Selective Hardening for Neural Networks in FPGAs

F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech

***Abstract*— Neural networks are becoming an attractive solution for automatizing vehicles in the automotive, military, and aerospace markets. Thanks to their low-cost, low-power consumption, and flexibility, field-programmable gate arrays (FPGAs) are among the promising devices to implement neural networks. Unfortunately, FPGAs are also known to be susceptible to radiation-induced errors. In this paper, we evaluate the effects of radiation-induced errors in the output correctness of two neural networks [Iris Flower artificial neural network (ANN) and Modified National Institute of Standards and Technology (MNIST) convolutional neural network (CNN)] implemented in static random-access memory-based FPGAs. In particular, we notice that radiation can induce errors that modify the output of the network with or without affecting the neural network's functionality. We call the former critical errors and the latter tolerable errors. Through exhaustive fault injection, we identify the portions of Iris Flower ANN and MNIST CNN implementation on FPGAs that are more likely, once corrupted, to generate a critical or a tolerable error. Based on this analysis, we propose a selective hardening strategy that triplicates only the most vulnerable layers of the neural network. With neutron radiation testing, our selective hardening solution was able to mask 40% of faults with a marginal 8% overhead in one of our tested neural networks.**

***Index Terms*— Field-programmable gate array (FPGA), hardening, neural networks, reliability.**

I. Introduction

FEED-FORWARD neural networks are computational approaches that have increasingly been adopted in many fields, including pattern recognition, high-performance computing, and data mining [1]. In addition, artificial neural networks (ANNs) and, more specifically, convolutional neural networks (CNNs) are extremely attractive for safety-critical applications, such as space exploration [2], autonomous driving [3], and military applications, such as unmanned aerial vehicles. These applications utilize a number of pattern recognition algorithms to identify and classify objects based on captured frames or signals. Most of these algorithms can be solved using ANNs and CNNs.

Due to the intrinsic parallelism between neurons and layers, the neural networks can be efficiently implemented on field-programmable gate arrays (FPGAs) [4]. Thanks to their low cost, low power consumption, and flexibility, FPGAs are an attractive solution for object detection in safety-critical applications. Unfortunately, FPGAs have been shown to have high-radiation sensitivity [5]. In particular, static random-access memory (SRAM)-based FPGAs may experience single-event upsets (SEUs) in the configuration memory that can change the configuration of a routing connection, the configuration of a Lookup table (LUT), or the configuration of a Block RAM (BRAM).

In this paper, we analyze the reliability of neural networks implemented in SRAM-based FPGAs. As case studies, we consider an *Iris Flower* ANN, which classifies flower subspecies, and a *Modified National Institute of Standards and Technology (MNIST)* CNN, which recognizes handwritten digits. Although neither of these networks is used in safety-critical applications, they share the same topology and the same computational flow with more complex object detection frameworks. The Iris Flower ANN is a fully connected network used to make decisions based on the preprocessed information (object attributes). The MNIST CNN has, in addition to the fully connected part, a few convolution and pooling layers beforehand, in order to extract pertinent attributes from the raw input image. Because of this, the MNIST CNN is the kind of network used in autonomous vehicles to process images and detects objects. In our case study, the MNIST CNN processes a data set of handwritten digits.

Our results show that not all the faults that reach the network output have a significant impact on the application. Thanks to the intrinsic approximate nature of neural network computation, some output errors are sufficiently close to the expected value that the application treats them as correct. Through extensive fault-injection campaigns, we identify the causes of *tolerable* and *critical* errors (i.e., the circuit portions that, once corrupted, are likely to significantly modify the network's answer).

Although the majority of errors are tolerable, for some applications, it could be essential to reduce as much as possible the probability of critical errors. Triple modular redundancy (TMR) has already been shown to be a very effective hardening solution for FPGAs. However, the very high amount of LUTs required to implement neural networks in FPGAs significantly reduces the extra resources that could be used to apply TMR. Moreover, as most of the errors are tolerable, simply applying TMR to the neural network as a

Manuscript received October 15, 2018; revised November 25, 2018; accepted November 28, 2018. Date of publication November 30, 2018; date of current version January 17, 2019. This work was supported in part by CAPES Foundation, Ministry of Education, in part by CNPq Research Council, Ministry of Science and Technology, in part by the Department of Energy of the United States, and in part by the Science and Technology Council, Oxfordshire OX11 0QX, U.K.

F. Libano and P. Rech are with the Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre 90040-060, Brazil (e-mail: fplibano@inf.ufrgs.br; prech@inf.ufrgs.br).

B. Wilson, J. Anderson, and M. J. Wirthlin are with Brigham Young University, Provo, UT 84602 USA (e-mail: brittany.wilson@byu.edu; jordan.anderson@byu.edu; wirthlin@byu.edu).

