How Reduced Data Precision and Degree of Parallelism Impact the Reliability of Convolutional Neural Networks on FPGAs

F. Libano, P. Rech, *Senior Member, IEEE*, B. Neuman, J. Leavitt, M. Wirthlin, *Senior Member, IEEE*, and J. Brunhaver

***Abstract*— Convolutional neural networks (CNNs) are becoming attractive alternatives to traditional image-processing algorithms in self-driving vehicles for automotive, military, and aerospace applications. The high computational demand of state-of-the-art CNN architectures requires the use of hardware acceleration on parallel devices. Field-programmable gate arrays (FPGAs) offer a great level of design flexibility, low power consumption, and are relatively low cost, which make them very good candidates for efficiently accelerating neural networks. Unfortunately, the configuration memories of SRAM-based FPGAs are sensitive to radiation-induced errors, which can compromise the circuit implemented on the programmable fabric and the overall reliability of the system. Through neutron beam experiments, we evaluate how lossless quantization processes and subsequent data precision reduction impact the area, performance, radiation sensitivity, and failure rate of neural networks on FPGAs. Our results show that an 8-bit integer design can deliver over six times more fault-free executions than a 32-bit floating-point implementation. Moreover, we discuss the tradeoffs associated with varying degrees of parallelism in a neural network accelerator. We show that, although increased parallelism increases radiation sensitivity, the performance gains generally outweigh it in terms of global failure rate.**

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

I. Introduction

IN RECENT years, the trillion dollar automotive industry has been very much focused on progressively adding technology to vehicles and on enabling self-driving capabilities [1]. Through the pairing of sensorial data collection with high-performance image processing, embedded computing systems used by companies like Mercedes-Benz [2], Volvo [3], and Tesla [4], are currently able to deliver semi-autonomous driving experiences. In fact, the push for autonomous vehicles is also very much true in military organizations (with the increased adoption of unmanned aerial vehicles (UAVs) [5]) and in the space exploration sector (with the use of rovers and helicopters on Mars [6], [7]). At the same time, convolutional neural networks (CNNs) have significantly evolved in terms of accuracy and became very attractive solutions for image processing and pattern recognition workloads [8]. Given that all of the aforementioned safety-critical applications depend on reliable computer vision [9]–[11], CNNs emerge as great alternatives to more old-fashioned algorithms.

Neural networks have a parallel structure, both in terms of neurons, and convolutional filters. Such inherent parallelism makes CNNs perfect candidates for efficient acceleration on parallel computing devices, such as application-specific integrated circuits (ASICs), graphics-processing units (GPUs), and field-programmable gate arrays (FPGAs). In a perfect world, we would be able to define an optimal hardware architecture, fabricate an ASIC, and never look back. However, the field of machine learning is ever evolving, which means that the requirements for accelerating the neural network topologies of today might not be the same a few years in the future. Therefore, design flexibility and reprogrammability must be considered. GPUs currently offer parallel general-purpose arithmetic hardware and design flexibility at the software level. However, most state-of-the-art GPU architectures are throughput-oriented [12] (rather than latency-oriented) and not power-efficient [13]. FPGAs, on the other hand, offer full hardware reprogrammability and lower power consumption [14], which enables system patches (for architectural improvements, weight tuning, bug fixing, and more), along with the ability for deployment on power constrained missions (which is often the case in space applications) [15].

Unfortunately, FPGAs are very sensitive to radiation-induced faults [16]. More specifically, SRAM-based FPGAs can experience single-event upsets (SEUs) in their configuration memory, which affects routing connections as well as modifies the settings and content of LUTs, DSPs, BRAMs, and FFs. With a corrupted configuration memory, the design on the FPGA can start malfunctioning

Manuscript received November 20, 2020; revised December 27, 2020; accepted January 7, 2021. Date of publication January 11, 2021; date of current version May 20, 2021. 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).

P. Rech is with the Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre 91501-970, Brazil (e-mail: prech@inf.ufrgs.br).

B. Neuman is with the Los Alamos National Laboratory (LANL), Los Alamos, NM 87544 USA (e-mail: bneuman@lanl.gov).

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

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNS.2021.3050707.

