Understanding the Impact of Quantization, Accuracy, and Radiation on the Reliability of Convolutional Neural Networks on FPGAs

F. Libano, B. Wilson, M. Wirthlin, P. Rech, and J. Brunhaver

***Abstract*— Convolutional neural networks are quickly becoming viable solutions for self-driving vehicles, military, and aerospace applications. At the same time, due to their high level of design flexibility, reprogrammable capability, low power consumption, and relatively low cost, the field-programmable gate arrays (FPGAs) are very good candidates to implement the neural networks. Unfortunately, the radiation-induced errors are known to be an issue in static random-access memory (SRAM)-based FPGAs. More specifically, we have seen that particles can change the content of the FPGA's configuration memory, consequently corrupting the implemented circuit and generating the observable errors at the output. Through extensive fault injection, we determine the reliability impact of applying binary quantization to the convolutional layers of neural networks on FPGAs, by analyzing the relationships between model accuracy, resource utilization, performance, error criticality, and radiation cross section. We were able to find that a design with quantized convolutional layers can be 39% less sensitive to radiation, whereas the portion of errors that are considered critical (misclassifications) in the network is increased by 12%. Moreover, we also derive generic equations that consider both accuracy and radiation in order to model the overall failure rate of neural networks.**

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

I. Introduction

CONVOLUTIONAL neural networks (CNNs) are computational solutions that have been growing in terms of adoption in many fields that depend mainly on pattern recognition and image processing tasks [1]. More specifically, CNNs are very attractive in the safety-critical niche, which comprises space exploration [2], self-driving cars [3], and military applications such as unmanned aerial vehicles (UAVs) [4]. Most of the algorithms involved in these applications aim to identify and classify the objects of interest, based on captured frames and various signals. Given that CNNs specialize in interpreting and reasoning about raw data in a very efficient way, they tend to be an optimal choice in most cases.

As we know, there is an intrinsic level of parallelism between neurons and layers in a neural network, which means that they can be efficiently implemented on field-programmable gate arrays (FPGAs) [5]. Thanks to their design flexibility, and typically low power consumption, FPGAs are a very attractive solution for a number of safety-critical tasks. Unfortunately, FPGAs have been shown to be highly sensitive to radiation [6]. In particular, static random-access memory (SRAM)-based FPGAs may experience single-event upsets (SEUs) in their configuration memory, which can affect routing connections, lookup tables (LUTs), flip-flops (FFs), and block RAMs (BRAMs).

In order to deliver high accuracy, CNNs end up being very computationally expensive, which means that, on FPGAs, they require a large number of resources. As an attempt to speed-up neural networks, weight quantization strategies, such as binarization, have emerged [7], as we further discuss in Section II. One of the goals of this article is to analyze how binary quantization in convolutional layers affects the overall reliability of neural networks implemented in SRAM-based FPGAs. To do so, we evaluate the trade-offs between model accuracy, resource utilization, execution time, architectural vulnerability, and radiation sensitivity. As a case study, we consider the Modified National Institute of Standards and Technology (MNIST) CNN, which recognizes the handwritten digits on 28×28 pixel grayscale images.

By performing extensive fault injection experiments, we are able to identify which portions of the CNN's circuit (when corrupted) are more likely to generate either *tolerable* or *critical* errors at the output. In particular, we are able to identify whether the feature extraction phase is more vulnerable than the classification process. Then, using radiation experimental data, we can estimate the likelihood of one impinging particle to generate an observable error at the output of the CNN and accurately calculate the expected error rate in a given radiation environment (terrestrial or space).

The remainder of the article is organized as follows. Section II gives a background on CNNs, discusses the benefits and drawbacks of binary quantization, and presents both the data set and the topology of the case study. Section III gives the details about the FPGA device used for testing and explains details about the fault injection framework. Section IV presents and discusses the implications of the experimental results and

Manuscript received February 17, 2020; revised March 20, 2020 and March 23, 2020; accepted March 24, 2020. Date of publication March 26, 2020; date of current version July 16, 2020. This work was supported in part by the Department of Energy of the United States, in part by the CAPES Foundation of the Ministry of Education, and in part by the CNPq Research Council of the Ministry of Science and Technology.

F. Libano and J. Brunhaver are with the School of Electrical, Computer and Energy Engineering (ECEE), Arizona State University (ASU), Tempe, AZ 85287 USA (e-mail: flibano@asu.edu; jbrunhaver@asu.edu).