C. Cazzaniga and C. Frost are with the Science and Technology Facility Council, Oxfordshire OX11 0QX, U.K. (e-mail: christopher.frost@stfc.ac.uk; carlo.cazzaniga@stfc.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2018.2884460

whole could result in a very inefficient solution. With that in mind, we design and apply *selective* TMR only to those layers in the networks that have been found to be more vulnerable (i.e., their corruption is more likely to generate an error). As discussed in the paper, we find that, for both networks, each layer has its own vulnerability. In fact, this was already observed in prior work with graphics processing units (GPUs) [6], but, to the best of our knowledge, it is a new finding for neural networks in FPGAs. In particular, we can identify a subset of layers that are more likely to generate critical output errors, thus becoming targets for the selective hardening. We validate our hardening solution through controlled neutron beam experiments. One of our selectively hardened ANNs masked 68% of faults with a 45% area overhead, while the other achieved 40% fault masking with only 8% overhead. Thus, we see that alternatives to full TMR can offer better tradeoffs.

The remainder of the paper is organized as follows. Section II provides a background on neural networks in general and discusses the data sets and topologies of our two case studies. Section III mentions the tested devices, discusses the details on both fault injection and beam experiments, and introduces the concept of error criticality. Section IV presents our proposed selective hardening strategy, while Section V shows our experimental results, where we validate and compare the effectiveness of our hardening approach to more traditional methods. Section VI concludes the paper and briefly mentions our plans for future work.

II. Background

A. Artificial and Convolutional Neural Networks

An ANN computes a solution by propagating data through layers of neurons connected and interacting via synapses (similar to the way a biological brain solves problems). Data streams from input to output, without feedback. A CNN is an ANN that performs filter operations to extract information from the input data. The ANNs need to be trained using a sufficiently complex and representative set of data instances. During the training phase, the weights of every connection between the neurons and the weights of filters (when used) are tuned. Then, when a new input is given, the network computes a solution based on the weights learned through training. We should highlight that our models were trained using the *Caffe* framework [7].

Neural networks are organized as the *layers* of connected neurons. For CNNs, before the fully connected layers, there are convolution layers (multiply and sum), pooling layers (only some elements are selected to propagate), and activation layers (to add nonlinearity to the model and to normalize data). In this paper, we consider a simple two-layer ANN and a much larger seven-layer CNN with two convolutional layers, two pooling layers, two inner product layers (i.e., traditional fully connected layers), and one activation layer (ReLU). To put them in perspective, the Iris Flower ANN has 19 weights, while the MNIST CNN has 1848 weights. Their respective topologies are described in Sections II-B and II-C.

Preliminary works have evaluated the reliability of CNNs executed in specific hardware (ASIC) [8], [9] and GPUs [6].

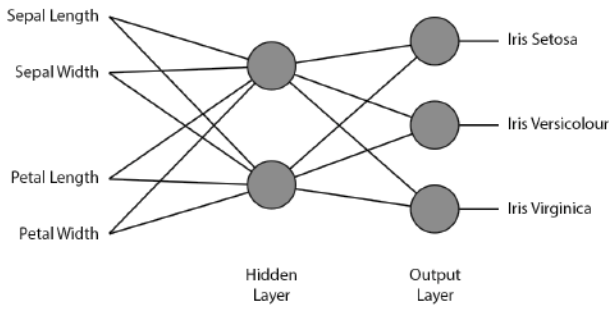

Fig. 1. Iris Flower ANN's topology.

These studies show that CNNs' layers have considerably different levels of sensitivity and, thus, should be treated accordingly when applying hardening techniques. This paper focuses, within other topics, in evaluating the sensitivity of the neural networks' layers in FPGAs, an aspect that has not been investigated, yet. We also design and validate a selective hardening strategy to efficiently improve neural networks reliability in FPGAs.

B. Iris Flower

The Iris Flower data set quantifies the morphologic variation of three related species: *Iris Setosa*, *Iris Versicolor*, and *Iris Virginica* [10], [11]. Each instance in the input correlates physical characteristics of a given flower to its species. This relation is modeled by the ANN illustrated in Fig. 1.

We should highlight that, due to its small topology, the Iris Flower ANN is not used in a safety-critical application. Nonetheless, we have decided to evaluate its reliability, as a supplementary case study, since it provides useful insights into the effectiveness of selective triplication.

A value is generated for each output, and the output with the highest value identifies the species classification. We should point out that the neurons represented in the hidden and output layers compose the two fully connected layers.

