Characterization of Single-Event Effects in a Microcontroller with an Artificial Neural Network Accelerator

Carolina Imianosky, André M P Mattos, Douglas A Santos, Douglas R Melo, Maria Kastriotou, Carlo Cazzaniga, Luigi Dilillo

electronics

MDPI

Article

Characterization of Single-Event Effects in a Microcontroller with an Artificial Neural Network Accelerator

Carolina Imianosky [1,*], André M. P. Mattos [1], Douglas A. Santos [1], Douglas R. Melo [2], Maria Kastriotou [3], Carlo Cazzaniga [3] and Luigi Dilillo [1,*]

[1] Institut d'Électronique et des Systèmes (IES), University of Montpellier, CNRS, 34095 Montpellier, France; andre.martins-pio-de-mattos@umontpellier.fr (A.M.P.M.); douglas.almeida-dos-santos@umontpellier.fr (D.A.S.)
[2] Laboratory of Embedded and Distributed Systems (LEDS), University of Vale do Itajaí, Itajaí 88302-901, SC, Brazil; drm@univali.br
[3] ISIS Facility, STFC, Rutherford Appleton Laboratory, Oxfordshire OX11 0QX, UK; maria.kastriotou@stfc.ac.uk (M.K.); carlo.cazzaniga@stfc.ac.uk (C.C.)
* Correspondence: carolina.imianosky@umontpellier.fr (C.I.); luigi.dilillo@umontpellier.fr (L.D.)

Abstract: Artificial neural networks (ANNs) have become essential components in various safety-critical applications, including autonomous vehicles, medical devices, and avionics, where system failures can lead to severe risks. Edge AI devices, which process data locally without relying on the cloud, are increasingly used to meet the performance and real-time demands of these applications. However, their reliability in radiation-prone environments is a significant concern. In this context, this paper evaluates the MAX78000, an ultra-low-power Edge AI microcontroller with a hardware-based convolutional neural network (CNN) accelerator, focusing on its behavior in radiation environments. To assess the reliability of the MAX78000, we performed a test campaign at the ChipIR neutron irradiation facility using two different ANNs. We implemented techniques to improve system observability during ANN inference and analyzed the radiation-induced errors observed. The results present a comparative analysis between the two ANN architectures, which shows that the complexity of the ANN directly impacts its reliability.

Keywords: artificial neural network; edge AI; artificial intelligence accelerator; radiation effects; reliability; single-event upset; single-event effects; atmospheric environments; neutrons

Citation: Imianosky, C.; Mattos, A.M.P.; Santos, D.A.; Melo, D.R.; Kastriotou, M.; Cazzaniga, C.; Dilillo, L. Characterization of Single-Event Effects in a Microcontroller with an Artificial Neural Network Accelerator. *Electronics* **2024**, *13*, 4461. https://doi.org/10.3390/electronics13224461

Academic Editor: Ping-Feng Pai

Received: 30 September 2024
Revised: 7 November 2024
Accepted: 12 November 2024
Published: 14 November 2024

1. Introduction

Artificial intelligence (AI) is widely applied across various domains, including critical areas such as autonomous vehicles, medical devices, and avionics systems [1–3]. In particular, edge AI devices integrate edge computing with AI, enabling data processing to occur locally rather than relying on cloud solutions. This approach has gained popularity, mainly due to the increasing demand for hardware AI accelerators in edge devices, driven by the need for higher performance and energy efficiency in real-time AI task processing [4,5].

Recent advancements in microcontrollers and accelerators have enabled running AI algorithms efficiently on small, resource-limited devices, supporting real-time applications even in environments with limited or no connectivity. In recent years, there has been an increasing utilization of microcontrollers for AI in edge computing applications for avionics, such as satellite imaging [6], drones [7], and aircrafts [8].

However, for avionics operating at high altitudes or in orbit, exposure to ionizing radiation raises significant concerns regarding system reliability and functionality [9,10]. Galactic cosmic rays (GCRs) and solar particles interact with Earth's atmosphere, producing secondary particles like neutrons, protons, and electrons through nuclear reactions [10,11]. At ground level, neutrons are the primary source of soft errors in electronics, with flux levels around 13 n/cm²/h for energies above 10 MeV and 6.5 n/cm²/h for thermal energies

in New York [11–13]. This flux increases by approximately 300 times at typical flight altitudes (12 km) and up to 500 times at 18 km [10]. The interaction of ionizing particles with electronic devices induces a variety of effects. single-event effects (SEEs), which occur when high-energy particles interact with sensitive regions of electronic components, are particularly impactful, causing faults that may be transient, intermittent, or permanent [14]. These faults manifest as bit flips, data corruption, or even permanent hardware damage [15].

Microcontrollers exposed to radiation environments encounter critical failure mechanisms primarily due to single-event effects (SEEs). One common SEE, the single-event upset (SEU), arises when charged particles alter logic states within memory cells or registers, resulting in bit flips that disrupt data integrity and program execution. According to [14], SEUs are a primary cause of transient faults in central processing unit (CPU) operations and can affect various CPU components, such as the bus control logic, where mode shifts can create faulty memory access patterns or unintended program jumps. SEUs impacting the program counter or segment registers can alter execution flow, redirecting it to incorrect memory addresses. Instruction registers, general-purpose registers, and state registers are also vulnerable, with faults in these areas either producing execution errors or self-correcting depending on timing and program flow [14].

More severe SEEs, including single-event latch-ups (SELs) and single-event functional interrupts (SEFIs), present additional risks to microcontroller stability. SELs occur when particle strikes activate parasitic structures within CMOS circuits, creating high, sustained currents that can cause irreversible damage if not mitigated immediately [16]. SEFIs may disrupt core functionalities by disabling control units or critical registers and frequently require a full power cycle to restore normal operation [17].

In AI accelerators, these radiation-induced faults can compromise the accuracy of computations, leading to incorrect outputs or unpredictable behavior [18,19]. Thus, as the use of AI accelerators grows in critical applications, it is essential to evaluate their behavior, identify potential weaknesses, and develop strategies to enhance their reliability [20].

In this context, we present a reliability analysis of the MAX78000 microcontroller, an ultra-low-power, low-cost application-specific integrated circuit (ASIC) designed for edge AI applications with a hardware-based convolutional neural network (CNN) accelerator. The primary focus of this research is to investigate the effects of radiation exposure on the performance and fault behavior of the MAX78000. We employ a layer-by-layer analysis technique to increase system observability during neural network inference, facilitating more effective monitoring and assessment of radiation-induced faults. To our knowledge, no other studies have specifically examined the effects of radiation on the MAX78000, highlighting both the novelty and importance of this work.