B. Wilson and M. Wirthlin are with the Department of Electrical and Computer Engineering, Brigham Young University (BYU), Provo, UT 84602 USA (e-mail: brittany.wilson@byu.edu; wirthlin@byu.edu).

P. Rech is with the Los Alamos National Laboratory (LANL), Los Alamos, NM 87545 USA, and also with the Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre 90040-060, Brazil (e-mail: prech@lanl.gov; prech@inf.ufrgs.br).

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

Digital Object Identifier 10.1109/TNS.2020.2983662

proposes a set of equations to model the failure rate of neural networks. Section V concludes this article and mentions a few possibilities for future work.

II. Background

A. Convolutional Neural Networks

An artificial neural network (ANN) computes its solution by propagating the data through layers of interconnected neurons (which is a process very similar to what actually occurs in a biological brain). A CNN is a special kind of ANN, which can be broken down into two distinct parts or phases: *feature extraction classification*. During the feature extraction part, the filters are convolved with the raw input image, and the most important features of the given input are propagated forward. During the classification part, a series of fully connected neurons uses these extracted features in order to accurately guess in which class the raw input image belongs to. Both the filters and neurons have associated weights to them, which are learned during an extensive training process. In such a process, a training set is presented iteratively to the model, which makes adjustments to its weight values every step of the way, up to the point where it converges to an acceptable level of accuracy. Then, whenever a new input is given, the trained network can compute its solution by executing a series of arithmetic operations involving the set of weights that were learned during the training phase.

The previous works have analyzed the reliability of CNNs executing in specific hardware [application-specific integrated circuits (ASICs)] [8], [9] as well as in graphics processing units (GPUs) [10], and in FPGAs [11]–[13]. Such studies show that there is a considerable difference in sensitivity across different layers of a CNN. This article focuses, within other topics, in evaluating the architectural vulnerability using a higher level of granularity, by dividing the CNN into only two parts (feature extraction and classification), instead of looking at each individual layer, a novel point of view for FPGAs.

B. Binary and Hybrid Neural Networks

While CNNs can be very effective, they also require extremely high computational power. In order to achieve lower execution times and, thus, higher throughput, a number of techniques have been developed, such as weight trimming [14] and weight quantization [15]. When it comes to the latter, the main idea is to reduce the precision, in which we choose to represent the trained weights (which are originally in 32-bit floating-point format, as a standard for training frameworks). Such reduction can be completely arbitrary, going as far as utilizing a single bit to represent the weights in a model. These are called binary neural networks (BNNs), where both the filters in the convolutional layers as well as the neurons in ithe fully connected layers use weights constrained to $\{-1, 1\}$. The adoption of BNNs instead of CNNs essentially eliminates all multiplications for a hardware implementation of a given neural network, which decreases resource utilization, but also brings down the accuracy of models.

In fact, for the case study presented in Section II-C, the accuracy drop was so significant that we would need to either make the model's topology much more complex or find a middle ground between CNNs and BNNs. We took the second route and came up with what we decided to call hybrid neural networks (HNNs), in which the binary quantization is only applied to the convolutional layers of the network (i.e., the feature extraction part), whereas the fully connected neurons (i.e., the classification part) remain at full precision. Then, as discussed in Section III, we compared a baseline CNN model versus a quantized HNN model in all of the experiments.

C. MNIST

The MNIST is a data set of 28×28 pixel images of handwritten decimal digits (from 0 to 9) [16]. The case study neural networks then receive 28×28 matrices as inputs and produce 10 outputs (one for each possible digit). As previously highlighted, both the baseline CNN and the quantized HNN use convolution to extract the specific features from the input image and then proceed to use a set of neurons to classify the input image based on such features. The CNN has an accuracy of about 93%, whereas the HNN has an accuracy of about 88%, and they both utilize the exact same topology (which is shown in Fig. 1).

- Input = 784 pixels.
- ConvLayer1 ($1\times 4 \times 4$ Filter).
- PoolLayer1 (3×3 Filter).
- ConvLayer2 ($3\times 4 \times 4$ Filters).
- PoolLayer2 (2×2 Filter).
- InnerProduct1 (48 Inputs, 20 Outputs).
- Rectified linear unit (ReLU1) (20 Inputs, 20 Outputs).
- InnerProduct2 (20 Inputs, 10 Outputs).
- Output = The image classification, which can be any integer from 0 to 9: *index(max(InnerProduct2.outputs))*.