C. MNIST

The MNIST is a data set of 28×28 pixel images of handwritten digits (ranging from 0 to 9) [12]. The MNIST CNN receives a matrix of 28×28 inputs (one for each pixel) and produces 10 outputs (one for each digit). It uses convolution to extract specific information from the input image. Some parameters of the CNN, such as the number and size of convolutional filters can be adjusted to improve accuracy. Our tested CNN has an accuracy of about 94%, utilizing the following topology (also illustrated in Fig. 2):

1) Input = 784 pixels;
2) ConvLayer1 (one 4×4 Filter);
3) PoolLayer1 (3×3 Filter);
4) ConvLayer2 (three 4×4 Filters);
5) PoolLayer2 (2×2 Filter);
6) InnerProduct1 (48 Inputs, 30 Outputs);
7) ReLU1 (30 Inputs, 30 Outputs);
8) InnerProduct2 (30 Inputs, 10 Outputs); and

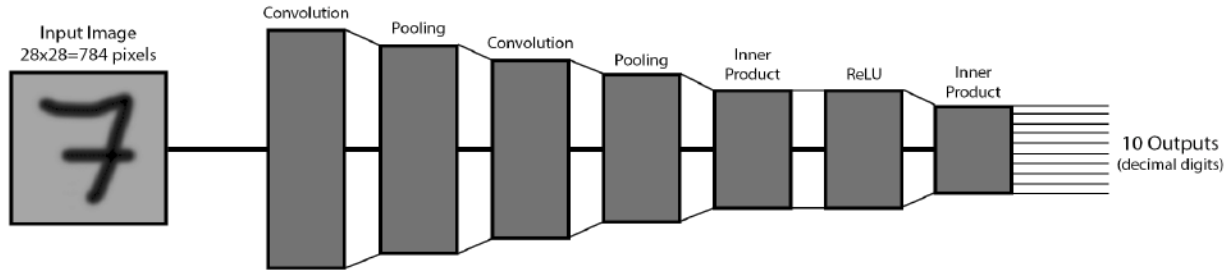

Fig. 2. MNIST CNN's topology.

TABLE I
ZYNQ-7000 AND ZYNQ ULTRASCALE+ RESOURCE UTILIZATION TO IMPLEMENT THE IRIS FLOWER ANN AND THE MNIST CNN, RESPECTIVELY

Zynq Device	Hardening	FF	DSP	LUT
7000 (Iris Flower)	Unhardened	3,5k (4%)	56 (25%)	16k (30%)
	Selective TMR	3,7k (4%)	56 (25%)	25k (47%)
	Full TMR	4,1k (4%)	168 (75%)	43k (81%)
Ultrascale+ (MNIST)	Unhardened	21k (4%)	0 (0%)	193k (70%)
	Selective TMR	21k (4%)	0 (0%)	206k (75 %)
	Full TMR	–	–	–

9) Output = the highest value between the last 10 outputs is the digit's classification (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

III. EXPERIMENTAL METHODOLOGY

We implemented the Iris Flower ANN on a 28-nm Zynq-7000 (CLG484 XC7Z020) [13], which is composed of a processing system (PS) formed around a dual-core ARM Cortex-A9 processor and programmable logic (PL) based on a Xilinx Artix-7 FPGA. For the MNIST CNN, which requires about five times the resources as the Iris Flower, we use the 16 nm FinFET Zynq Ultrascale+ multiprocessor system-on-chip (ZU9EG), also composed of a PS and a PL. The details about resource utilization by the two neural networks in their respective devices can be seen in Table I. We should point out that the training phases for both of our networks were performed before the experiments, in a fault-free environment. In other words, the neural network training was not affected by any upsets.

A. Fault-Injection Framework

For the Iris Flower ANN, we use the fault-injection framework presented in [14], which relies on the internal configuration access port (ICAP) block of Xilinx FPGAs, an ICAP controller, and a monitor computer. Using the read and write capabilities of the ICAP block, we can emulate SEUs in the configuration memory. For the MNIST CNN, we use the processor configuration access port of the MPSoC system and emulate SEUs by inserting upsets through software (SW) executed on the A53 quadcore processor.

In both experiments, we isolate each of the neural network layers into separate regions of the device using "Pblocks." The location of the injected fault can be correlated with the Pblock and, thus, a specific layer of the network. For the Iris Flower ANN, an exhaustive fault-injection campaign produces bitflips on *all* the configuration bits of the injection area, one at a time. For the MNIST CNN, a fault-injection campaign randomly injected about 13 million faults.

The fault-injection area is comprised of only the configuration bits related to configurable logic blocks [LUTs, digital signal processors (DSPs), flip-flops (FFs), and interconnections]. The BRAM configuration bits are not considered. During the fault injection, a microprocessor analyzes the correctness of neural network execution, and an error is detected when the result differs from the expected values. Observed errors are then classified based on the discussion in Section IV.

B. Neutrons Experimental Setup

Our radiation experiments were conducted at the Los Alamos Neutron Science Center (LANSCE) (Fig. 3) facility of the Los Alamos National Laboratory (LANL) and the ChipIR facility (Fig. 4) at ISIS, Didcot, U.K. Both LANSCE and ChipIR provide a neutron spectrum that mimics the atmospheric neutron one and have already been demonstrated to be suitable to emulate terrestrial neutrons effects in electronic devices and systems. To avoid any result bias due to the facility, we tested the Zynq-7000 implementing the Iris Flower ANN only at LANSCE and the Ultrascale+ MPSoC implementing the MNIST CNN only at ChipIR.