The main contributions of this paper include the following: (i) an evaluation of the MAX78000 microcontroller to explore its potential in real-world edge AI applications within radiation-exposed environments; (ii) the application of a method that examines fault behavior in a deeply embedded device, despite challenges posed by a black-box design; and (iii) a comprehensive report on sensitivity to neutron-induced faults, including an analysis of error propagation and insights into performance under adverse conditions. Together, these contributions provide valuable information on the reliability of low-cost microcontrollers, addressing notable gaps in the existing literature.

The remainder of this paper is structured as follows: Section 2 describes the background of this work; Section 3 presents the related works; Section 4 presents the devices used for the experiment, the test facility, and the applied test methodology; Section 5 presents the results obtained during the experiment; and Section 6 concludes this work.

2. Background

This section introduces the base concepts of artificial neural networks (ANNs). Moreover, we present discussions on the radiation reliability of ANNs.

2.1. Artificial Neural Networks

Artificial neural networks (ANNs) were first presented in 1986 as a back-propagation model based on the concept of the neural synapses [21]. Its processing is based on the convolution of an input vector by a set of iterative weights, which are separated as ANN layers. One of the most critical aspects of back-propagation is the ability to perform learning by feed-backing the error to calibrate the weights, improving the result's precision. The layers, namely the input, hidden, and output, are separated by their function. The input layer will have the same data format to be considered for the neural network (NN) and will be responsive for the first formatting of the data. The hidden layers are considered a black box for processing. The designer will model the output layer, depending on the purpose of the ANN. All of the layers have weights adjusted by the learning procedure, which aims to produce the most precise result possible.

The versatility of ANNs has enabled its use in numerous fields, including computer vision, natural language processing, and robotics. Notably, convolutional neural networks (CNNs) [22] have become an important ANN variant for the field of image recognition, which has enabled precise image classification due to its ability to deal accurately with 2D shapes. Although initially, ANNs had inferior precision, advances in deep learning have significantly improved their performance. More specifically, deep neural networks (DNNs) have achieved state-of-the-art results in areas such as image classification, natural language processing, and reinforcement learning [23].

2.2. Neural Networks in Radiation Environments

Early studies on the reliability of ANNs [24,25] have analyzed the robustness of the ANN architecture against errors, presenting masking traits that generate inherent resilience. Still, radiation-induced errors can propagate within ANN layers and generate critical failures, which are further accentuated with the proposal of complex ANN architectures (e.g., CNNs and DNNs) that depend on several hardware components [18].

In embedded applications, ANNs are predominantly implemented using hardware accelerators based on ASICs, field-programmable gate arrays (FPGAs), and graphics processing units (GPUs). Each hardware platform provides unique performance, power, and reliability traits [18,26]. For instance, while GPUs provide significant parallelism and performance to ANNs, their complex hardware architecture and deeply integrated technology allow for the propagation of single faults to several outputs, significantly impacting the results of ANNs [20]. In [27], the authors provide a comprehensive literature review on hardware reliability assessment methods for DNNs. Using fault injection, fault emulation, and irradiation in particle accelerators, the analyzed studies demonstrate the susceptibility and vulnerability of transient and permanent errors in several elements of DNNs.

Similarly to other processing workloads, ANNs depend on several memory and computing elements that are susceptible to radiation. For example, storage and buffering elements used to save weights, activation inputs, biases, layer outputs, and accelerator configuration are sensitive to SEUs, which corrupt the stored value, resulting in the accumulation and propagation of errors during execution [19]. For this reason, as important as the ANN architecture's impact on the reliability of the target application is, the hardware platform's sensitivity to radiation significantly determines the error rate and propagation within the ANN. System failures, such as crashes, hangs, and overcurrent events, generally correlate to device technology, critically disrupting ANN execution, even with commonly used error mitigation strategies [20].

3. Related Work

In recent years, there has been a significant focus on the reliability of AI applications [27]. Many studies have examined the impact of radiation-induced faults in ANNs executed in FPGAs and GPUs. In [28], the authors investigated the susceptibility of a design implemented in a Xilinx 28 nm SRAM-based FPGA to SEUs using various fault injection techniques to analyze the resilience of a NN trained for classification tasks. In [29],

the impact of radiation-induced errors on CNNs implemented on FPGAs was assessed, emphasizing how binary quantization of convolutional layers can reduce sensitivity to radiation while analyzing the relationship between model accuracy and radiation effects.

Other research has focused on the reliability of ANN algorithms running on GPUs. A study [20] evaluated the performance of object detection algorithms on three NVIDIA GPU architectures using neutron beam experiments to assess the propagation of faults during execution. Additionally, another study [30] combined neutron beam experiments with software-level fault injection to characterize the fault model for deep neural networks executed on GPUs, reporting error rates and the effectiveness of error correction codes.

While FPGAs and GPUs have proven to be effective platforms for executing deep learning algorithms, edge AI applications based on low-power ASIC accelerators offer a practical alternative. These accelerators are designed to be more energy-efficient and cost-effective. A discussion [31] highlighted how edge AI accelerators can deliver improved performance at lower costs and power levels. Furthermore, another study [32] emphasized that edge AI accelerators are increasingly being utilized in various applications, requiring performance and resilience to operate effectively in challenging environments. Thus, microcontrollers like MAX78000 are increasingly relevant, incorporating hardware-based CNN accelerators and offering low-cost, energy-efficient solutions for implementing AI algorithms.

Recent studies on the MAX78000 microcontroller have demonstrated its potential for edge AI applications. In particular, the authors in [33] evaluated MAX78000's performance and efficiency in executing CNN inference using datasets such as MNIST and CIFAR-10. The research highlights the trade-offs between performance and accuracy when utilizing the microcontroller for various AI tasks. Also, the study in [34] analyzed the performance of the MAX78000 microcontroller, highlighting that simpler DNN models result in lower latencies and energy consumption, while more complex models increase both. The study also notes that optimizing layer sizes, bit widths, and memory usage can further improve efficiency, supporting the suitability of MAX78000 for low-power edge AI applications. In [35], a low-power smart camera that integrates MAX78000 was proposed, emphasizing its ability to operate effectively under intermittent power conditions. This research features a dual-stage energy harvesting approach, enabling the system to sustain energy-intensive data transmission to sub-GHz low Earth orbit (LEO) radios.

In this context, our work focuses on a reliability analysis of the MAX78000 microcontroller. By examining the effects of radiation on this platform, we aim to provide insights into its reliability and operational characteristics.