Digital Object Identifier 10.1109/TNS.2021.3050707

and providing erroneous answers. Furthermore, each type of logic resource available on the FPGA corresponds to a different percentage of the total configuration bits [17] and produces a different type of error/failure in the presence of SEUs. Therefore, the specific mapping of the circuit (i.e., the neural network accelerator) is an important component of overall resilience. More broadly speaking, it follows that reliability must always be taken into account when developing FPGA-based computational solutions for safety-critical applications.

Previous works have discussed the general reliability of CNNs executing in ASICs [18], GPUs [19], [20], and FPGAs [21], [22]. Other studies have considered the tradeoffs in performance and radiation sensitivity of using reduced and mixed precision in different computer architectures, while executing well-known benchmarks (including state-of-the-art neural networks for object detection) [23], [24]. Furthermore, the combined impact of accuracy (or lack thereof) and radiation on the overall failure rate of neural networks was discussed and modeled [25]. This article focuses on the reliability benefits of reduced precision in CNNs, whenever paired with proper quantization techniques (that allow for zero loss of accuracy), which is, to the best of our knowledge, a novel evaluation for FPGAs. We also evaluate architectural variations with majorly different degrees of parallelism and discuss the tradeoffs between area, speed, and reliability.

In order to deliver high accuracy, CNNs require high computational complexity. As an example, the number of arithmetic operations required for processing each input image in Google's state-of-the-art InceptionV3 architecture is over 5 billion [26]. On FPGAs, the need for higher computing power often translates to higher resource utilization, which in turn increases device cost, energy consumption, and sensitive area (i.e., radiation susceptibility). In order to speed-up neural networks, designers have explored reducing the precision of data representation throughout the network. As a standard, state-of-the-art frameworks such as Google's TensorFlow use 32-bit floating point at training time, but recently developed quantization techniques have allowed the deployment of 16-bit floating-point and even 8-bit integer versions of same network, with little to no accuracy loss [27]. The first contribution of this article is an evaluation on how such data precision reduction impacts the overall reliability of CNNs. Our experimental data show that it is possible to decrease the failure rate of a neural network by over 70%, simply using proper quantization methods and a corresponding 8-bit integer representation.

Another important design decision for neural network hardware accelerators is the degree of parallelism, as the architect must define how many processing elements (PEs) are going to be instantiated in the programmable logic of the FPGA. Strictly from the performance standpoint, it seems that the answer would be "*as many as possible*." However, parallelism increases resource utilization, which in turn leads to increased radiation sensitivity, and, possibly, lower reliability. In addition, given the sizes of the current state-of-the-art neural network topologies, it is most often infeasible to implement *ALL* of the necessary PEs (providing maximum parallelism), as the logic resources on FPGA devices are limited. Therefore, as the second contribution of this article, we analyze and discuss the area, performance, and reliability tradeoffs associated with varying degrees of parallelism for neural network accelerators. We show that radiation sensitivity can vary as much as two orders of magnitude, when comparing the two ends of the parallelism spectrum (i.e., minimum number of PEs versus maximum number of PEs), while the estimated failure rate is over $7\times$ lower in the faster, more parallel design.

The remainder of this article is organized as follows. Section II gives a more comprehensive background on neural networks, the available quantization tools, and the specifics of our case study CNN. Section III discusses the details of our FPGA designs, methodology, and radiation beam experiment. Section IV presents our results regarding data precision reduction and degree of parallelism. Section V goes over our main conclusions as well as intentions for future work.

II. Background

A. Convolutional Neural Networks

Artificial neural networks (ANNs) are biologically inspired computational structures that are able to calculate their outputs by propagating data through sets of interconnected neurons, distributed across different layers. From a mathematical perspective, the output of a neural network is generated by a series of matrix multiplications, which, in turn, are calculated through a series of multiply accumulated operations. A CNN is a special kind of ANN, mostly dedicated to image-processing tasks. The first few layers in a CNN are responsible for extracting features from the input image (such as edges and shadows) before feeding the layers of fully connected neurons (responsible for classification and decision-making). By using convolution and pooling operations, it is possible to reduce the amount of data to be processed by the neurons, ultimately making the computation more efficient. All filters and neurons have weights associated with them, which are learned during an extensive training process. Input data is iteratively presented to the model, and minor adjustments are made after each step. Once the training is complete, the network is able to compute solutions for novel inputs, based on its learned set of weights.