The neutron flux at LANSCE and ChipIR is about eight to nine orders of magnitude higher than the terrestrial flux [$13n/(\text{cm}^2 \times h)$] at sea level [15], [16]). The Zynq-7000 device was tested for approximately 40 h at LANSCE, which is equivalent to about 1.5 million years of natural exposure. The Ultrascale+ MPSoC was tested for about 20 h at ChipIR, which is equivalent to more than 2.5 million years of natural exposure.

C. Error Criticality

As we previously mentioned, when dealing with ANNs and CNNs whose purpose is to work as classifiers, not every error is considered critical. This means that, even if the output is different than the expected one, the classification of a given instance/image might still be correct. Based on these considerations we identify two possible error classes on the tested networks.

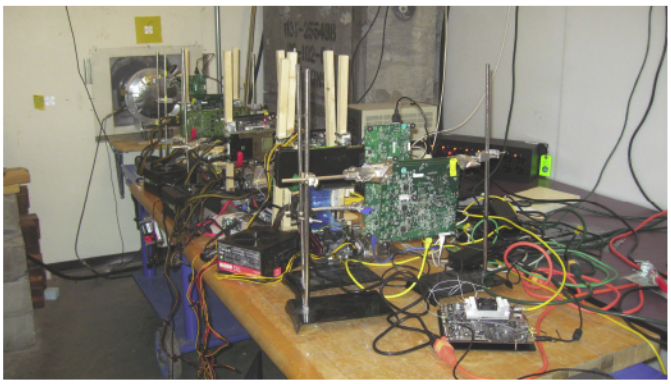

Fig. 3. Radiation experiment at the LANSCE facility of the LANL, USA.

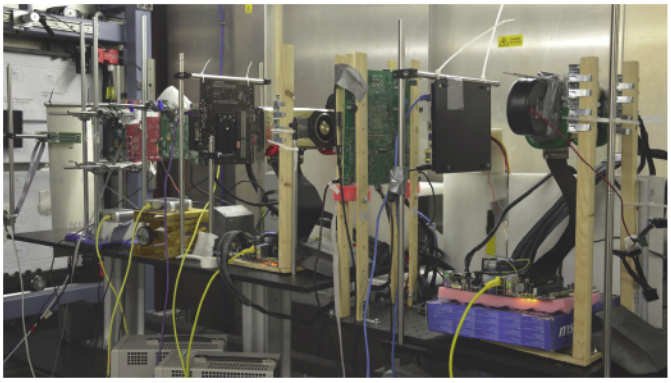

Fig. 4. Radiation experiment at the ChipIR facility of the Rutherford Appleton Laboratory, U.K.

1) *Tolerable:* the network produces outputs different than the expected ones, but the application's behavior can still be considered correct.
2) *Critical:* the network produces output errors, and they are severe enough to compromise the instance classification.

These concepts can also be expressed in a more formal/algorithmic way:

```
if (computedClassification == expectedClassification) then
  errorType = tolerable
else
  errorType = critical
end if
```

In Section V, we classify all of our experimental results with fault injection and radiation errors as *Tolerable* or *Critical*. Although the full circuit TMR would mask all errors (including the tolerable ones), our selective hardening strategy is focused on reducing *critical* occurrences. We then expect to achieve high reliability without unnecessary overhead.

IV. Selective Hardening

TMR is one of the most effective solutions to mitigate transient errors in FPGAs [17], [18], particularly common for safety-critical applications [19]. Although TMR is extremely effective, it also imposes a nonnegligible overhead in terms of area and static/dynamic power consumption. Our idea is to improve the efficiency of TMR by protecting only those layers of a neural network which are most critical. A critical layer is a layer that, once corrupted, is more likely to modify the network output in a way that impacts the correct application execution.

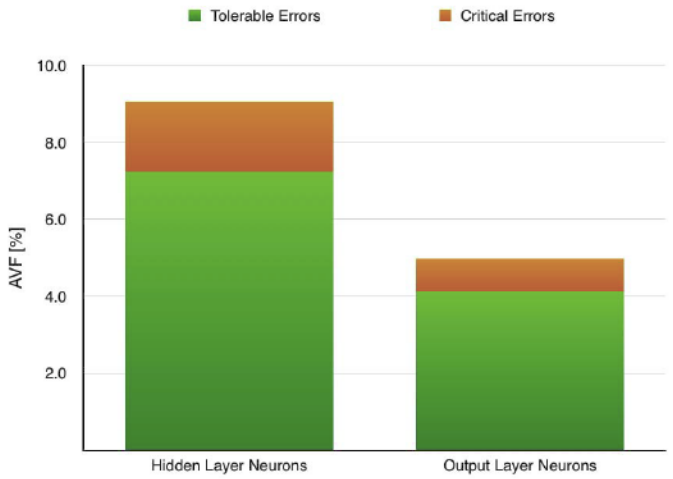

Fig. 5. AVF calculated from fault injection in different layers of the Iris Flower ANN.