III. Experimental Methodology

We implemented both the baseline MNIST CNN and the quantized MNIST HNN using the 16-nm FinFET Zynq UltraScale+MPSoC (ZU9EG) [17], which is composed of a processing system (PS), which uses a quad-core ARM A53, and a programmable logic (PL), based on a Xilinx FPGA. The details about the resource utilization by the two neural networks are shown in Table I. We can clearly see that the quantized version of the network uses significantly less LUTs. To be specific, the quantization allows for a 41% reduction in LUT utilization on the layers responsible for feature extraction. We can also notice that the utilization of FFs remains constant, as the pipeline stages in between the layers need to register the same amount of bits each cycle, regardless of having binary weights on the convolutional units.

It is relevant to highlight, before moving forward, that the training phases for both the networks were performed ahead of the experiments, in a fault-free environment. This means that the neural networks' training was not affected by the injected upsets or radiation in any capacity. The goal of this work is to evaluate how radiation can affect computation on a trained model with stabilized/nonchanging weights. Training in a radiation-rich environment is out of the scope for this article.

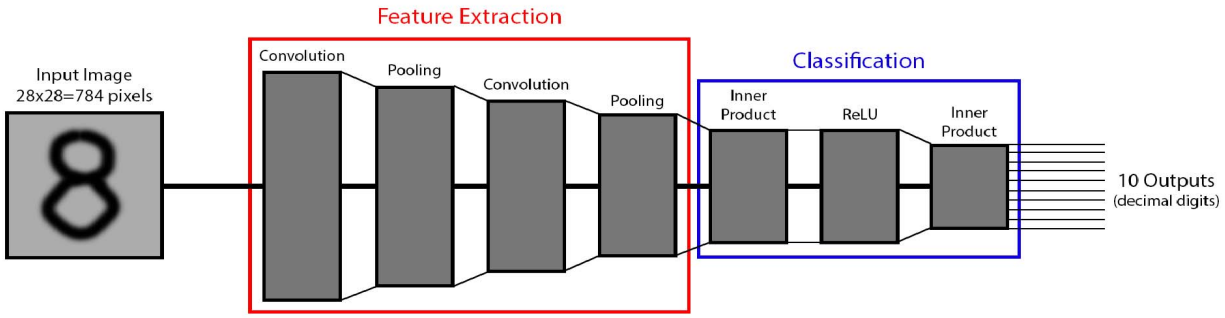

Fig. 1. Topology of both the baseline MNIST CNN and the quantized MNIST HNN.

TABLE I
ZYNQ ULTRASCALE+RESOURCE UTILIZATION TO IMPLEMENT THE BASELINE MNIST CNN AND THE QUANTIZED MNIST HNN

MNIST Design	CNN Portion	LUTs	FFs
Baseline CNN	Feature Extraction	165k	4.2k
	Classification	48k	0.4k
Quantized HNN	Feature Extraction	98k	4.2k
	Classification	48k	0.4k

A. Fault Injection Framework

In order to perform the fault injection campaigns, we take the advantage of the processor configuration access port (PCAP) of the MPSoC system and emulate SEUs by injecting faults in the FPGA's configuration memory, through software being executed on the ARM A53 processor. For both the baseline CNN and the quantized HNN, we isolate the feature extraction and classification portions of the networks into separate regions of the FPGA fabric, using *Pblocks*. Then, it is possible to correlate the location of an injected fault to one of the P blocks and, thus, to a specific portion of the network. We have conducted fault injection campaigns that randomly injected a total of about 11 million faults in the designs [out of around 39 million possible in an exhaustive campaign, which would be immensely time-consuming to conduct, given the size of the device under tests (DUTs)]. The random fault injection methods are well known and has been shown by the previous works to be very much representative [18].

The area of injection is strictly comprised of the configuration bits related to configurable logic blocks (CLBs) (LUTs, DSPs, FFs, and interconnections). During each iteration of the fault injection, the A53 microprocessor verifies whether or not the neural network execution produced the expected outputs and logs the results for further review. Finally, the data log that was produced during the experiment is analyzed, and the observed errors are classified as to be described in Section III-B.

B. Error Criticality and Benign Errors

When dealing with CNNs and HNNs, which essentially work as classifiers, not every error is to be considered critical. In other words, even if the corrupted output is different than what was expected, the classification of a given image might still be correct. Thus, we identify two possible error classes.

