Analysis and Mitigation of Soft-errors in GPU-accelerated Hyperspectral Image Classifiers

# Analysis and Mitigation of Soft-errors in GPU-accelerated Hyperspectral Image Classifiers

Sergiu-Mohamed Abed*, Juan-David Guerrero-Balaguera*, Josie E. Rodriguez Condia*, Gianluca De Lucia†, Marco Lapegna‡, Matteo Sonza Reorda*

* Politecnico di Torino - Department of Control and Computer Engineering (DAUIN)

† CASD - Scuola Superiore Universitaria, Rome, Italy

‡ University of Naples Federico II - Department of Mathematics and Applications

***Abstract*—[1] This work assesses the reliability of a hyperspectral image classifier for edge devices under transient faults by using a fine-grain strategy based on the *Hardware Injection Through Program Transformation* (*HITPT*) technique. The results identified the most vulnerable software parts and the corruption effects due to hardware faults (from 5.1% to 100.0% of accuracy drop). Then, the results supported the adoption of a selective-hardening software mechanism (based on the *Duplication with Comparison* strategy) to effectively mitigate the most critical effects under limited costs.**

***Index Terms*—Edge Computing, GPU, Hyperspectral Image Classification, AI, Reliability, Hardware faults, Soft errors.**

## I. Introduction

In the recent years, *Artificial Intelligence* (AI) has seen a widespread adoption in edge computing systems present in industries such as precision agriculture and automotive. To address the computational complexity of AI applications, edge computing systems integrate accelerators such as *Graphics Processing Units* (GPUs) to speed up their processing [1]. In particular, embedded computer vision applications (e.g., *Hyperspectral Imaging* or HSI [2]) leverage both AI algorithms and hardware accelerators for successful deployment.

The reliability of AI-powered edge systems is crucial to ensure their operation in the presence of faults, especially when an error can cause costly losses (e.g., misclassifications in precision agriculture). For this purpose, reliability evaluation techniques have been developed to identify vulnerable (hardware/software) parts in a system. The results are later used to adopt fault countermeasure mechanisms [3]. Unfortunately, the increasing computational complexity, the large amount of data processed by edge applications, and the transistor density of the new hardware platforms make the adoption of traditional reliability assessment strategies unfeasible [4].

In this work, we assess the reliability of an HSI classifier for edge computing platforms under hardware faults. More in detail, we adopted the Hardware Injection Through Program Transformation (HITPT) strategy as an efficient method to evaluate different GPU-based systems and analyze the impact that transient faults occurring in the systems have on the HSI classifier. The framework we developed for this purpose is available in a public repository[2].

Our results indicate that some routines corresponding to the pre-processing stage in the classifier are highly sensitive to corruptions independently of the underlying hardware. Based on these results, we developed and validated a software-based hardening mechanism (using a *Duplication with Comparison* approach [5]) to reduce fault impacts on these highly sensitive kernels.

[1]This work has been partially supported by the European Commission through the HORIZON-MSCA-2023-DN-01 TIRAMISU project under grant 101169378 and by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing.

## II. Background

### A. Reliability assessment strategies

Reliability analysis of AI-powered systems is crucial for identifying vulnerabilities and understanding fault behavior. In practice, three experimental strategies can be used to assess the system's reliability: *i) Physical experiments*, *ii) Simulation-based experiments*, and *iii) Software-based experiments*.

The first strategy, (*Physical/beam experiments*), exposes the system to external sources (e.g., radiation or electromagnetic) to induce faults during operation, effectively assessing overall reliability against transient faults (e.g., *Single Event Upsets* or SEUs) [6]. However, this approach is costly, requiring specialized facilities, and can hardly identify the most vulnerable hardware structures due to observability restrictions on the system [7]. On the other hand, *Simulation-based experiments* consist of a set of fault simulation campaigns injecting one or several faults into a representative system model (Register-Transfer Level, RTL). This method can provide fine-grain identification of the most vulnerable hardware/software modules in a system. However, a representative and accurate model is required. Unfortunately, the large evaluation times and the required computational power restrict the adoption of this approach for complex and large systems, such as AI-powered ones (e.g., evaluating a simple CNN on a GPU model might require more than 10,000 days [8]). The third strategy (*Software-based experiments*), using the HITPT approach, instruments the application binary code to emulate hardware faults. Then, the instrumented code is executed on the platform to determine the fault impacts during the real application's operation. The HITPT strategy is fast and can effectively analyze fault effects on the system's data path (e.g., memories and execution units).