Selective hardening is promising for ANNs and CNNs as not all the errors at the network output are critical; a corrupted output can still maintain a correct behavior. To illustrate the concept of critical errors let us consider the Iris Flower ANN. If any of the three outputs is different than expected, but the flower classification is still correct, the error is considered *tolerable*, otherwise *critical*. The same applies to the MNIST CNN, meaning that a minor difference in any of the 10 outputs can still maintain the overall classification of the image.

V. Experimental Results

A. Fault-Injection Results

Using the frameworks described in Section III-A, we inject faults in the networks' layers in order to comprehend the propagation of such faults through the connections, allowing us to rank the networks' layers by criticality.

Fig. 5 shows the results obtained through our fault injection in the Iris Flower ANN. They are expressed as architectural vulnerability factor (AVF), i.e., the percentage of injections that propagate to the output. We divide the fault-injection outcomes into tolerable and critical errors, as discussed in Section IV. Faults were separately injected in the hidden layer (HL), and output layer neurons of the ANN.

From Fig. 5, it is clear that faults in the HL are more likely to generate tolerable and critical errors in the output. This is because when a fault is closer to the input, it has more room to propagate through the fully connected topology of the ANN. In addition, the tolerable errors are about four times more frequent than critical errors.

In Fig. 6, we present the AVF for MNIST CNN layer. Unlike the Iris Flower ANN, the MNIST CNN is composed of layers with different functions and sizes. Thus, the sensitivity of each layer depends on the layer type and the size in addition to its position within the network.

As shown in Fig. 6, even for the MNIST CNN, the dominance of tolerable errors over critical errors holds, suggesting

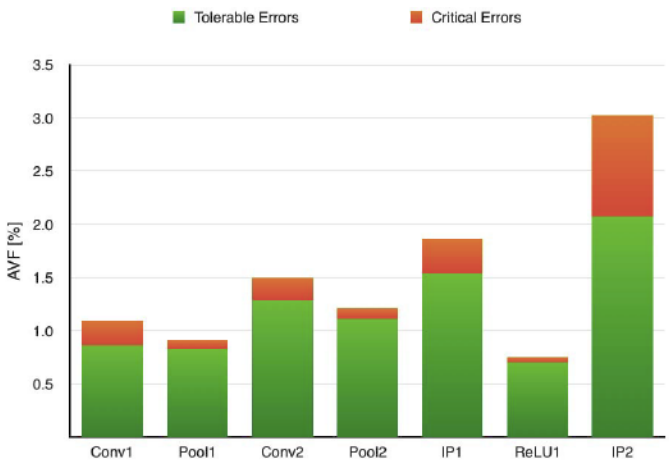

Fig. 6. AVF calculated from fault injection in different layers of the MNIST CNN.

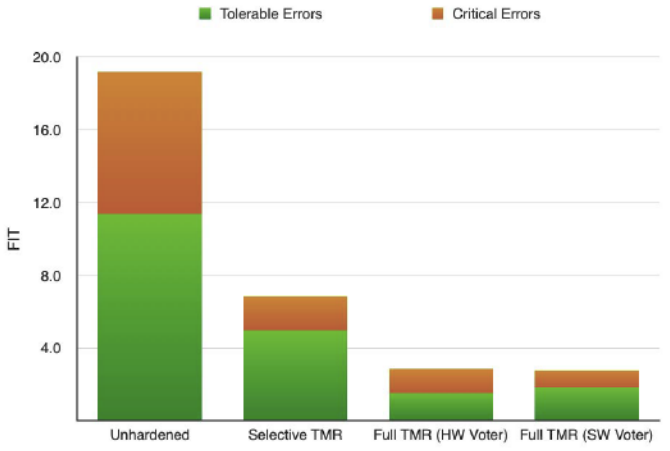

Fig. 7. FIT rate of the Iris Flower ANN using four different hardening configurations.

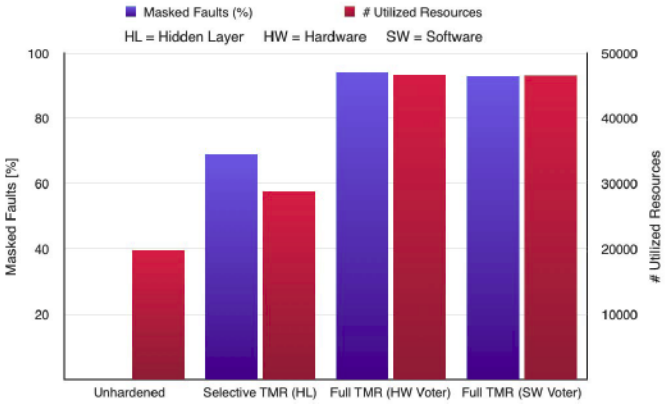

Fig. 8. Masked faults percentage and resource utilization of the four different hardening configurations of the Iris Flower ANN.

that this result could be a general characteristic of pattern recognition feed-forward neural networks. In addition, it is evident that a fault in InnerProduct2 (IP2) layer (a fully connected layer) has a considerably greater chance of propagating to the output, making it a great candidate for our selective hardening approach.