B. Quantization of Neural Networks

In order to achieve high levels of accuracy, CNNs end up being very much computationally expensive. As safety-critical applications usually require low latency systems and algorithms, a number of simplification techniques have been developed for neural networks, such as weight trimming and quantization [28]. In this article, we focus on the latter. The main idea behind quantization is to reduce the precision in which the weights of a given model are represented, in order to speed-up computation and reduce resource utilization. At training time, all industry-leading frameworks use 32-bit floating point as a default. For the specific case of TensorFlow, they provide a subset of tools called TensorFlow Lite [29], which allows developers to quantize their models to lower precisions, such as 16-bit floating point (IEEE's half-precision) and 8-bit integer. Previous works have shown that careless

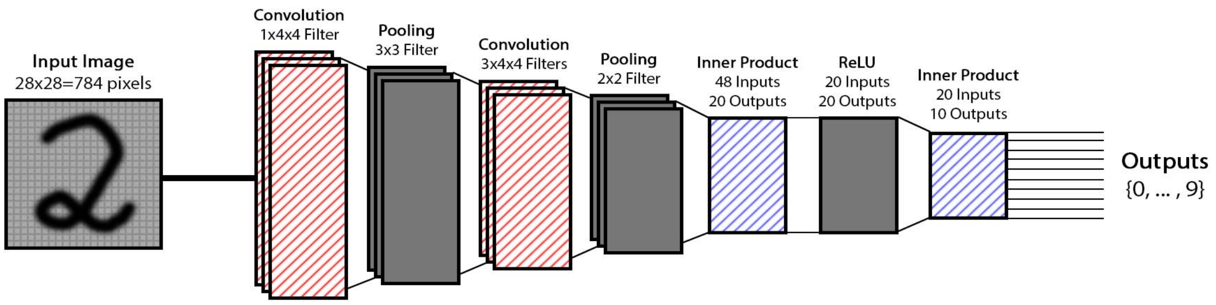

Fig. 1. Topology of the MNIST CNN. Striped layers have weights associated with them (which go through the quantization process).

quantization procedures can often compromise accuracy (point at which the efforts in reducing radiation-induced errors cease to pay off) [25]. The clever thing about TensorFlow's tools is that their quantization process uses a small subset of the training data for calibration purposes. The end result is that the quantized models show little to no accuracy loss when compared to their 32-bit float counterpart.

C. Related Work

As stated before, this article is not the first to evaluate the reliability of CNNs. Prior work has studied neural network resilience on ASICs, proposing and validating a framework for pre-silicon fault tolerance estimation, mostly based on topological characteristics [18]. Furthermore, GPU-focused papers have experimentally evaluated the reliability of state-of-the-art neural networks for object detection (with an emphasis on automotive applications), executing on different NVIDIA architectures. The authors have also tested an algorithm-based fault-tolerance (ABFT) technique for matrix multiplication, achieving significant error detection/correction rates [19], [20]. Specifically on FPGAs, prior work has discussed the concept of error criticality and used extensive fault injection campaigns to identify differences in vulnerability across the layers in ANNs and CNNs. Additionally, a low-overhead selective hardening strategy was proposed, for scenarios in which traditional triple modular redundancy (TMR) is not possible due to limited resource availability [30]. Moreover, the correlation between faults in different logic resources of the FPGA and corresponding output errors was studied [31].

Finally, we must acknowledge that the reliability impact of reduced data precision on neural networks has also been explored before [24]. In such study, the authors have evaluated the different failure in time (FIT) rates of a CNN on a GPU, using double, single, and half-precision floating point. They have found that lower precision computation in GPUs reduces radiation sensitivity, while improving performance. Such finding further motivates our work, which intends to guide the design of reliable hardware accelerators for neural networks on FPGAs (by looking at precision and parallelism).

D. MNIST

We have chosen the well-known Modified National Institute of Standards and Technology (MNIST) as our case study data set, since its simplicity translates to tractability for implementing and testing several design variations. We should, however, acknowledge that this aspect of our work limits the extent to which our experimental data can be fully generalized, as more complex CNNs could have different behaviors. The data set itself is composed by 60 000 28×28 pixel images of handwritten decimal digits (from 0 to 9) [32]. As such, our neural network models receive 28×28 matrices as inputs and produce ten outputs (one for each decimal digit), where the index of the highest value corresponds to the classification.