1) *Tolerable:* the network produces outputs different than the expected ones, but the classification is still correct.
2) *Critical:* the network produces output errors, and they are severe enough to compromise the image classification.

In the specific case of the MNIST case study, we are trying to identify the handwritten digits. A tolerable error would happen when one or more of the ten outputs differ from the expected/golden values, but the digit on the input image is still correctly identified. A critical error would happen when the computation is so significantly disturbed that it led to an "8" being classified as a "6," for instance.

In addition to that, we also identify the cases where we were expecting the fault-free model to wrongly classify an input image (due to lack of accuracy), but, due to an injected upset, it actually ends up classifying it correctly. As this occurrence is beneficial to the overall system reliability, we have decided to call it a **Benign Error**. In Section IV, we divide all of the results in either *benign*, *tolerable*, or *critical* errors.

IV. EXPERIMENTAL RESULTS

A. Fault Injection Results

Using the setup described in Section III-A, we have injected faults separately in the *feature extraction* and *classification* parts of the baseline MNIST CNN and the quantized MNIST HNN. Fig. 2 shows the results obtained through our fault injection experiment, expressed in terms of architectural vulnerability factor (AVF), which represents the percentage of faults that propagated all the ways to the output. Furthermore, we divide the total AVF in the three error categories discussed in Section III-B: *benign*, *tolerable*, or *critical*. The error bars are lower than 2% of the reported values in the worst case. As also explained in Section III-B, the benign errors are beneficial to neural networks. Although the impact of benign errors is almost negligible (when compared to the other error categories), it is an interesting finding that the percentage of benign errors in the quantized HNN is about three times higher than on the baseline CNN even though their accuracies are only 5% apart from each other. In addition to that, whenever the faults occur in the earlier stages of the neural networks

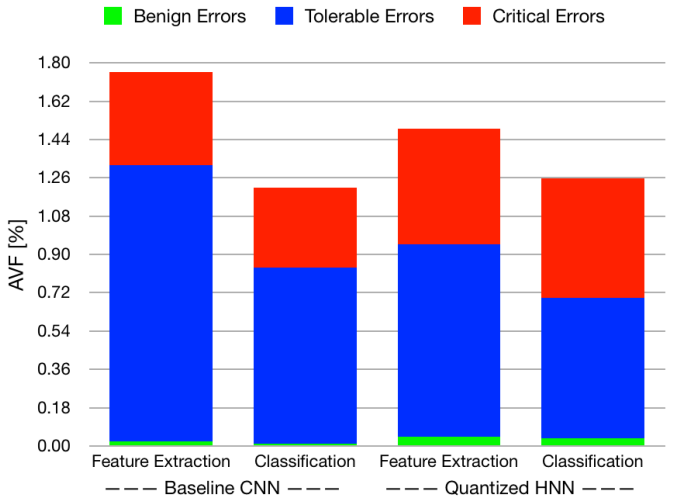

Fig. 2. AVF calculated from fault injection in the feature extraction and classification portions of the baseline CNN and the quantized HNN.

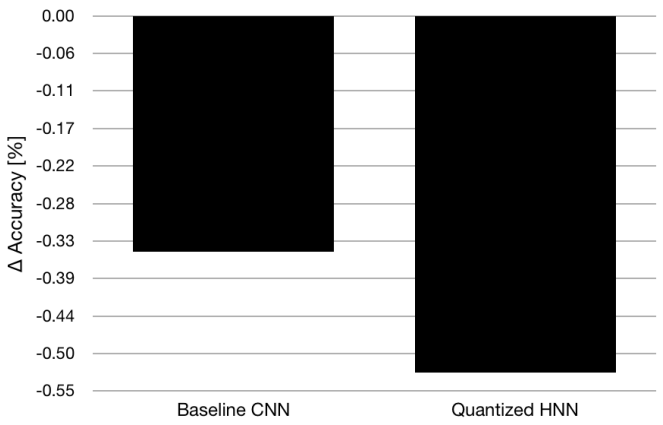

Fig. 3. Variation in accuracy for the baseline CNN and the quantized HNN when comparing their fault-free accuracies with their measured accuracies after extensive fault injection campaigns.

(in the feature extraction part), we are a bit more likely to see benign errors manifesting at the output.