B. Proposed Selective Hardening

Based on our fault-injection campaigns, we conclude that the criticality of a given layer in a neural network is determined by its function, size, and position. In both cases, there is a considerable gap in vulnerability across the different layers, which motivates the idea of selective hardening.

We design four configurations of the Iris Flower.

1) *Unprotected:* the plain ANN, without any hardening.
2) *Selective TMR:* the ANN, with a triplication of the HL alone.
3) *Full TMR (HW Voter):* the ANN fully triplicated, with the voter implemented in the PL part (FPGA).
4) *Full TMR (SW Voter):* the ANN fully triplicated, with the voter implemented in the PS part (ARM).

We initially chose to evaluate the two voting mechanisms to see if there was an advantage in terms of reliability, by delegating the task to the processor (consequently, reducing resource utilization in the FPGA). However, as we will show in Section V-C, there was a negligible difference between the two designs, so we decided not to test the *SW Voter* version with the MNIST CNN.

C. Radiation Experiment Results

With the setup described in Section III-B, we have irradiated the Zynq-7000 device for about 40 h, for a total fluence of $1,6 \times 10^{11}$ neutrons/cm^2.

In Fig. 7, we can see the FIT rate of the FPGA executing the Iris Flower ANN. Statistical error at 95% confidence is lower than 15% of reported values.

The cross section is the ratio between the number of observed errors and the received particles fluence. Thus, the cross section is expressed in a unit of an area (square centimeters) and is directly proportional to the probability for an impinging particle to generate an observable error. Multiplying the cross section by the particle flux that the device will experience on a realistic application ($13n/(\text{cm}^2 \times h)$ [15]) provides the expected error rate of the device expressed in failure in time (FIT), i.e., errors per 1 billion hour of operation.

Interestingly, the trends observed in fault injection and beam experiments (unprotected version) are similar. This may suggest that the criticality of errors in the ANNs is an intrinsic property of the network and is not dependent on the source of error or the probability of data corruption. The trend changes for the hardened versions, as most faults are masked.

We can clearly perceive, from Fig. 7, that the probability of error occurrence for the selective hardened version decreases by 64.2% with respect to the unhardened version. Similarly, the percentage reductions considering the full TMR configurations are 85.1% and 85.6% for the voter implemented in the FPGA and in the ARM processor, respectively. A promising result is that the selective TMR and full TMR with SW voters significantly reduce the number of critical errors.

The full TMR versions outperform the selective TMR configurations in terms of pure reliability, which is to be expected. Fig. 8 also shows the number of used resources (LUTs, DSPs, and FFs) for all the different designs. As shown,

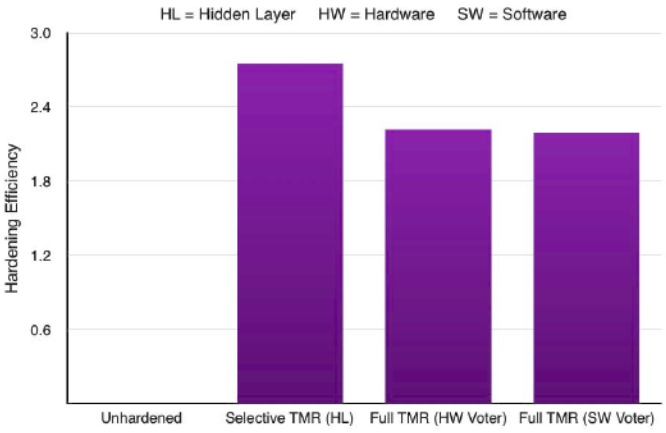

Fig. 9. Percentage of the masked faults divided by resource utilization of the four different hardening configurations of the Iris Flower ANN.

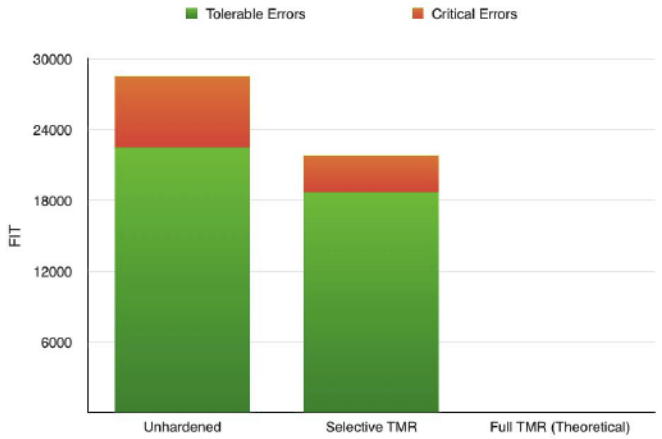

Fig. 10. FIT rate of the MNIST CNN using three different hardening configurations.

the overhead of a full TMR is much bigger (almost 2×) than the one of the selective TMR. In other words, with a 45% increase in resource usage our selective hardening achieves 68% of fault masking, while the traditional TMR approaches require around a 200% increase in area to reach near 100% rate of fault masking (94% in this case).