4. Experimental Evaluation

To evaluate the device under test (DUT) reliability, we performed an experiment at the ChipIR beamline [36], part of the ISIS Neutron and Muon Source, at the Rutherford Appleton Laboratory, United Kingdom. ChipIR generates a neutron beam with a spectrum representative of atmospheric environments, which causes SEE on electronic devices. The facility can generate neutron irradiation with fluxes that are several orders of magnitude higher than those found at ground level on Earth, which enables an accelerated characterization of devices and systems. The beam flux is 5×10^6 cm^{-2}/s for energies higher than 10 MeV, and during the experiment, the total accumulated fluence was 1.11×10^{12} n/cm^2.

4.1. Device Under Test

MAX78000 is an ultra-low-power edge AI microcontroller. It has a heterogeneous architecture that combines two main processing hardware modules: a microprocessor cluster and a CNN accelerator. The microprocessor cluster combines an ARM Cortex-M4 with a floating-point unit processor running at up to 100 MHz and a 32-bit RISC-V coprocessor running at up to 60 MHz, both interfaced to the bus, where several peripherals and the CNN accelerator are connected.

Figure 1 shows the DUT architecture, highlighting the CNN accelerator's internal structures. The hardware-based CNN accelerator in MAX78000 features 64 parallel convo-

lutional processors and supports a wide variety of CNN and DNN models. It has dedicated memory for corresponding weights (442 KB), biases (2 KB), and data (512 KB), and it operates at a frequency of up to 50 MHz. Each convolutional processor consists of a pooling engine, input cache, weight memory, and convolution engine and is responsible for running a convolutional operation of a single input channel.

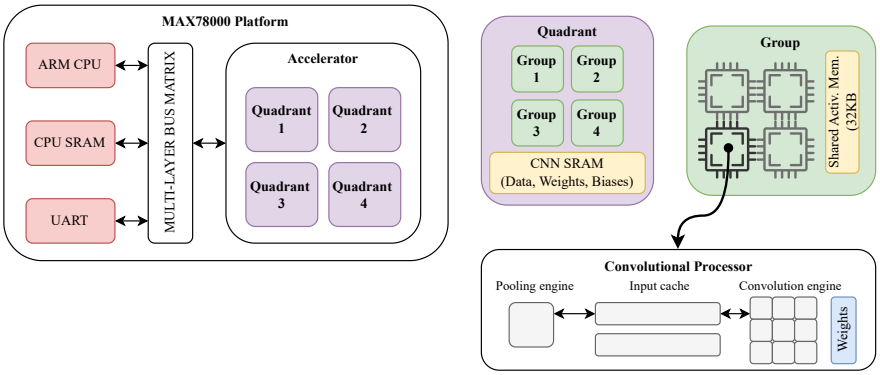

Figure 1. MAX78000 platform architecture.

The convolutional processors are divided into four groups, and processors within a group share a common activation memory for data input and activation data. These groups are then clustered into four sets, forming quadrants, each containing programmable registers to configure the execution of individual layers. This hierarchical structure, dividing the accelerator into quadrants, groups, and processors, allows for precise control over which accelerator sections are active during operation.

MAX78000's software development kit (SDK) [37] handles the complete training, synthesizing, and deploying of ANN models. The training is performed with the help of the PyTorch library for machine learning, data loader, and corresponding datasets. The trained ANN model is converted into C code during the synthesis and deployment stages. Once deployed on the device, the accelerator is initialized by configuring the SRAM control bits and specifying the number of CNN layers. Subsequently, weights and biases are moved from general-purpose SRAM into their dedicated memory locations. The configuration then continues by defining the rows and columns for each quadrant, which consists of 16 processors, and mapping the necessary memory pointers. Input data are then loaded into the memory of the accelerator, triggering the inference process. MAX78000 performs NN inference as a single, indivisible task, which inherently limits the ability to monitor intermediate computational results and restricts system observability.

4.2. Layer-by-Layer Implementation

In this study, we address a challenge posed by the MAX78000 CNN accelerator, which performs NN inference as a single, indivisible task. This design feature limits our capacity to observe intermediate computational results, making it challenging to verify the accuracy and reliability of the inference process. To overcome this limitation, we used the Layer-by-Layer Transient Toolchain (LbLTT) [38], an external tool designed to enable layer-by-layer processing on MAX78000. LbLTT converts NN inference into several single-layer NNs, allowing each layer to be individually processed. This method allows us to observe the output of intermediate layers and verify that the parameters are correctly stored, improving observability without altering the NN structure.

LbLTT is an open-source toolchain developed specifically for NNs compatible with the MAX78000, optimized to allow for intermittent execution by shrinking the atomic execution size to a single layer. This tool achieves this segmentation without modifying network weights or configurations, ensuring that inference accuracy remains consistent with the original network design. It is important to note that LbLTT is applied solely to improve observability, and it introduces some latency to the inference speed [38].

Initially, the toolchain uses the *ai8xize.py* Python script to translate the PyTorch model into C code. This process generates two essential files: *cnn.c*, which contains the functions necessary for loading, starting, and obtaining outputs from the NN accelerator, and *weights.c*, which stores the values of the NN weights. Following this, LbLTT segments the NN into individual layers. The LbLTT then segments the neural network into separate layers, creating a standalone network for each layer, with an input, a layer, and an output. This segmentation enables the creation of configuration files to set up the CNN accelerator at the beginning of the computation of each layer. In the final stage, the LbLTT toolchain merges the segmented layers into a unified execution framework for the MAX78000 CNN accelerator. This framework includes functionality that allows the intermediate output to be extracted and verified as needed. The resulting C code is then ready to be deployed on the MAX78000 CNN accelerator, supporting intermittent execution.

The execution flow, as illustrated in Figure 2, contrasts the standard implementation of the entire NN, presented on the left, with the layer-by-layer approach enabled by LbLTT presented on the right. The green dashed rectangle indicates the loop performed for each layer of the original NN model, and the purple one indicates the verification steps that we implemented to increase the observability of the accelerator. It is important to note that while network weights are loaded at the start of execution, biases are loaded individually for each layer. This is done by modifying the *ai8xize.py* script to include memory offsets for the biases; a necessary step because the Maxim toolchain treats each layer as a separate NN. The default script maps all layers from a fixed address (e.g., 0 × 0000) and does not allow for specifying different memory locations for weights, requiring their preloading. After the CNN initialization, the input data for the first layer are loaded into the memory of the accelerator. Subsequently, the biases for this layer are loaded, and the accelerator is configured and started. This process is repeated for each layer of the NN model, allowing for fine-grained control and observation of intermediate results.