[2]https://github.com/sergiuabed/nvbitfi_v2/tree/main

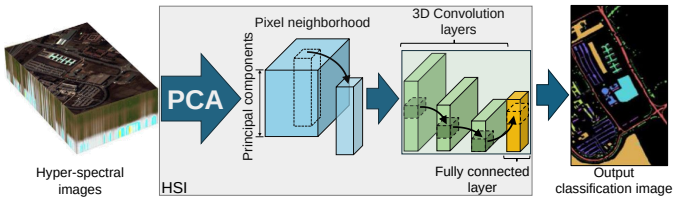

Fig. 1. Scheme of HSI classifier comprising PCA and AI algorithms.

### B. Hyperspectral Image classification on edge computing

Hyperspectral Images (HSI) are data obtained through sensors such as hyperspectral cameras. A single HSI is composed of spatial pixels, each representing a specific geographical location, and has a spectral depth of several wavelength bands, creating a cube of hyperspectral data. The collected information is therefore processed by a *HSI classifier* that assigns labels per image pixel, which commonly requires considerable computational resources.

For such reason, modern strategies of HSI classification involve two main stages [9]: *i)* a pre-processing stage (e.g., *Principal Component Analysis*, PCA) that removes redundant features from HSI data, and *ii)* a classification stage through a neural network. To speed-up the execution, both steps may be performed on hardware accelerators, such as GPUs.

## III. Proposed strategy to evaluate and mitigate fault effects in HSI applications

We propose using a hardware-aware technique to characterize *Single-Event Upsets* (SEUs) possibly arising in the system's hardware accelerator and assess the reliability of an HSI application. For this purpose, we employ a fine-grain software-based error injection framework (*NVBitFI* [10]) to identify the most susceptible code blocks in the application. Our analysis is then used to improve the reliability by developing a selective software-based hardening mechanism.

### A. Application Profiling

First, we dynamically profile the fault-free execution of the HSI application to trace and collect the operative code blocks (i.e., *CUDA kernels*) and their instructions, which are then used to support the reliability analysis in consecutive steps.

### B. Fault Assessment

Then, we use the software-based HITPT strategy to inject errors into the application's code and evaluate the effects of faults affecting the underlying hardware (i.e., the GPU in our case). In detail, fault injection is performed by instrumenting the application code; each corrupted/mutated instruction (e.g., *FMUL*) mimics a fault effect inside one of the hardware structures (e.g., the register file or the core executing *FMUL*). The code instrumentation is automatically performed based on a profiling procedure of the HSI application.

Several fault injection campaigns, using the adapted NVBitFI framework [10], characterize the HSI application by injecting faults in a GPU's targeted hardware structure. In particular, the framework randomly forces individual soft errors per each application's execution (i.e., corrupting one instruction according to a specific error model, such as single bit-flip, random, or zero value). After each injection, the output results are collected for analysis and error classification.

### C. Fault Classification and Hardening

The errors in the HSI application are classified according to their impact on the application outputs. We classify benign corruption effects as *Masked* when the HSI application remains unaffected (unaltered outputs) even in the presence of faults. An error is labeled as *Detected Unrecoverable Error* (DUE) when the system collapses during the execution of the application (e.g., hanging or crashing). In contrast, when corruption affects and changes the output values, we classify the error as *Silent Data Corruption* (SDC). This group is further divided into two categories: *i) SDC-safe* and *ii) SDC-critical*. The *SDC-safe* cases represent corruptions (mismatches) in the output values of the application (output logits), which still lead to the correct classification results. In contrast, *SDC-critical* are corruptions impacting both the logits and the classification.

We evaluated for each code block (kernel) the amount of times they led to *Masked*, *DUE*, *SDC-safe* and *SDC-critical* outcomes when targeted for fault injection. Then, we identified the most sensitive kernels (i.e., that led to most *SDC-critical* outcomes) and applied a software-based hardening mechanism to mitigate soft error impacts in the application [5].

## IV. Study Case

Our study case is [9], where the authors presented an HSI classifier consisting of PCA and CNN stages for preprocessing and classification, respectively, as shown in Figure 1.

### A. Pre-processing stage

This stage is implemented using Gram-Schmidt PCA (GS-PCA) [11], which is an iterative algorithm for performing PCA, here implemented in cuBLAS to speed up its execution on GPUs. During the operation, the PCA algorithm returns a lower dimensional HSI representation, which serves as input for the inference stage. The algorithm allows configuring the number of principal components (PCs), e.g., *PCA7* with 7 PCs or *PCA50* with 50 PCs, according to the priorities of the target edge system (e.g., energy efficiency or high accuracy).