If we consider the tradeoff between the percentage of the masked faults and the resource utilization, our selective hardening technique is shown to be more efficient. As shown in Fig. 9, the ratio between the masked faults and the number of resources used is 25% higher for the selective TMR.

In the case of the MNIST CNN, our target layer for selectively applying TMR was the very last one (called *IP2*). As an insight from our fault-injection campaign, highlighted in Fig. 6, the IP2 layer appears as the most vulnerable one in this neural network.

We then tested two versions of the MNIST CNN in our beam experiments: the unhardened neural network and the selective TMR one (with the IP2 layer fully triplicated). We did not test the full TMR version as we did with the Iris Flower ANN, simply because it was not feasible resource wise, meaning that there were not enough available LUTs in the Zynq Ultrascale+ device to fully triplicate the MNIST CNN. This fact is showcased on the very last line of Table I, and it reinforces the importance of selective hardening as a smarter, more efficient use of additional resources. We have irradiated the Zynq Ultrascale+ device for about 20 h, with a total fluence of $3,0 \times 10^{11}$neutrons/cm^2.

Fig. 10 shows the FIT rate of the MNIST CNN. We can clearly see that the selective TMR reduced the probability of occurrence of both the tolerable and critical errors (precisely, 14% reduction of the latter). In addition to that, we have also put Fig. 11 here to highlight the percentage of the masked faults by each design (*benefit*, in blue) along with their absolute number of utilized resources (*cost*, in red). Note that our selective hardening approach achieves around 40% fault masking with a marginal overhead of only 8%. Also, note that, as we mentioned before, it was not possible to test the full TMR version, but we completed the graph with

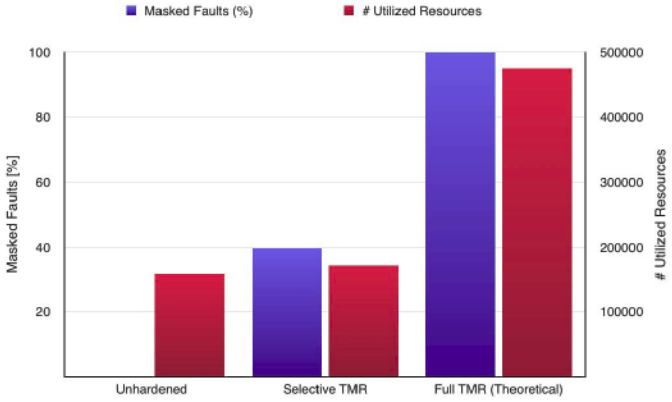

Fig. 11. Masked faults percentage and resource utilization of the three different hardening configurations of the MNIST CNN.

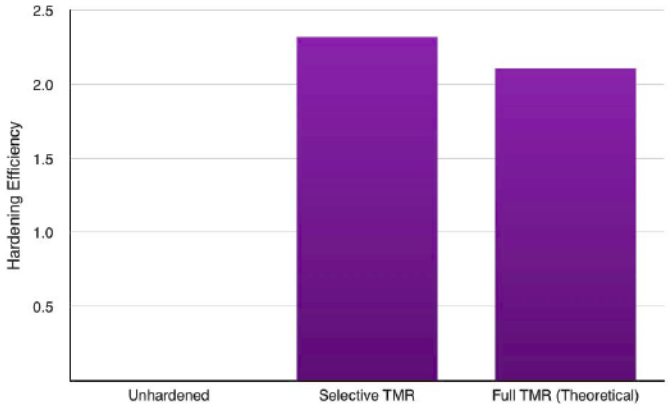

Fig. 12. Percentage of the masked faults divided by resource utilization of the three different hardening configurations of the MNIST CNN.

theoretical values (near 100% fault masking and with near 200% overhead).

As we did with the Iris Flower ANN, we have added a hardening efficiency graph for the MNIST CNN (Fig. 12). This graph was generated simply by dividing the values in Fig. 11, so it represents the ratio between the benefit and cost of each hardening strategy. We can see that our selective hardening approach comes out on top once more, leaving the traditional

TMR behind (even if this is a calculated theoretical efficiency that considers perfect fault masking using full TMR).

VI. Conclusion

We have seen that, for both ANNs and CNNs, the tolerable errors are dominant over critical errors. This means that most of the masked faults of a full TMR will not jeopardize the application's functionality, indicating that this traditional hardening strategy may not be optimal for neural networks. Furthermore, we have noticed that the overhead of full triplication might even be too high to be feasible, which strengthens our argument about the importance of determining the most critical layers in each neural network, and adopting more efficient hardening solutions. Another interesting insight is that our selective hardened designs showed considerable reductions on the probability of occurrence of critical errors, as highlighted by Figs. 7 and 10.