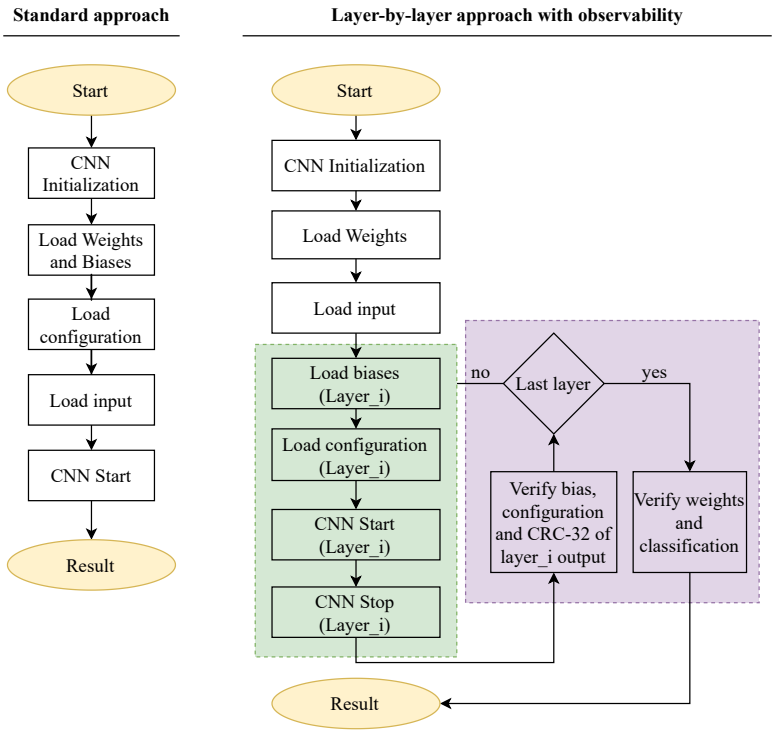

Figure 2. Comparison of CNN execution runtime flow. The left diagram shows the standard approach. The right one shows the combination of the LbLTT [38] layer-by-layer approach with the implemented techniques for observability.

We performed golden runs, i.e., error-free runs, on all selected test images from the dataset, obtaining the expected output values for each layer and each image. We then

calculated the Cyclic Redundancy Check-32 (CRC-32) for these outputs. CRC-32 is a widely used error-detection method that produces a unique checksum for data [39,40], enabling the detection of discrepancies that may indicate errors in any output of a layer. This approach allowed us to verify the outputs of each layer without storing all of the intermediate results. Additionally, at the end of each layer, we verified the values of the bias and configuration registers. The weights were checked only at the final layer, as verifying them after each layer would significantly extend the inference time.

4.3. Neural Networks and Datasets

We evaluated and characterized MAX78000 using two different CNN models provided by Maxim Integrated [37]: the AI85Net5 NN model trained on the Modified National Institute of Standards and Technology (MNIST) [41] dataset and a model found through NAS (Neural Architecture Search) trained on the Canadian Institute For Advanced Research (CIFAR-10) [42] dataset. The MNIST is a collection of 70,000 28 × 28 grayscale images of handwritten digits from 0 to 9. The CIFAR-10 consists of 60,000 32 × 32 color images in 10 classes, with 6000 images per class.

Datasets are also usually split into a training set and a testing set. MNIST is divided into 60,000 training images and 10,000 testing images, and CIFAR-10 is split into 50,000 training images and 10,000 test images. For training, we used the full training sets of both datasets. For testing the applications, we selected a subset of 20 images from each dataset testing set due to the processor's internal memory constraints. The goal was to investigate the accelerator's performance by comparing the results of each image with the ones from the golden execution rather than evaluating the overall accuracy of the AI on a larger dataset. The subset of images was chosen randomly while guaranteeing an equal number of images for each category.

Table 1 presents the characteristics of the CNNs used in this work, including inference times for both the standard and layer-by-layer approaches. The layer-by-layer method increases inference time by 13.7% for CIFAR-10 and 12.8% for MNIST. While this approach adds some latency overhead, it enables detailed fault monitoring across layers, increasing the observability of the system.

Table 1. Characteristics of the CNNs.

Dataset	Num. of Layers	Weights [Bytes]	Bias [Bytes]	Num. of Operations	Inference Time (µs)	
					Standard	Layer-by-Layer
CIFAR-10	11	301,760	842	36,484,536	4872	5541
MNIST	5	71,148	10	10,883,968	1486	1676

4.4. Test Setup

We used the MAX78000FTHR evaluation board for the experiment, which provides a platform for programming and debugging the MAX78000 microcontroller and several peripherals. The test setup, shown in Figure 3, consists of four MAX78000FTHR boards, where #B1 and #B2 were performing inferences with the CIFAR-10 dataset, and #B3 and #B4 performed inferences with the MNIST one.

For logging the experimental data, we used serial connections to transmit the logs generated to a host computer outside the irradiation room. The setup is fixed in a frame containing the mentioned test boards and other boards utilized for other experiments to optimize the beam time utilization. The supply current for each board was individually monitored and limited in case of a single-event latch-up. The UART output was also being monitored to trigger a power cycle in case of a hang, i.e., if no output was detected for more than one minute.