III. EXPERIMENTAL METHODOLOGY

A. Designs Under Test

In order to evaluate the tradeoffs associated with data precision reduction, we have implemented three versions of the MNIST CNN (FP32, FP16, INT8) using the 28-nm Zynq-7000 (XC7Z020) [33]. Table I and Fig. 2 detail resource utilization and execution times. In addition to that, we should point out that the accuracy on all of the design variations has stayed the same (95%), regardless of the quantization process performed with TensorFlow Lite. Although a higher accuracy level could have been achieved, we opted for a very minimalist CNN topology (and implementation), as detailed in Fig. 1.

In order to evaluate the tradeoffs associated with degree of parallelism, we have implemented two versions of the MNIST CNN (Min PEs, Max PEs) using the 16-nm Zynq UltraScale+ (XCZU9EG) [34]. The first design (Min PEs) has only one PE per layer of the network and represents the lower end of the parallelism spectrum. Likewise, the second design (Max PEs) has all of the necessary PEs in each layer of the network and represents the top end of the parallelism spectrum. In other words, if there is a total of N operations involved in a layer's computation, the single PE version is going to take N iterations to finish, while the fully parallel implementation (in this case. with N PEs), will take only one clock cycle. The aforementioned simplicity/tractability of the MNIST CNN, along with the use of a high-end FPGA (as the one found on the UltraScale+), allow it to be implemented in a fully parallel fashion. Table II and Fig. 3 provide details on resource utilization and execution times. Note that neither of the designs utilize DSPs. This is because both use 8-bit precision, which is too small for DSP inference at synthesis time. Also note that the axes in Fig. 3 are in logarithmic scale.

TABLE I

ZYNQ-7000 RESOURCE UTILIZATION TO IMPLEMENT THE MNIST CNN USING 32-bit FLOATING POINT, 16-bit FLOATING POINT, AND 8-bit INTEGER

MNIST Design	LUTs (Logic)	LUTs (Mem)	FFs	DSPs
32-bit Float (FP32)	6.5k	1016	336	8
16-bit Float (FP16)	3.4k	510	240	4
8-bit Integer (INT8)	1.5k	280	224	0

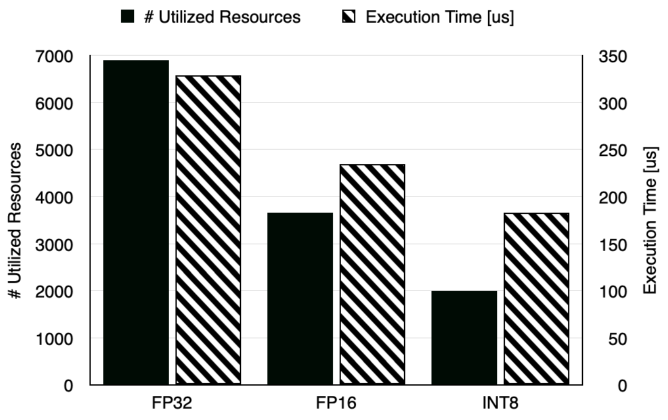

Fig. 2. Total resource utilization and execution times for three levels of data precision on the MNIST CNN, implemented on the Zynq-7000.

We should point out, before moving forward, that the training phases and quantization processes for all of our neural networks were fully completed ahead of the experiments (in a fault-free environment). This means that the neural networks' set of weights was not affected by radiation-induced upsets in any capacity. Furthermore, we must also mention that the input stimuli for our designs under tests (DUTs) is a subset of 100 images from the MNIST data set (all of which are correctly classified in undisturbed executions).

B. Radiation Experimental Setup

Our radiation beam experiments were performed at the Los Alamos Neutron Science Center (LANSCE) facility of the Los Alamos National Laboratory (LANL). While LANSCE's neutron spectrum mimics the atmospheric one, the particle flux is about 8 orders of magnitude higher than the average terrestrial flux [(13 n/(cm^2 × h)] at sea level [35]). We ran our experiments for around 64 h, accumulating a total fluence in our DUTs of 344 × 10^9 n/cm^2, roughly equivalent to 3 million years of natural exposure. The experimental setup, with the Zynq-7000 and the Zynq UltraScale+ is shown in Fig. 4, mounted at the beam line. We specifically chose Zynq devices because their heterogeneous nature considerably simplifies the setup. Using the ARM processors, we are able to provide stimuli to the FPGA and implement result checking. Since the DDR is protected with ECC, input/output corruption becomes extremely unlikely. Furthermore, given the trivial nature of the C program, we very rarely observe hangs during execution (in which case, it suffices to power cycle the device). A server PC controls/logs the experiment from outside the beam room. We are also able to distinguish between transient and permanent upsets: Whenever an output error is observed,