In Fig. 2, one of the most easily perceived results is that, as expected, tolerable errors are the majority across all cases. If we analyze it further, we are able to see that a fault injected in the feature extraction portion of the quantized HNN has a 24% higher chance of generating a critical error at the output, when compared to a fault injected on the feature extraction portion of the baseline CNN. Similarly, a fault in the classification part of the quantized HNN has a 49% higher chance of provoking an erroneous image identification when compared to a fault in the classification part of the baseline CNN. If we combine both of these findings, we will ultimately come to the conclusion that, overall, the quantized HNN is less architecturally vulnerable than the baseline CNN, but, at the same time, it has a higher level of error criticality.

Another very interesting insight from the experimental results is that, in both cases, the total AVF of the feature extraction part is higher than on the classification part. This is because, as a fault occurs earlier in the network, it has more room to propagate across the computing units that follow ahead, increasing the odds of output corruption. Now, if we focus the attention just on the classification parts of the networks, we can see that the total AVF is slightly higher for the quantized HNN. We believe that there are two reasons as to why that is. First, the percentage of benign errors is three times lower than the baseline CNN. Second, the more nuanced argument is that, as a result of the quantization process in the convolutional layers, the weights in the classification part of the network end up carrying more significance in the design, which means that the neurons that form the fully connected layers become more sensitive to upsets than they normally would be, resulting in a higher AVF.

It is also important to mention that, for each injected fault, we ran the neural networks with 100 test images. In the experiments, we observed a little-to-no difference in error likelihood between the possible classes of the MNIST data set (decimal digits ranging from 0 to 9).

Furthermore, we have also used the fault injection experimental data to calculate the impact of upsets in the accuracy of each neural network. In order to do so, we simply calculated a variation (delta), from the starting fault-free accuracy to the final experimentally calculated accuracy (accounting for the observed critical and benign errors). The results are shown in Fig. 3. As the fault-free accuracy is obviously higher, the variation ends up being negative. What is clear to observe here is that, as the quantized HNN is generally more vulnerable, its accuracy drop was bigger than than the baseline CNN. This is because a higher level of error criticality means that a larger percentage of upsets is going to disturb the final classification computed by the neural network, ultimately leading to a lower accuracy when in a radiation environment.

B. Dynamic Cross Section Estimation

We have been able to estimate the dynamic cross sections based on the fault injection data, the resource utilization of the designs, and the per bit static neutron cross section of the configuration memory in the UltraScale+ (which is provided by Xilinx on their Device Reliability Report document as 2.67×10^{-16} cm^2) [19]. Since the AVF represents the probability of one bitflip to generate an observable error at the output, if we multiply it by the number of bits that a design utilizes and then by the per bit cross section of the configuration memory, we can roughly estimate how sensitive we expect the design to be. Such a method for dynamic cross section estimation has been shown to be very accurate by the previous works [20]. Thus, using the formula in (1), we calculate and plot the estimated cross sections in Fig. 4

$$\sigma = \text{AVF} \times (\#\text{EssentialBits}) \times \left(\frac{\sigma_{\text{static}}}{\#\text{CRAMBits}}\right). \quad (1)$$

In Fig. 4, we can clearly perceive that the quantized HNN is significantly less sensitive to radiation than the baseline CNN. To be precise, its estimated cross section is 39% lower. Meanwhile, we can also see that the percentage of errors that are considered critical to the network rises with quantization. While on the baseline CNN 26% of all observed errors is critical, 38% of data corruptions led to misclassifications

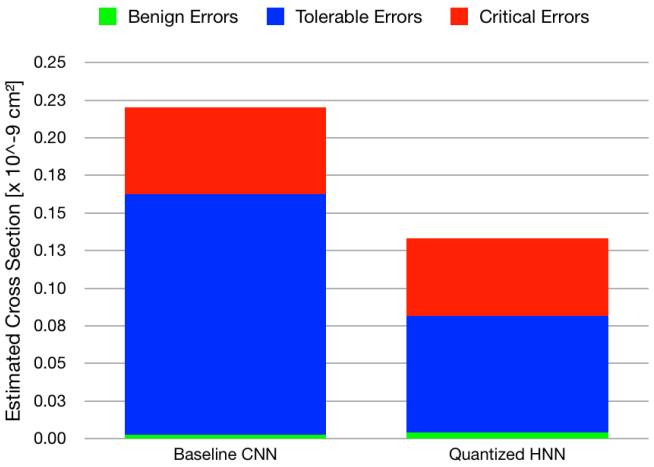

Fig. 4. Estimated neutrons cross section, calculated according to (1) for the baseline CNN and the quantized HNN designs.

on the binary quantized design in the experiments. This is because, as we quantize the weights on the convolutional layers of the network, two things happen: first, there is a slight decrease in the overall accuracy of the model (as discussed in Section II-C). Second, there is an increase in the relative importance of each weight to the computation, meaning that the design becomes more vulnerable to any given bitflip in the FPGA's configuration memory.