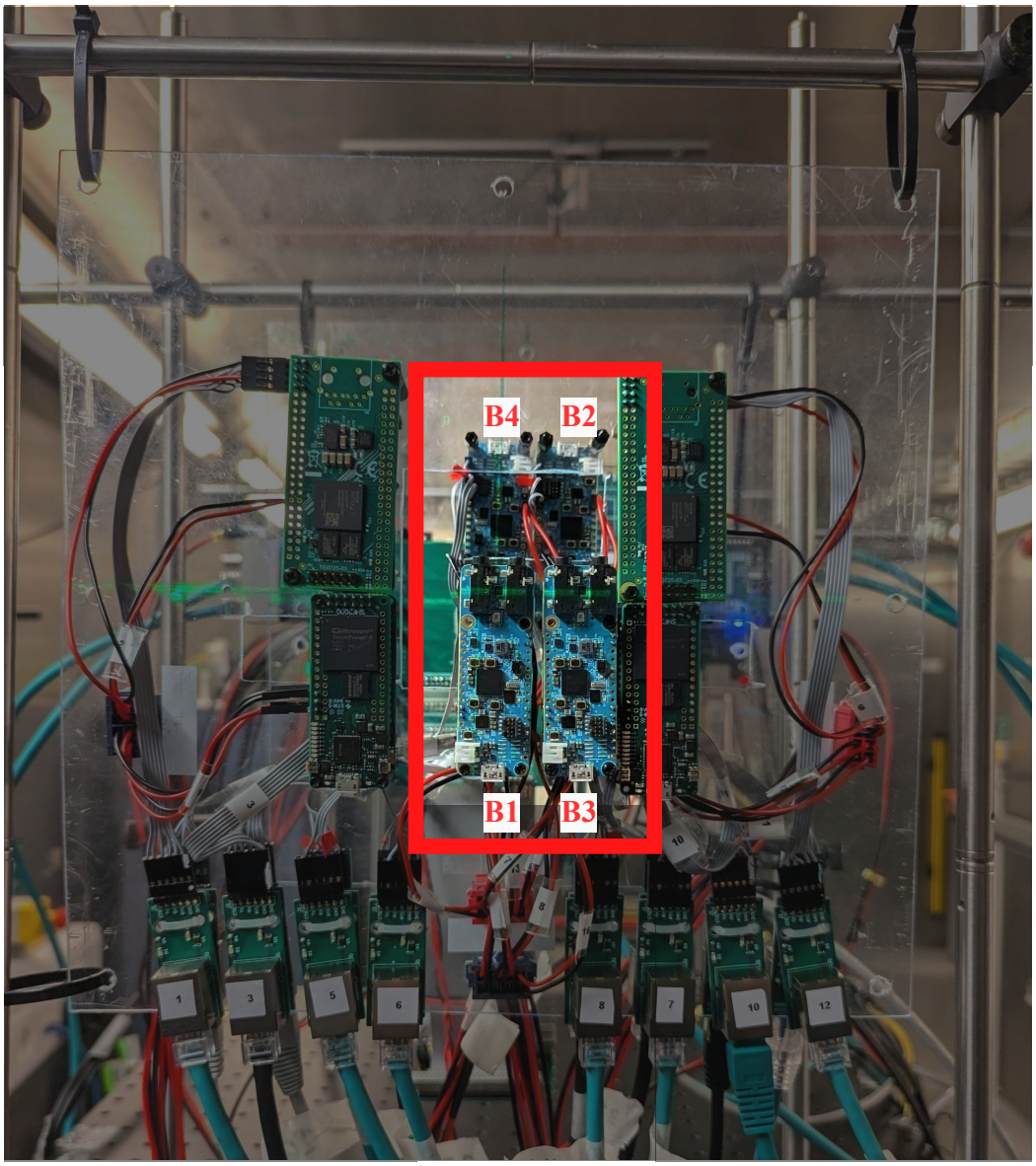

Figure 3. Experimental setups prepared for the irradiation campaign composed of four MAX78000FTHR boards.

4.5. Metrics

We estimated the system failure and weight error cross-section to assess the reliability. The cross-section calculates the device's failure rate based on the fluence, given in cm^2/device. Furthermore, we calculated the failure in time (FIT) metric based on New York City's sea-level neutron flux (Φ_{NYC}). The FIT metric shows how many failures occur in a billion hours, depending on the sensitivity of the device and the particle flux to which it will be exposed. The FIT is given by Equation (1), where σ represents the cross-section and 10^9 is a billion hours.

$$FIT_{NYC} = \sigma \times \Phi_{NYC} \times 10^9 \quad (1)$$

5. Results

During the experiment, various types of radiation-induced errors were observed. These errors had different levels of impact on the system, ranging from no noticeable effects to complete system failure. We classified the executions in which errors were reported into five categories:

- No impact (NI): Executions where the corrupted data were either unused or did not affect the circuit's functionality, resulting in no observable change in the program output.
- Major output disturbance (MaOD): The execution finished, but the classification result was entirely incorrect.
- Medium output disturbance (MeOD): The execution finished, but the confidence level or percentage of the classification was incorrect, even though the predicted class itself was still correct.
- Minor output disturbance (MiOD): The execution finished, but the error caused a slight disturbance in the output of the last layer. Despite this, the final classification remained unaffected.
- System failure (SF): When the error led to the program to stop prematurely or caused the entire system to crash.

Table 2 presents the number of images that were processed, the number of images that shoedw any error, and the classification of errors per board, detailing the number of errors reported for each category. It is important to note that the CNN used for the MNIST dataset, processed in #B3 and #B4, was less complex, with fewer layers and parameters, and it had a lower latency than the one used for the CIFAR-10 in #B1 and #B2.

The error rates observed for the MNIST dataset were notably lower compared to CIFAR-10, likely due to a combination of factors. The simpler CNN architecture used for MNIST had lower resource demands, which reduced the critical area exposed to faults and limited fault propagation within the system. Also, the inherent characteristics of the MNIST images, such as their lower complexity and grayscale format, reduced the computational load, making the system less vulnerable to errors. These findings aligns with existing literature suggesting that simpler models can offer improved reliability [43–45].

Table 2. Classification of errors per board.

Board	Number of Images	Images with Errors	MaOD	MeOD	MiOD	NI	SF
#B1	134,783	130,140	775	44,161	82,698	2506	167
#B2	131,416	127,031	686	44,134	79,839	2372	220
#B3	169,927	124,357	9	27	14,764	109,557	171
#B4	203,574	136,881	86	9	18,123	118,663	218

5.1. Major, Medium, and Minor Output Disturbance

Tables 3–5 show the errors reported during MiOD, MeOD, and MaOD executions, respectively. Of all the images processed that reported an error (third column of Table 2), 97.08% of them reported only weight errors. Both configuration and bias values were updated at each layer; therefore, they were updated frequently, leading to a reduced probability of a bit upset occurrence.

Table 3. Errors reported in MiOD executions.

Board	Weight	Weight and Config.	Weight and Bias	All	Unknown
#B1	80,391	10	0	1	9
#B2	78,088	5	1	0	6
#B3	14,752	0	0	0	12
#B4	18,088	3	0	0	32

Table 4. Errors reported in MeOD executions.

Board	Weight	Config.	Weight and Config.	Weight and Bias	All	Unknown
#B1	42,365	1	6	1	1	4
#B2	42,952	0	3	0	1	4
#B3	27	0	0	0	0	0
#B4	9	0	0	0	0	0

Table 5. Errors reported in MaOD executions.

Board	Weight	Config.	Weight and Config.
#B1	763	7	3
#B2	573	0	1
#B3	8	0	1
#B4	85	0	1

Of the few executions that reported errors in the configuration or bias values, most of these errors were not attributable to bit upset. For the configuration cases, most of the observed values were significantly different rather than just altered by a single bit. In all the cases where a bias error was reported, it indicated that all the values were completely wrong. Therefore, most configuration and bias errors were caused by a failure in the writing process. For MeOD and MiOD executions, the unknown cases refer to instances where no errors besides the values of the output of the layers were reported, yet the final result was still impacted. These were likely caused by an error in the control of the accelerator.

In our analysis of MaOD executions, we observed that, although these executions showed weight errors, the total number was low compared to the overall weights, reaching a maximum of 490 errors for CIFAR-10 and 110 for MNIST. Given this limited number, we conducted fault injection simulations on the weights to determine if these errors alone could explain the observed misclassifications. Faults were injected exclusively into the weights, as bias errors were minimal, and most configuration errors appeared unrelated to actual upsets.

