and correspond to the easy-to-detect faults. A significant proportion, 96.77% of non-dangerous faulty inferences **TNs**, are correctly allowed to propagate to

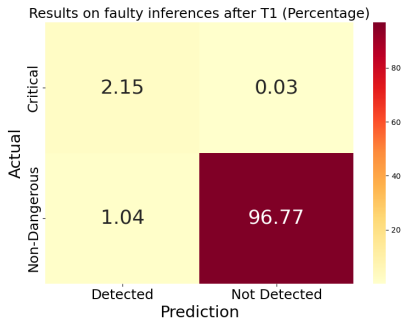

Fig. 6. Confusion matrix of the t_1 effectiveness.

the final layer without interruption. A small percentage, 1.04% of **FNs** corresponds to inferences with final Top-1 predictions identical to the golden (fault-free) version but are interrupted by the control system based on the t_1 threshold. These cases represent the first "cost" of the technique. The remaining 0.03% (**FPs**) represent the hard-to-detect faults that remain unmitigated producing a wrong prediction, and correspond to the red dots in Fig. 2. The next phase involving t_2 and test images is specifically designed to address these remaining cases classified as FPs in the confusion matrix.

Phase 2

1) Logits Comparison

At this stage, the Top2Diff of the uninterrupted faulty inferences in phase 1 is calculated on the model's logit.

TABLE I
t_2 THRESHOLD FILTERING RESULTS

t_2 Triggered images	5.5%
Accuracy of t_2	90.0%
Filtering Effectiveness	66.89%
Total Critical Faults	645
Subset Fault Overlap	100%

Table I summarizes the efficacy of the approach at this stage:

- **Triggered Images** represents the percentage of images identified as sensitive within the 2000-image validation subset, based on the threshold t_2 check.
- **Accuracy of** t_2 denotes the proportion of identified triggered images that result in critical inferences during the FI campaign.
- **Filtering Effectiveness** evaluates the filtering process's ability to capture critical images within the 2000-image validation subset. Specifically, it measures the percentage of critical images (which constitute 90% of the filtered images) that are correctly identified and flagged for further analysis.
- **Total Critical Faults** quantifies the total number of faults capable of inducing critical perturbations in the inference process within the 2000-image validation subset.
- **Subset Fault Overlap** highlights the critical faults that impact images detected by t_2 and compares them with the faults that affect the entire 2000-image validation subset. In all 10 case studies analyzed, these faults exhibited a 100% overlap.

2) ITL Fault Detection

In this final stage, the ITL is evaluated for each image triggered by t_2 in the previous stage. The final results regarding the coverage of critical inferences and dangerous faults at the end of Phase 2 are presented in Table II. In the table, alongside critical inferences coverage, a second type of coverage is also considered: the coverage related to the number of identified faults, referred to as dangerous faults, that trigger critical inferences. The coverage values are categorized based on the number of test images selected for the final control stage. Constraints can be introduced into the greedy set cover algorithm to limit the maximum number of test images used. According to the experimental data, the optimal number of images required to cover all the faults in the training dataset is around 21. When applied to the validation dataset, these images cover 92.5% of the 645 faults listed in Table I and 99.67% of the critical inferences. In more conservative cases, where the goal is to minimize computational cost by using fewer test images and reducing the technique's overhead, the coverage values evolve as follows: starting with an inference coverage of 98.53% and dangerous fault coverage of 17.22% at the end of Phase 1, the coverage increases to 99.62% for critical inferences and 86.75% for dangerous faults when using only 5 test images. Beyond 9 test images, the use of additional test images provides minimal additional benefit.

TABLE II
FINAL COVERAGE OF CRITICAL INFERENCES AND DANGEROUS FAULTS

	Test Images Used	Critical Inferences Coverage (%)	Dangerous Faults Coverage (%)
Phase 1	0	98.53	17.22
Phase 2	1	99.11	51.32
	2	99.37	67.70
	3	99.51	77.94
	4	99.59	83.78
	5	99.62	86.75
	6	99.64	88.96
	7	99.65	90.18
	8	99.65	91.03
	9	99.66	91.67
	...	...	...
	21	99.67	92.50

VI. OVERHEAD ANALYSIS

It is possible to estimate the total overhead, denoted as H, associated with the proposed technique.

H can be expressed as the sum of two separate terms: A and B. The term A represents the overhead in case a fault does occur, while the term B captures the overhead in the case that a fault does not occur. Both terms depend on various factors, including the number of test images I used, the conditional probabilities of the thresholds results (t_1 a t_2)

and the probability of occurrence or non-occurrence of a fault in a real-world scenario.

$$H = A + B \; with$$

$$A = P(F) \cdot [I \cdot P(t_2 \mid F) + P(FP_{t_1} \mid F)]$$

$$B = P(\overline{F}) \cdot \left[I \cdot P(t_2 \mid \overline{F}) + P(FP_{t_1} \mid \overline{F})\right]$$

Where:

- $P(F)$ and $P(\overline{F})$ represent the probabilities of a fault occurring or not occurring, respectively, in a real-world scenario.
- I is the number of images selected for fault detection, which are our operational cost in terms of inferences for each sensitive image triggered by t_2.
- $P(t_2 \mid F)$ and $P(t_2 \mid \overline{F})$ define the probability that the designed t_2 on the Top2Diff of a faulty fault-free inference will trigger the ITL-based control system. In this case, we experimentally calculated that $P(t_2 \mid F)$ is 5.5%, while for fault-free, $P(t_2 \mid \overline{F})$ is 5.525%.
- $P(FP_{t_1} \mid F)$ and $P(FP_{t_1} \mid \overline{F})$ represent the probabilities of a faulty and fault-free inference being erroneously interrupted by the threshold t_1, corresponding to the false positives in phase 1. $P(FP_{t_1} \mid F)$ accounts for 1.04% already introduced in the confusion matrix shown in Figure 6, while $P(FP_{t_1} \mid \overline{F})$ is negligible due to the $k = 6$ term in the computation of t_1.

By substituting all experimentally calculated values, the overhead terms can be expressed as follows:

$$A = P(F) \cdot [I \cdot 0.055 + 0.0104]$$

$$B = P(\overline{F}) \cdot \left[I \cdot 0.05525 + P(FP_{t_1} \mid \overline{F}) \approx 0\right] = P(\overline{F}) \cdot I \cdot 0.05525$$

Since $P(F)$ and $P(\overline{F})$ represent complementary events, their sum satisfies $P(F) + P(\overline{F}) = 1$. Therefore, the total overhead H lies within the following bounds:

$$I \cdot 5.525\% \leq H \leq (I \cdot 5.5\% + 1.04\%)$$

This means the overhead H is linearly dependent on the number of test images I. The lower bound represents the minimal cost per image, while the upper bound accounts for the additional false-positive cost. Consequently, increasing the number of test images directly increases the overall cost of the technique.

VII. Conclusions and Future Works

This paper presented a safety mechanism for in-field fault detection in CNN-based systems. The main objective was to identify dangerous anomalies caused by permanent faults and to determine whether such anomalies had a significant impact on the network's final prediction. Experimental results on the validation data show that the proposed technique is effective in detecting critical inferences in CNNs with floating-point precision, granting more than 99% coverage for critical inferences. In terms of operational cost, the overhead introduced by the proposed approach strongly depends on the number of images used, but the experimental results suggest that a few images can still provide good coverage with a good trade-off between accuracy and computational cost. Looking ahead, the proposed technique will be extended to different models and datasets beyond CIFAR-10 to evaluate its robustness and adaptability across diverse contexts. Additionally, a version compatible with quantized CNNs is under development. Furthermore, we believe the technique's performance can be significantly improved by generating ad hoc-test test images designed specifically to highlight failures in inferences.

VIII. Acknowledgment

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] A. Ruospo, E. Sanchez, L. M. Luza, L. Dilillo, M. Traiola, and A. Bosio, "A survey on deep learning resilience assessment methodologies," *Computer*, vol. 56, no. 2, pp. 57–66, 2023.

[2] A. Mahmoud *et al.*, "Optimizing selective protection for cnn resilience," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 127–138.

[3] Y. Owada, Y. Tomioka, and H. Saito, "Dual modular redundancy unit of convolutional layer for low-cost and reliable cnns," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022, pp. 379–384.

[4] M. Liu, L. Xia, Y. Wang, and K. Chakrabarty, "Fault tolerance for rram-based matrix operations," in *2018 IEEE International Test Conference (ITC)*, 2018, pp. 1–10.

[5] M. Safarpour, R. Inanlou, and O. Silvén, "Algorithm level error detection in low voltage systolic array," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 2, pp. 569–573, 2022.

[6] W. Li, Y. Wang, K. Zou, H. Li, and X. Li, "Adversarial testing: A novel on-line testing method for deep learning processors," in *2023 IEEE 32nd Asian Test Symposium (ATS)*, 2023, pp. 1–6.

[7] S. Kundu, S. Banerjee, A. Raha, and K. Basu, "Special session: Effective in-field testing of deep neural network hardware accelerators," in *2022 IEEE 40th VLSI Test Symposium (VTS)*, 2022, pp. 1–4.

[8] N. Bellarmino, A. Bosio, R. Cantoro, A. Ruospo, E. Sanchez, and G. Squillero, "Investigating on Gradient Regularization for Testing Neural Networks," in *LOD 2024 : 10th International Conference on machine Learning, Optimization and Data science*, Riva de Sole, Italy, Sep. 2024. [Online]. Available: https://hal.science/hal-04604264

[9] V. Turco, A. Ruospo, E. Sanchez, and M. S. Reorda, "Early detection of permanent faults in dnns through the application of tensor-related metrics," in *2024 27th International Symposium on Design Diagnostics of Electronic Circuits Systems (DDECS)*, 2024, pp. 13–18.

[10] A. Ruospo *et al.*, "Image test libraries for the on-line self-test of functional units in gpus running cnns," in *2023 IEEE European Test Symposium (ETS)*, 2023, pp. 1–6.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[13] 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, pp. 1–6.