Beyond the cross section, we can also look at the mean executions between failures (MEBF) metric [21]. The MEBF can often be seen as a more complete indicative of a design's reliability and expected failure rate as it depends on both the circuit area and execution time. Before presenting further data, we should point out again that, as the case study is performing a classification task, we have translated the common *error* and *failure* definitions to *tolerable* and *critical*, respectively, (as per Section III-B). Furthermore, as neural networks are intrinsically approximate (meaning that they are not 100% accurate classifiers), we can also calculate the MEBF by taking into account such a lack of accuracy.

Based on all of the above considerations, we have decided to present a graph that considers both the sources of failures (critical errors): inaccuracy and radiation. From the inaccuracy standpoint, if a neural network is said to have an accuracy of a, it means that it will wrongly classify an input with a probability of *(1 - a)*. It follows that the MEBF (here, named MEBIF, where "I" stands for "inaccuracy") would be given by:

$$\text{MEBIF}(a) = \frac{a}{1-a}. \tag{2}$$

It is worth noticing that (2) does not depend on the cross section, particle flux, or execution time of the circuit. Even though we are referring to "circuit" in the scope of this work, this analysis would also apply for devices other than FPGAs.

From the radiation point of view, all of these variables become relevant. If we multiply the cross section by a given particle flux (e.g., the neutron flux at sea level), we get the failure-in-time (FIT) metric, which, as the name suggests, indicates how many failures we expect to observe in a time interval. Thus, the inverse of the FIT becomes the mean time to failure (MTTF), which is quite self-explanatory. Finally, dividing the MTTF by the execution time, we get the MEBF. Equation (3) expresses the MEBF (here, named MEBRF, where "R" stands for "radiation") as a function of *"c": cross section, "f": particle flux, "e": execution time*

$$\text{MEBRF}(c, f, e) = \frac{1}{c \cdot f \cdot e}. \tag{3}$$

From (3), we can see the following:

1) If the cross section is zero (meaning that the design is rad-hard), MEBRF goes to infinity (meaning that we will never see a radiation-induced failure).
2) If the particle flux is zero (meaning that we are in a radiation-free environment), MEBRF also goes to infinity.
3) If the execution time tends to zero, MEBRF goes to infinity again (meaning that there is only an infinitesimally small time interval, in which an impinging particle could affect the circuit).

Finally, we need to come up with an equation that, somehow, combines (2) and (3). It is quite obvious that a critical error can only be induced by radiation when the expected (fault-free) execution originally led to a correct classification. With that in mind, if we solve (4) for x, we will get a number in the interval [0,1] that symbolizes how the accuracy of the network is affected (attenuated) in the presence of radiation (noise)

$$\text{MEBRF}(c, f, e) = \frac{x}{1-x}. \tag{4}$$

Note that (4) looks very similar to (2). This is because, in a way of looking at it, we are calculating the "accuracy of the accuracy" of the neural network or the "radiation-induced attenuation factor" on the original accuracy. Further, note that the higher the MEBRF, the closer x gets to 1 (meaning the less attenuation there is). Solving (4) for x gives us

$$x(c, f, e) = \frac{\text{MEBRF}(c, f, e)}{1 + \text{MEBRF}(c, f, e)} \tag{5}$$

$$x(c, f, e) = \frac{1}{1 + (c \cdot f \cdot e)}. \tag{6}$$

The very last step is to calculate the overall MEBF as a function of *"a": accuracy, "c": cross section, "f": particle flux, "e": execution time*

$$\text{MEBF}(a, c, f, e) = \frac{a \cdot x(c, f, e)}{1 - a \cdot x(c, f, e)} \tag{7}$$

$$\text{MEBF}(a, c, f, e) = \frac{a}{1 + c \cdot f \cdot e - a}. \tag{8}$$

Note that for either c, f, or e equal to zero,(8) trivially becomes(2) (if there are no radiation effects, the only source of errors is inaccuracy). Similarly, for $a = 1$,(8) reduces to(3) (if the network is fully accurate, the only source of errors is radiation). These relationships between(2),(3), and(8) prove the consistency in the findings.