For the fault injection, we implemented a function that randomly selects a bit within a random address in the weights storage and flips it. To closely match the original experimental conditions, we ran the same flow used in the experiment, processing the same set of images from both the MNIST and CIFAR-10 datasets. The fault injection function was called once before processing each image, ensuring one injected fault per inference. Whenever a misclassification occurred, we restarted the entire execution, resetting the injected faults. We repeated this procedure 20 times to observe the average number of weight errors required to trigger a MaOD event.

The results indicated that a large number of weight errors was needed to induce misclassification, with an average of 1316 errors for CIFAR-10 and 2628 for MNIST. The lowest number of weight errors leading to misclassification was 570 for CIFAR-10 and 824 for MNIST, though these were exceptions, with most cases closer to the average thresholds.

Given that the weight error counts observed in the experiment were significantly below the thresholds needed to induce misclassifications in our fault injection tests, these misclassifications were likely influenced by factors other than only the weight errors. During our analysis, we noticed recurring patterns in the MaOD executions, where specific types of misclassifications were repeated multiple times. For instance, 10, 12, 5, and 5 distinct patterns accounted for all misclassifications on devices #B1, #B2, #B3, and #B4, respectively. It is worth noting that these patterned occurrences ceased only after a processor reset was triggered by one of the SF occurrences. The identified patterns and their potential causes are detailed below:

- Stuck output values: There were cases of stuck output values where the values from the last layer of the NN unloaded from the accelerator remained the same over a sequence of images, despite the images being different. The intermediate layer values varied in some instances, indicating that the error occurred only at the final output. In other cases, the intermediate and final output values were identical, likely indicating that an error occurred during image loading. In a few cases, no errors were detected in the intermediate layers, probably caused by an event on the controller of the accelerator.
- Sequential misclassifications: One type of pattern involves sequential misclassifications of the same image with similar output values. The same image was misclassified multiple times in sequence in a few instances. While the classification was wrong each time, the output values varied slightly but remained similar. Since, in these cases, the error occurred for only one image (and not for the others from the same class), it is most likely that the issue lies in the variable that stores the memory pointer to the image, which results in loading the wrong values for the input memory and thus an incorrect classification.
- Multiple misclassifications in sequence: Finally, there was one scenario where several images were misclassified in sequence. Each one had different output values and class predictions. However, there was no report of wrong output between the layers, and only a few weight errors were reported. This was also probably caused by an event on the controller of the accelerator.

5.2. System Failure

During the testing phase of the microcontroller, several types of system failures were identified. Table 6 summarizes the frequency of these failures across different boards during testing. Hard faults occur when the microcontroller encounters a critical error, such as attempting to access invalid memory or executing an illegal instruction. Another failure type involved the watchdog timer (WDT) triggering a reset when the system failed to respond within a set time limit, typically due to software hangs or deadlocks. Additionally, there were instances where the system reset itself due to an unknown cause, categorized as an unrecognized reset, likely caused by an SEU on the control of the system. In all these mentioned cases, the system performed a soft reset.

As mentioned in Section 4.4, the current and the UART output were monitored during the experiment. Some failures were attributed to system hangs, identified when no output was received from the UART for over one minute, indicating that the system became unresponsive. Also, some overcurrent cases were detected. In both cases, the monitoring system triggered a power cycle on the board.

Table 6. Reported cases of system failure.

Board	Hard Fault	WDT Reset	Unrecognized Reset	Overcurrent	Hang
#B1	55	15	44	5	30
#B2	55	28	47	17	36
#B3	41	21	56	3	28
#B4	63	32	49	3	51

5.3. Reliability Analysis

Table 7 summarizes the cross-sections for each type of SF for each board. We summed all the errors from each category across all boards to determine the total number of SFs. The overall cross-section (XS) was calculated to be 1.72×10^{-10} cm^2/device. Based on this, we derived a total SF FIT value of 2.24.

Table 7. Summary of SF cross-sections per board.

Board	Fluence [n/cm^2]	XS [cm^2/Device]				
		Hard Fault	**WDT Reset**	**Unrecognized Reset**	**Overcurrent**	**Hang**
#B1	1.03×10^{12}	5.33×10^{-11}	1.45×10^{-11}	4.27×10^{-11}	4.85×10^{-12}	2.91×10^{-11}
#B2	9.01×10^{11}	6.11×10^{-11}	3.11×10^{-11}	5.22×10^{-11}	1.89×10^{-11}	4.00×10^{-11}
#B3	9.53×10^{11}	4.30×10^{-11}	2.20×10^{-11}	5.88×10^{-11}	3.15×10^{-12}	2.94×10^{-11}
#B4	1.05×10^{12}	5.97×10^{-11}	3.03×10^{-11}	4.65×10^{-11}	2.84×10^{-12}	4.84×10^{-11}

Table 8 shows the accumulated fluence for weight errors, the number of errors reported in the weight memory, the weight errors cross-section, the number of weights, the FIT, and the normalized FIT. The cross-section and FIT values for boards #B1 and #B2 were significantly higher than those for boards #B3 and #B4. This is because the memory used by the weights in the CNNs on boards #B1 and #B2 is much larger than that of boards B3 and B4, resulting in a substantially larger sensitive memory area and, consequently, a higher probability of soft errors. Therefore, the sixth column shows the normalized weight error cross-section by dividing the cross-section by the number of weights. Since the cross-section value is used to calculate the FIT, we also estimated a normalized FIT, represented in the seventh column, using the normalized cross-section value. After the normalization of the cross-section and the FIT, it became evident that the values across all the boards were similar, which means that these events were independent of the ANN architecture and dataset.

Table 8. Weight errors FIT and cross-section.

Board	Fluence [n/cm^2]	Weight Errors	XS [cm^2/Device]	Num. of Weights	XS [cm^2/Weight]	FIT	Normalized FIT
#B1	9.43×10^{11}	14,318	1.52×10^{-8}	301,760	5.03×10^{-14}	197.30	6.54×10^{-4}
#B2	7.93×10^{11}	14,258	1.80×10^{-8}	301,760	5.96×10^{-14}	233.73	7.75×10^{-4}
#B3	8.77×10^{11}	3368	3.84×10^{-9}	71,148	5.40×10^{-14}	49.94	7.02×10^{-4}
#B4	9.76×10^{11}	4232	4.33×10^{-9}	71,148	6.09×10^{-14}	56.35	7.92×10^{-4}

6. Conclusions

This work provides an analysis of the MAX78000, a microcontroller with a hardware CNN accelerator, offering insights into the behavior and reliability of this device. We increased system observability through layer-by-layer monitoring, enabling detailed tracking of faults at each NN layer. Additionally, we examined the types of errors caused by radiation and evaluated the reliability of the DUT.