### B. Classification stage

This stage uses a compacted CNN model including two *Pytorch*-based configurable 3D Convolution layers and one Fully Connected layer that allows its deployment on edge platforms. In particular, the 3D convolution layers apply 3D weights (i.e., stencil filters) on the pre-processed HSI.

## V. Experimental results

For the experiments, we evaluated the reliability of three HSI classifier configurations (*PCA7*, *PCA10*, and *PCA50* [2], [12]) when they are deployed on two systems (`RTX_3060`$_{TI}$ and `GTX_1050`). Then, the kernels, identified as the most vulnerable, serve as the primary candidates for hardening.

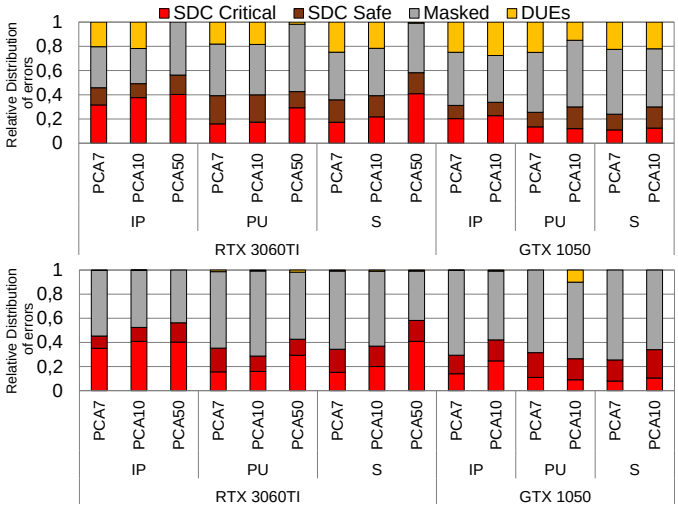


Fig. 2. Relative distribution of SDCs, DUEs, and Masked effects for the three HSI classifiers (PCA7, PCA10, and PCA50) on the evaluated benchmarks, when soft errors affect the Register files (*Top*) and the functional units (*Bottom*) in the GPUs of two systems: RTX_3060$_{TI}$, and GTX_1050.

We employ three representative hyperspectral benchmarks (IndianPines or 'IP,' PaviaUniversity or 'PU,' and Salinas or 'S' with 224, 103 and 204 hyperspectral bands, respectively) for the evaluations and injected a total of 23,600 soft errors (SEU faults) to characterize the 3 HSI classifiers with a statistical confidence interval of 95% and 5% of error margin. In detail, the experiments target the input and output registers of the functional units (**FPUs**) and the Register File (**RF**) of each system. The fault injection campaigns required around 73,5 and 29.5 hours for the evaluation of the RTX_3060$_{TI}$, and GTX_1050 systems, respectively.

### A. Impact analysis

First, we determined the relative distribution of SDCs (safe and critical), DUEs, and masked faults for each HSI classifier on the evaluated benchmark, as shown in Figure 2. Despite the general belief that larger GPUs (in terms of the number of Streaming Multiprocessors, SMs) are more resilient to faults, we noticed a contradicting trend. Figure 2 shows that GTX_1050, while having just 5 SMs, saw a significantly lower amount of faults leading to critical SDCs compared to RTX_3060$_{TI}$ with 84 SMs. This counterintuitive trend required further analyses reported in subsection V-B. However, we anticipate that the system's compiler and architecture impact the overall susceptibility of an HSI classifier.

Regarding the PCA configuration, the results indicate a consistent trend for all benchmarks, showing that HSI classifiers that use many PCs (such as PCA50) are more likely to produce critical SDCs than those with fewer PCs (e.g., PCA7). For example, on RTX_3060$_{TI}$ on S benchmark, going from PCA7 to PCA50 sees a SDC-critical count increase from 152 to 409. In fact, the PCA algorithm uses iterative procedures, which proportionally increase its execution time according to the number of analyzed PCs. Further analyses of all the dynamic execution profiles (*from NVBitFI*) indicate that HSI classifiers with fewer PCs broadly process a lower proportion of PCA-related kernels (PCA7 from 6,2% to 46,5%) than classifiers with more PCs (PCA10 and PCA50 in the ranges [8,6% to 69.9%] and [68.6% to 98.7%], respectively).

Then, we analyzed the critical SDC effects on the evaluated benchmarks. The results for both GPU structures (**RF** and **FPU**) showed similar corruption impacts for PU and S benchmarks, and a slightly higher amount of corruption for the IP benchmark. In fact, IP has a higher number of hyperspectral bands, which causes the classifier to execute more PCA-related kernels compared to the other benchmarks. These results indicate a relation between PCA's execution time and overall HSI classifier susceptibility.