Finally, by looking at (8), we arrive to the following, very straightforward, conclusions regarding how one would be able to reduce the failure rate of neural networks.

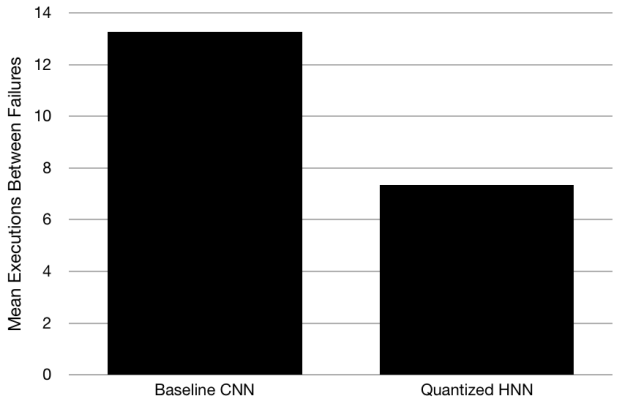

Fig. 5. MEBF of the baseline CNN and the quantized HNN, considering both inaccuracy and radiation effects.

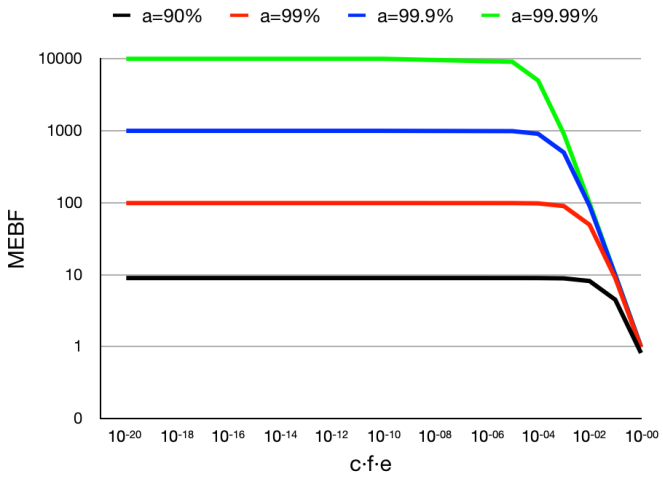

Fig. 6. MEBF as a function of inaccuracy and radiation.

1) Increase the accuracy.
2) Make it rad-hard.
3) Get away from radiation.
4) Reduce the execution time.

Considering the case studies, at sea level, the *cfe* product becomes very small, with an order of magnitude of 10^{-20}. This is mainly because the neutron flux at sea level is low ($13n/(\text{cm}^2 \times h)$), and the execution time of the designs is very low too (233 ns).

Fig. 5 shows the overall MEBF of the neural networks, calculated using (8). We can immediately see that, despite having a 39% lower cross section, the binary quantized network only executes, on average, seven to eight times before it experiences a critical error, whereas the baseline version executes about 13 to 14 times before a failure. This indicates that, at this point in time, a lack of accuracy seems to be a much bigger problem for neural networks than radiation is. In fact, we can estimate, for each level of accuracy, when does radiation become a problem. Fig. 6 shows how the MEBF varies as a function of the *cfe* product for different values of accuracy.

We can clearly see that the curves in Fig. 6 stay flat for the most of the parts. However, precisely at the point when they start to drop, is where we should start worrying about the radiation effects. The graph shows that, if we have a 99.99% accurate model, such a tipping point is at $cfe = 10^{-5}$. Similarly, a model with 99.9% accuracy should start facing significant radiation-induced problems at $cfe = 10^{-4}$. As the red and black curves follow a similar pattern, we can draw a generic conclusion: as we give up one *9* of accuracy, we postpone the radiation problem by $10\times$.

V. Conclusion

We have seen that not all errors need to be considered critical in neural networks. Some of them can be classified as tolerable, and in rare cases, errors can also contribute to the overall accuracy of a given model. Furthermore, we have mentioned that, in the case study, the binary quantization of weights in the convolutional layers led to generally lower vulnerability factors, but increased error criticality, and a more volatile accuracy in the presence of upsets on the FPGA's configuration memory.