Overall, we saw that selectively applying TMR to the right layers of neural networks proved to be an efficient hardening solution, since it delivers the best tradeoff between reliability (through fault masking) and resource utilization. In addition to that, a very promising part of our experimental results is that our strategy was equally efficient in two very different neural networks. We should be aware that the topology's different parameters are always directly related to the format and the complexity of the instances in the data set we want to process. Our case studies (data sets and networks) are considerably distinct from each other; yet, their overall reliability improvement followed a similar trend as a result of applying selective TMR. This suggests that the selective hardening proposed by us seems to be fairly generic, and might bring the best hardening efficiency in many other neural networks as well.

As future work, we intend to evaluate the impact of triplicating not only the most vulnerable layer of a given neural network but also protecting the second, third, and so on. We believe that this iterative approach might enhance the efficiency of our selective hardening. We also plan on implementing the algorithm-based fault-tolerance techniques for each type of layer in CNNs since we have seen that this is a viable option in GPUs for increasing reliability [6].

References

[1] S. U. Amin, K. Agarwal, and R. Beg, "Genetic neural network based data mining in prediction of heart disease using risk factors," in *Proc. IEEE Conf. Inf. Commun. Technol.*, Apr. 2013, pp. 1227–1231.

[2] G. R. Allen *et al.*, "2017 compendium of recent test results of single event effects conducted by the jet propulsion laboratory's radiation effects group," in *Proc. IEEE Radiat. Effects Data Workshop (REDW)*, Jul. 2017, pp. 1–9.

[3] V.-E. Neagoe, A.-D. Ciotec, and A.-P. Bărar, "A concurrent neural network approach to pedestrian detection in thermal imagery," in *Proc. 9th Int. Conf. Commun. (COMM)*, Jun. 2012, pp. 133–136.

[4] C. He, A. Papakonstantinou, and D. Chen, "A novel SoC architecture on FPGA for ultra fast face detection," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2009, pp. 412–418.

[5] M. Wirthlin, "High-reliability FPGA-based systems: Space, high-energy physics, and beyond," *Proc. IEEE*, vol. 103, no. 3, pp. 379–389, Mar. 2015.

[6] F. F. D. Santos, L. Draghetti, L. Weigel, L. Carro, P. Navaux, and P. Rech, "Evaluation and mitigation of soft-errors in neural network-based object detection in three GPU architectures," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2017, pp. 169–176.

[7] Y. Jia *et al.* (2014). "Caffe: Convolutional architecture for fast feature embedding." [Online]. Available: https://arxiv.org/abs/1408.5093

[8] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. 55th ACM/ESDA/IEEE Design Automat. Conf. (DAC)*, New York, NY, USA, Jun. 2018, pp. 1–6, doi: 10.1145/3195970.3195997.

[9] G. Li *et al.*, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, New York, NY, USA, Nov. 2017, p. 8, doi: 10.1145/3126908.3126964.

[10] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[11] E. Anderson, "The irises of the gaspe peninsula," *Bull. Amer. Iris Soc.*, vol. 59, pp. 2–5, 1935.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[13] Xilinx. *Zynq-7000 All Programmable SoC*. Accessed: Nov. 2018. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[14] J. Tonfat, L. Tambara, A. Santos, and F. Kastensmidt, "Method to analyze the susceptibility of HLS designs in SRAM-based FPGAS under soft errors," in *Proc. Int. Symp. Appl. Reconfigurable Comput.*, vol. 9625. Berlin, Germany: Springer-Verlag, 2016, pp. 132–143.

[15] *Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors in Semiconductor Devices*, JEDEC Standard JESD89A, 2006. [Online]. Available: https://www.jedec.org/sites/default/files/docs/jesd89a.pdf

[16] C. Cazzaniga and C. D. Frost, "Progress of the scientific commissioning of a fast neutron beamline for chip irradiation," *J. Phys. Conf. Ser.*, vol. 1021, no. 1, p. 012037, May 2018. [Online]. Available: http://stacks.iop.org/1742-6596/1021/i=1/a=012037

[17] S. D'Angelo, C. Metra, S. Pastore, A. Pogutz, and G. R. Sechi, "Fault-tolerant voting mechanism and recovery scheme for TMR FPGA-based systems," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Nov. 1998, pp. 233–240.

[18] L. Sterpone and M. Violante, "Analysis of the robustness of the TMR architecture in SRAM-based FPGA," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 5, pp. 1545–1549, Oct. 2005.

[19] J. R. Azambuja, F. Sousa, L. Rosa, and F. L. Kastensmidt, "Evaluating large grain TMR and selective partial reconfiguration for soft error mitigation in SRAM-based FPGAs," in *Proc. 15th IEEE Int. On-Line Test. Symp.*, Jun. 2009, pp. 101–106.