### B. Identification of the most Susceptible Structures

We analyzed the HSI classifier composition (e.g., the number of kernels) and then associated the observed corruptions with their sourcing kernels to identify the most vulnerable elements in the classifiers. A static evaluation indicates that HSI classifiers comprise several kernels (i.e., from 23 to 28 static kernels). However, the dynamic number of executed kernels varies (from 9,277 up to 31,219) according to the GPU architecture, the compiler, and the targeted benchmark.

Figure 3 shows the most susceptible kernels (i.e., those that are most prone to produce critical SDCs from the PCA and CNN stages) for all evaluated benchmarks on both GPUs for the HSI PCA10 configuration. In general, the most susceptible kernels are part of the PCA stage, independently of the system and the evaluated structure (**FPU** or **RF**). However, the kernel susceptibility differs among the systems according to the GPU architecture and the compiler used. The classifier will consist of a different set of kernels when compiled for a different GPU architecture. In fact, our analysis showed that the classifiers for RTX_3060$_{TI}$ included one highly susceptible kernel (*enable_if*), which is not present in the classifiers for GTX_1050. This highly susceptible kernel is the main responsible for the observed system vulnerability in Figure 2, allowing us to state that the compiler and GPU architecture play a crucial role in the overall fault vulnerability.

Then, we evaluated the accuracy drop due to critical SDCs in the outputs, observing that less than 1% of the critical SDCs corrupt all pixels and impact the complete output (100% accuracy dropping). On the other hand, most SDCs cause moderate effects (from 9.0% to 62.6% of accuracy dropping), as shown in Figure 4 for the IP benchmark. As observed, several areas in the image are misclassified (different class/color) by error propagation across the HSI classifier.

The previous results pinpoint the two most important factors affecting the HSI classifier's susceptibility:

- **The PCA size**: the larger the number of PCs employed, the higher the susceptibility to corruptions. More PCs used implies more sensitive kernels executed.
- **The GPU architecture and the Compiler**: both aspects influence the fault tolerance of the application. The two GPUs have different compilers, resulting in different sets of kernels for the same application with different resilience to faults.

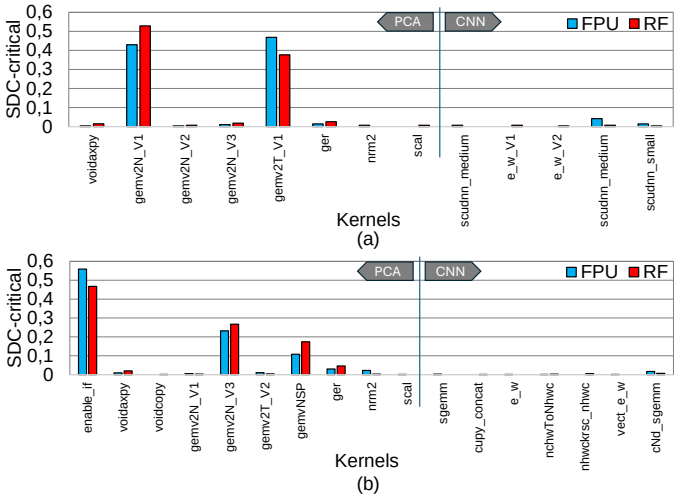


Fig. 3. Normal distribution of critical SDCs among the susceptible kernels in *HSI PCA10* from faults in **FPU** and **RF** for the `GTX_1050` (a) and `RTX_3060`$_{TI}$ (b) systems.

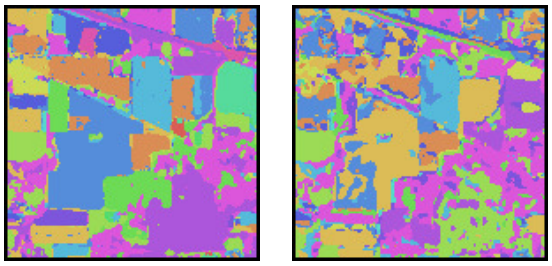

Fig. 4. Fault-free (*left*) and accuracy corrupted (62.6% drop) output (*right*) by a critical SDC in the HSI classifier for the `Indian Pines` (IP) benchmark.

### C. SW Hardening for HSI classifiers

This sub-section describes a software-based hardening mechanism using the *Duplication with Comparison* (or DWC) strategy, targeting only the most vulnerable code blocks (i.e., the ones most likely to cause critical SDCs). The mechanism duplicates the execution of sensitive blocks. Then, comparisons between block outputs identify corruptions. When a corruption is identified, the vulnerable code block (*kernel)* is executed again using the original inputs to remove soft error effects, this way continuing the application execution correctly.