Besides, using the fault injection data, combined with experimental static cross section measurements from Xilinx, we were able to estimate the dynamic cross sections of the case studies. We have seen that quantizing the weights of the convolutional layers led to a 39% lower radiation sensitivity. On the other hand, we have also seen that the percentage of errors to be considered critical in a classification task increased by 12% on the quantized design.

Beyond that and perhaps the most significant contribution in this article, we have acknowledged that a neural network can fail in two ways: by being inherently inaccurate or by being vulnerable to radiation. From this idea, we have derived expressions that model the failure rates in any given neural network. First, we have explained how inaccuracy leads to failures [see (2)]. Second, we have explained how radiation leads to failures [see(3)]. Finally, we have combined these two expressions in an equation that provides the global perspective [see (8)]. It is important to highlight here that (2) comes from the object-detection community, and (3) was introduced by [21]. Equation (8), however, is the own.

Using the newly created equation, we were able not only to determine the overall failure rate of the case studies but also to extend the idea to a generic case. Fig. 6 shows a good job of explaining, in a very simple manner, the effects of inaccuracy and radiation sensitivity on neural networks. On top of that, it also shows, for each level of accuracy, the precise point where the radiation effects become considerably relevant.

As the future work, we first intend to explore different case studies. As explained in Section II, the MNIST data set is a very simple one and as such does not require a very large and complex CNN topology. We believe that, when testing state-of-the-art networks, the execution time is going to increase by quite a few orders of magnitude, which, as previously stated, will translate to a bigger impact on the overall MEBF of the design. We also want to test the designs with different types of particles. This is because it would be very interesting to see how the MEBF is affected when the cross section and the particle flux are also the orders of magnitude different from what we observe in neutron experiments/environments.

Finally, we believe that the contributions made on this article will serve the community as a whole, considering the fact that, even though we are specifically dealing with an FPGA, the equations hereby presented are generic enough to be applied to any device that runs neural networks (e.g., GPUs, CPUs, and ASICs).

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. 7–15.

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

[4] V. M. Polyakov, I. N. Kaliteevsky, K. S. Amelin, V. A. Smyslov, and M. A. Permyakov, "Complexed NIR laser detector and LWIR camera optical system with neural network management for UAV collision avoidance system," in *Proc. Int. Conf. Laser Opt. (ICLO)*, Jun. 2018, p. 280.

[5] 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.

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

[7] X. Chen, G. Liu, J. Shi, J. Xu, and B. Xu, "Distilled binary neural network for monaural speech separation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 3350–3357.

[8] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. 55th Annu. Design Autom. Conf. (DAC)*, New York, NY, USA, 2018, pp. 19:1–19:6, 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. (SC)*, New York, NY, USA, 2017, pp. 8:1–8:12, doi: 10.1145/3126908.3126964.

[10] F. Fernandes dos 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.

[11] B. Du, S. Azimi, C. de Sio, L. Bozzoli, and L. Sterpone, "On the reliability of convolutional neural network implementation on SRAM-based FPGA," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2019, pp. 143–148.

[12] F. Benevenuti, F. Libano, V. Pouget, F. L. Kastensmidt, and P. Rech, "Comparative analysis of inference errors in a neural network implemented in SRAM-based FPGA induced by neutron irradiation and fault injection methods," in *Proc. 31st Symp. Integr. Circuits Syst. Design (SBCCI)*, Aug. 2018, pp. 164–169.

[13] F. Libano *et al.*, "Selective hardening for neural networks in FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 1, pp. 216–222, Jan. 2019.

[14] J.-K. Kim, M.-Y. Lee, J.-Y. Kim, B.-J. Kim, and J.-H. Lee, "An efficient pruning and weight sharing method for neural network," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Oct. 2016, pp. 49–50.

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 6869–6898, Jan. 2017.

[16] 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.

[17] Xilinx. *Zynq UltraScale+ MPSoC*. Accessed: Mar. 23, 2020. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

[18] F. Benevenuti and F. L. Kastensmidt, "Comparing exhaustive and random fault injection methods for configuration memory on SRAM-based FPGAs," in *Proc. IEEE Latin Amer. Test Symp. (LATS)*, Mar. 2019, pp. 111–116.

[19] Xilinx. *Device Reliability Report*. Accessed: Mar. 23, 2020.[Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug116.pdf

[20] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 2147–2157, Dec. 2003.

[21] P. Rech, L. L. Pilla, P. O. A. Navaux, and L. Carro, "Impact of GPUs parallelism management on safety-critical and HPC applications reliability," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 455–466.