TABLE II

ZYNQ ULTRASCALE+ RESOURCE UTILIZATION TO IMPLEMENT THE MNIST CNN WITH VASTLY DIFFERENT DEGREES OF PARALLELISM

MNIST Design	LUTs (Logic)	LUTs (Mem)	FFs	DSPs
Min PEs	1.4k	280	240	0
Max PEs	212k	0	11.2k	0

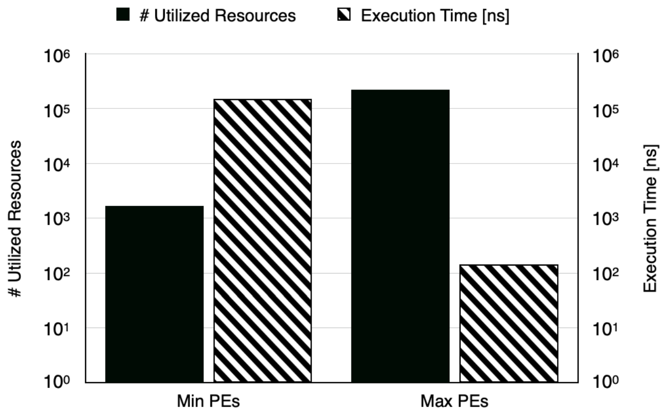

Fig. 3. Total resource utilization and execution times for two degrees of parallelism on the MNIST CNN, implemented on the Zynq UltraScale+.

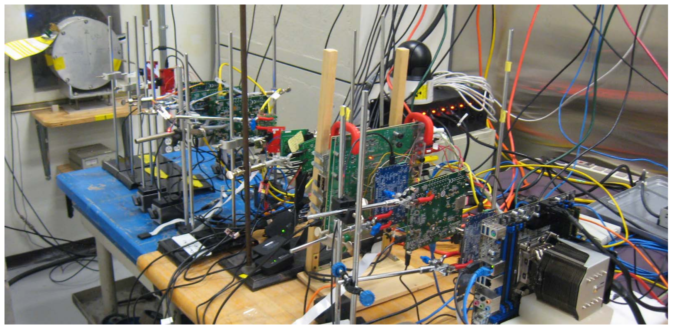

Fig. 4. Neutron beam experiment at the LANSCE facility of LANL, USA.

we run the same image again. If the second execution also presents corruptions, the upset is permanent, otherwise it is a single-event transient (SET). However, we have not registered any of such transients in our neutron beam experiments.

C. Error Criticality

In most cases, CNNs are used for classification tasks, which means that not all errors need to be considered critical. In other words, even if the outputs computed by the network are different than the golden/expected ones, the classification of a given input image might still be correct. Hence, we identify two error classes in our neural networks:

1) *Tolerable error:* The network's outputs differ from the expected ones, but the image classification still comes out correct (i.e., the image classification is correct despite the output corruptions).
2) *Critical error:* The network's outputs differ from the expected ones, being severe enough to compromise the image classification (i.e., an input image of the digit "7" is classified as the digit "1").

IV. Experimental Results

A. Reduced Data Precision

Using the experimental setup and methodology described in Section III, we have tested three versions of the MNIST CNN with varying levels of data precision (FP32, FP16, INT8). Fig. 5 plots the neutrons cross section of the three designs. We classify output errors as tolerable or critical, depending on whether or not the image classification was affected (as explained in Section III-C).

As we have shown in Table I and Fig 2, lower precision hardware means lower resource utilization, which consequently means that the sensitive area of the FPGA for each design also gets reduced. Specifically, the 16-bit floating-point implementation uses about 40% less resources than the 32-bit version, while the 8-bit integer design decreases resource utilization by 64%. By analyzing Fig. 5, we can clearly see that, as we reduce precision, the probability of observing radiation-induced data corruptions diminishes. To be exact, the 16-bit floating-point version of the network has a 22% lower chance of producing errors at the output, when compared to the original 32-bit design. Similarly, when using 8-bit integers, we see a massive 72% cross section reduction from FP32.