The varying performance across the different boards tested, particularly between those processing the MNIST dataset and those handling CIFAR-10, underscores the influence of the CNN architecture on error rates. The CNN used in boards #B3 and #B4 exhibited lower error rates, likely due to their simpler architecture requiring less memory usage, memory pointers, and variables. Thus, this reduced resource usage makes them less vulnerable to errors. This finding aligns with existing literature suggesting that simpler models can offer improved reliability [43–45].

Many observed errors originated from the processor, with configuration and bias errors mostly related to memory access issues rather than direct SEUs. Additionally, the number of MaOD executions remained relatively low, which indicates that the inference process with the CNN accelerator is reasonably robust under radiation.

While these findings provide valuable insights into the reliability of MAX78000, the study is limited by the specific CNNs, datasets, and conditions used. Thus, as future work, we aim to expand our investigations by conducting experiments with additional radiation sources to understand their effects on the MAX78000 microcontroller better.

Furthermore, we plan to evaluate larger datasets and different CNNs to simulate real-world applications. Lastly, we will conduct a comparative analysis of layer-by-layer implementation versus the original architecture to assess whether this approach impacts the reliability of CNN inference.

Author Contributions: Conceptualization, C.I., A.M.P.M., D.A.S. and L.D.; methodology, C.I., A.M.P.M., D.A.S. and L.D.; software, C.I., A.M.P.M. and D.A.S.; validation, C.I., A.M.P.M. and D.A.S.; formal analysis, C.I., A.M.P.M. and D.A.S.; investigation, C.I., A.M.P.M. and D.A.S.; resources, M.K., C.C. and L.D.; data curation, C.I., A.M.P.M. and D.A.S.; writing—original draft preparation, C.I., A.M.P.M. and D.A.S.; writing—review and editing, C.I., A.M.P.M., D.A.S., D.R.M., M.K., C.C. and L.D.; visualization, C.I., A.M.P.M. and D.A.S.; supervision, L.D.; project administration, L.D.; funding acquisition, L.D. All authors have read and agreed to the published version of the manuscript.

Funding: The results presented in this paper were obtained in the framework of the EU project RADNEXT, receiving funding from the European Union's Horizon 2020 research and innovation programme (Grant Agreement no. 101008126), the Region d'Occitanie and the École Doctorale I2S from the University of Montpellier (contract no. 00137932/22009671), the Foundation for Support of Research and Innovation, Santa Catarina (FAPESC-2021TR001907), the Brazilian National Council for Scientific and Technological Development (CNPq - processes 138179/2021-2, 140368/2021-3 and 350794/2023-5), and Project HARV (project PE24PR01) in the framework of the action "Accelerateur d'innovation" from the University of Montpellier.

Data Availability Statement: The data presented in this study are available from the corresponding authors upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Atakishiyev, S.; Salameh, M.; Yao, H.; Goebel, R. Explainable Artificial Intelligence for Autonomous Driving: A Comprehensive Overview and Field Guide for Future Research Directions. *IEEE Access* **2024**, *12*, 101603–101625. [CrossRef]
2. Bitkina, O.; Park, J.; Kim, H. Application of artificial intelligence in medical technologies: A systematic review of main trends. *Digital Health* **2023**, *9*, 20552076231189331. [CrossRef] [PubMed]
3. Bello, H. Towards certifiable AI in aviation: Landscape, challenges, and opportunities. *arXiv* **2024**, arXiv:2409.08666. http://xxx.lanl.gov/abs/2409.08666 (accessed on 16 September 2024).
4. Surianarayanan, C.; Lawrence, J.J.; Chelliah, P.R.; Prakash, E.; Hewage, C. A Survey on Optimization Techniques for Edge Artificial Intelligence (AI). *Sensors* **2023**, *23*, 1279. [CrossRef] [PubMed]
5. Li, W.; Liewig, M. A Survey of AI Accelerators for Edge Environment. In *Proceedings of the Trends and Innovations in Information Systems and Technologies*; Rocha, Á., Adeli, H., Reis, L.P., Costanzo, S., Orovic, I., Moreira, F., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 35–44.
6. Bui, T.A.; Lee, P.J.; Liang, C.S.; Hsu, P.H.; Shiu, S.H.; Tsai, C.K. Edge-computing-enabled deep learning approach for low-light satellite image enhancement. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 4071–4083. [CrossRef]
7. Kyrkou, C.; Plastiras, G.; Theocharides, T.; Venieris, S.I.; Bouganis, C.S. DroNet: Efficient convolutional neural network detector for real-time UAV applications. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 967–972. [CrossRef]
8. Guan, X.; Lou, S.; Li, H.; Tang, T. Intelligent control of quad-rotor aircrafts with a STM32 microcontroller using deep neural networks. *Ind. Robot. Int. J. Robot. Res. Appl.* **2021**, *48*, 700–709. [CrossRef]
9. Boudenot, J.C. Radiation space environment. In *Radiation Effects on Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–9.
10. Cannon, P.; Angling, M.; Barclay, L. Chapter 7 and 9—Radiation impacts on satellites and Ionising radiation impacts on avionics and ground systems. In *Extreme Space Weather: Impacts on Engineered Systems and Infrastructure*; Royal Academy of Engineering: London, UK, 2013.
11. Barth, J.L.; Dyer, C.; Stassinopoulos, E. Space, atmospheric, and terrestrial radiation environments. *IEEE Trans. Nucl. Sci.* **2003**, *50*, 466–482. [CrossRef]
12. Normand, E. Single event upset at ground level. *IEEE Trans. Nucl. Sci.* **1996**, *43*, 2742–2750. [CrossRef]
13. Jedec, J. Measurement and reporting of alpha particles and terrestrial cosmic ray-induced soft errors in semiconductor devices: JESD89A. *JEDEC Stand. JEDEC Sold State Technol. Assoc.* **2006**, *89*, 1–85.
14. Yang, M.; Hua, G.; Feng, Y.; Gong, J. *Fault-Tolerance Techniques for Spacecraft Control Computers*; John Wiley & Sons: Hoboken, NJ, USA, 2017.