Several fault injection campaigns targeting both GPU structures (**RF** and **FPU**) have been performed to verify the hardening mechanism's effectiveness on all benchmarks. Table I reports the mechanisms' hardening benefits and overhead costs on the *HSI PCA10* classifier. In detail, the results indicate that the mechanism can effectively mask the most critical SDCs (from about 97.5% to 100.0% on all benchmarks), improving the overall resilience of the whole application with minimal memory costs (<1%) and limited cost of extra kernel executions (up to 116.1%) and performance overhead (up to 10.8%). Thus, adopting the DWC strategy showed the feasibility of developing software-based hardening mechanisms as a first step for more efficient approaches.

TABLE I
HARDENING EFFECT AND OVERHEAD COSTS OF THE SOFTWARE-BASED HARDENING MECHANISM ON THE HSI PCA10 CLASSIFIER.

| Benchmark | Targeted GPU Structure | Critical SDCs | | | Instructions Overhead (%) | Memory Footprint (%) | Performance Overhead (%) |
|---|---|---|---|---|---|---|---|
| | | Before Hardening | After Hardening | Drop (%) | | | |
| IP | RF | 376 | 2 | 99.5 | 116.1 | < 1 | 8.4 |
| | FPU | 409 | 2 | 99.5 | | | |
| PU | RF | 174 | 0 | 100.0 | 15.6 | < 1 | 10.8 |
| | FPU | 159 | 4 | 97.5 | | | |
| S | RF | 218 | 3 | 98.6 | 22.4 | < 1 | 9.2 |
| | FPU | 200 | 5 | 97.5 | | | |

## VI. CONCLUSIONS

This work assessed the reliability of GPU-accelerated Hyperspectral Image (HSI) classifiers comprising PCA and 3D neural networks stages for 3 PCA configurations. We target fault effects from 2 GPU structures (register file and functional units) on 2 different GPUs. Our evaluation demonstrates that the PCA stage is responsible for most output corruptions.

To mitigate errors, this work evaluated the effectiveness of a software-based hardening mechanism (based on Duplication with Comparison), targeting the most sensitive code blocks in the HSI classifier. The results show that the mechanism removed most critical errors with moderate performance costs (up to 10.8%) and extra kernel executions (up to 116.1%).

As future work, we plan to explore efficient fault-tolerant solutions to implement PCA algorithms in HSI classifiers.

## REFERENCES


[1] B. Dally, "Hardware for deep learning," in *IEEE Hot Chips 35 Symp. (HCS)*. IEEE Computer Society, 2023, pp. 1–58.

[2] G. De Lucia *et al.*, "Towards explainable ai for hyperspectral image classification in edge computing environments," *Comput. Electr. Eng.*, vol. 103, p. 108381, 2022.

[3] P. Rech, "Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions," *IEEE Trans. Nucl. Sci.*, vol. 71, no. 4, pp. 377–404, 2024.

[4] T. Meuser *et al.*, "Revisiting edge ai: Opportunities and challenges," *IEEE Internet Computing*, vol. 28, no. 4, pp. 49–59, 2024.

[5] M. A. Solouki, *et al.*, "Dependability in embedded systems: a survey of fault tolerance methods and software-based mitigation techniques," *IEEE Access*, 2024.

[6] F. F. d. Santos, P. F. Pimenta *et al.*, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Trans. Reliab.*, vol. 68, no. 2, pp. 663–677, 2019.

[7] D. Gnad *et al.*, "Reliability and security of ai hardware," in *Eur. Test Symp. (ETS)*, 2024, pp. 1–10.

[8] J. E. R. Condia, *et al.*, "A multi-level approach to evaluate the impact of gpu permanent faults on cnn's reliability," in *IEEE Int. Test Conf. (TC)*, 2022, pp. 278–287.

[9] G. De Lucia *et al.*, "Unlocking the potential of edge computing for hyperspectral image classification: An efficient low-energy strategy," *Future Gener. Comp. Sy.*, vol. 147, pp. 207–218, 2023.

[10] T. Tsai *et al.*, "Nvbitfi: dynamic fault injection for gpus," in *Ann. IEEE/IFIP Int. Conf. on Dependable Syst. and Netw. (DSN)*, 2021, pp. 284–291.

[11] M. Andrecut, "Parallel GPU implementation of iterative PCA algorithms," *J. Comput. Biol.*, vol. 16, no. 11, pp. 1593–1599, 2009.

[12] Y. Li *et al.*, "Spectral–spatial classification of hyperspectral imagery with 3d convolutional neural network," *Remote Sens.*, vol. 9, no. 1, 2017.