15. Sorin, D. *Fault Tolerant Computer Architecture*; Morgan & Claypool Publishers: Williston, VT, USA, 2009.
16. Bruguier, G.; Palau, J.M. Single particle-induced latchup. *IEEE Trans. Nucl. Sci.* **1996**, *43*, 522–532. [CrossRef]
17. Koga, R.; Penzin, S.; Crawford, K.; Crain, W. Single event functional interrupt (SEFI) sensitivity in microcircuits. In Proceedings of the RADECS 97. Fourth European Conference on Radiation and its Effects on Components and Systems (Cat. No. 97TH8294), Cannes, France, 15–19 September 1997; IEEE: Piscataway Township, NJ, USA, 1997; pp. 311–318.
18. Ibrahim, Y.; Wang, H.; Liu, J.; Wei, J.; Chen, L.; Rech, P.; Adam, K.; Guo, G. Soft errors in DNN accelerators: A comprehensive review. *Microelectron. Reliab.* **2020**, *115*, 113969. [CrossRef]
19. Gutiérrez-Zaballa, J.; Basterretxea, K.; Echanobe, J. Evaluating single event upsets in deep neural networks for semantic segmentation: An embedded system perspective. *J. Syst. Archit.* **2024**, *154*, 103242. [CrossRef]
20. Santos, F.F.d.; Pimenta, P.F.; Lunardi, C.; Draghetti, L.; Carro, L.; Kaeli, D.; Rech, P. Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs. *IEEE Trans. Reliab.* **2019**, *68*, 663–677. [CrossRef]
21. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
22. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
23. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
24. Alippi, C.; Piuri, V.; Sami, M. Sensitivity to errors in artificial neural networks: A behavioral approach. *IEEE Trans. Circuits Syst. Fundam. Theory Appl.* **1995**, *42*, 358–361. [CrossRef]
25. Piuri, V. Analysis of Fault Tolerance in Artificial Neural Networks. *J. Parallel Distrib. Comput.* **2001**, *61*, 18–48. [CrossRef]
26. Furano, G.; Meoni, G.; Dunne, A.; Moloney, D.; Ferlet-Cavrois, V.; Tavoularis, A.; Byrne, J.; Buckley, L.; Psarakis, M.; Voss, K.O.; et al. Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities. *IEEE Aerosp. Electron. Syst. Mag.* **2020**, *35*, 44–56. [CrossRef]
27. Ahmadilivani, M.H.; Taheri, M.; Raik, J.; Daneshtalab, M.; Jenihhin, M. A Systematic Literature Review on Hardware Reliability Assessment Methods for Deep Neural Networks. *ACM Comput. Surv.* **2024**, *56*, 1–39. [CrossRef]
28. Benevenuti, F.; Libano, F.; Pouget, V.; Kastensmidt, F.L.; Rech, P. Comparative Analysis of Inference Errors in a Neural Network Implemented in SRAM-Based FPGA Induced by Neutron Irradiation and Fault Injection Methods. In Proceedings of the 2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI), Bento Goncalves, Brazil, 27–31 August 2018; pp. 1–6. [CrossRef]
29. Libano, F.; Wilson, B.; Wirthlin, M.; Rech, P.; Brunhaver, J. Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs. *IEEE Trans. Nucl. Sci.* **2020**, *67*, 1478–1484. [CrossRef]
30. Dos Santos, F.F.; Kritikakou, A.; Condia, J.E.R.; Guerrero-Balaguera, J.D.; Reorda, M.S.; Sentieys, O.; Rech, P. Characterizing a neutron-induced fault model for deep neural networks. *IEEE Trans. Nucl. Sci.* **2022**, *70*, 370–380. [CrossRef]
31. Blower, S.; Rech, P.; Cazzaniga, C.; Kastriotou, M.; Frost, C.D. Evaluating and Mitigating Neutrons Effects on COTS EdgeAI Accelerators. *IEEE Trans. Nucl. Sci.* **2021**, *68*, 1719–1726. [CrossRef]
32. Lin, W.; Adetomi, A.; Arslan, T. Low-Power Ultra-Small Edge AI Accelerators for Image Recognition with Convolution Neural Networks: Analysis and Future Directions. *Electronics* **2021**, *10*, 2048. [CrossRef]
33. Clay, M.; Grecos, C.; Shirvaikar, M.; Richey, B. Benchmarking the MAX78000 artificial intelligence microcontroller for deep learning applications. In Proceedings of the Real-Time Image Processing and Deep Learning 2022, Orlando, FL, USA, 3–7 April 2022; Kehtarnavaz, N., Carlsohn, M.F., Eds.; International Society for Optics and Photonics, SPIE: Bellingham, WA, USA, 2022; Volume 12102, p. 1210207. [CrossRef]
34. Moss, A.; Lee, H.; Xun, L.; Min, C.; Kawsar, F.; Montanari, A. Ultra-Low Power DNN Accelerators for IoT: Resource Characterization of the MAX78000. In Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems, New York, NY, USA, 6–9 November 2023; pp. 934–940. [CrossRef]
35. Nardello, M.; Caronti, L.; Brunelli, D. Intermittent Intelligent Camera with LEO sensor-to-satellite Connectivity. In Proceedings of the 11th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems, Istanbul, Turkiye, 12 November 2023; pp. 79–85.
36. Cazzaniga, C.; Frost, C.D. Progress of the scientific commissioning of a fast neutron beamline for chip irradiation. In *Proceedings of the Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2018; Volume 1021, p. 012037.
37. Integrated, M. Maxim Integrated AI Development. 2023. Available online: https://github.com/MaximIntegratedAI (accessed on 9 September 2024).
38. Caronti, L.; Akhunov, K.; Nardello, M.; Yıldırım, K.S.; Brunelli, D. Fine-grained hardware acceleration for efficient batteryless intermittent inference on the edge. *ACM Trans. Embed. Comput. Syst.* **2023**, *22*, 1–19. [CrossRef]
39. Peterson, W.W.; Brown, D.T. Cyclic Codes for Error Detection. *Proc. IRE* **1961**, *49*, 228–235. [CrossRef]
40. Castagnoli, G.; Brauer, S.; Herrmann, M. Optimization of cyclic redundancy-check codes with 24 and 32 parity bits. *IEEE Trans. Commun.* **1993**, *41*, 883–892. [CrossRef]
41. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
42. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. 2009. Available online: http://www.cs.toronto.edu/~kriz/cifar.html (accessed on 5 September 2024).

43. Amarnath, C.; Mejri, M.; Ma, K.; Chatterjee, A. Error Resilience in Deep Neural Networks Using Neuron Gradient Statistics. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *43*, 1149–1162. [CrossRef]
44. Luza, L.M.; Ruospo, A.; Söderström, D.; Cazzaniga, C.; Kastriotou, M.; Sanchez, E.; Bosio, A.; Dilillo, L. Emulating the effects of radiation-induced soft-errors for the reliability assessment of neural networks. *IEEE Trans. Emerg. Top. Comput.* **2021**, *10*, 1867–1882. [CrossRef]
45. Libano, F.; Rech, P.; Neuman, B.; Leavitt, J.; Wirthlin, M.; Brunhaver, J. How reduced data precision and degree of parallelism impact the reliability of convolutional neural networks on FPGAs. *IEEE Trans. Nucl. Sci.* **2021**, *68*, 865–872. [CrossRef]