A more nuanced result that can be drawn from our experimental data is that the rate of critical errors (as defined in Section III-C) is very different across the three designs. If we think about the way floating-point numbers are represented in computers (sign, exponent, and mantissa), it is easy to conclude that radiation-induced corruptions in the sign and in the exponent of a number can cause a much greater discrepancy. As the IEEE 754 standard establishes, the 32-bit floating-point representation uses 1 bit for the sign, 8 bits for the exponent, and 23 for the mantissa, while the 16-bit floating-point representation uses 1, 5, and 10, respectively. We can say that 28.13% (9/32) of the bits in an FP32 word have a high potential of causing significant discrepancies between expected and computed outputs, while 37.50% (6/16) of the bits in an FP16 word could lead to considerable differences. In our CNN, this ultimately means that, as we reduce precision from FP32 to FP16, the likelihood of an output error affecting the final classification of the input image increases. This result can even be intuitively generalized for a hypothetical FP8 representation, which would very likely have a critical error rate higher than FP16. However, this pattern does not continue when we further reduce precision to 8-bit integers. This is because there are no exponent bits in an integer representation, so when a given bit *n* gets corrupted, the difference between expected and computed value can only be $\pm 2^n$, as opposed to a $x2^n$ discrepancy in an exponent corruption of a floating-point number. In our experiments, we have found that only 7% of the errors were critical in the FP32 design, while the error criticality rose to nearly 18% on the FP16 design and then fell back down to about 15% on the INT8 version of the MNIST CNN. Regardless, 8-bit integer representation had the lowest absolute critical error rate out of all three implementations.

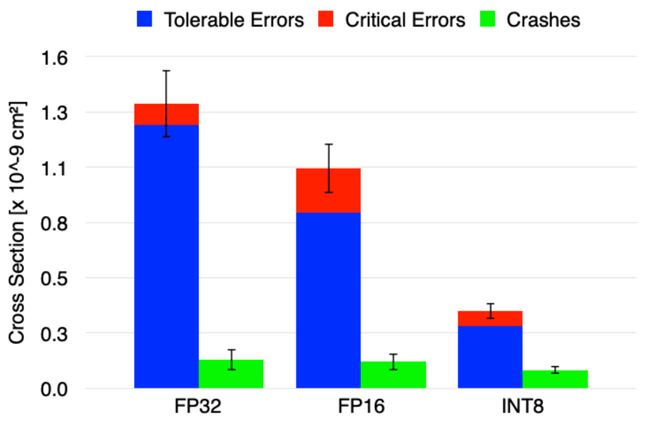

Fig. 5. Neutron cross section for the FP32, FP16, and INT8 versions of the MNIST CNN.

Additionally, we have also observed instances where the CNN implemented on the FPGA did not provide any outputs after its expected execution time. This happens whenever the finite state machine (FSM) responsible for the control logic of the hardware gets stuck, failing to reach its "*done*" state. Such instances are shown in Fig. 5 as *Crashes*. As this type of event is much rarer than silent data corruptions (SDCs) (only accounts for 10%–20% of the total cross section), the error bars do not allow us to draw any conclusions from the experimental data. But, from an architectural perspective, we intuitively know that reducing the data precision only reduces the area occupied by the arithmetic pipeline and not the area occupied by the FSM, which means that the likelihood of observing crashes should be roughly the same across all three versions of the MNIST CNN.

The impact of reducing data precision can further be explored as we make use the notion of tolerated relative error (TRE) [24]. Fig. 6 basically shows how the neutron cross sections would be reduced if we were to allow a certain percentage of tolerance for discrepancies between expected and computed outputs. With a TRE of 0%, any bitflip in the output is considered an error, but as we increase the TRE, we start to establish intervals of tolerance, instead of Boolean decisions. For example, with a TRE of 1%, any output corruption from 99% to 101% of the expected/golden value would still be considered correct. Interestingly, with a TRE level of only 1%, the cross section of tolerable errors on the 32-bit floating-point version of the CNN would be reduced by 43%. This is because, as we previously discussed, 23 out of 32 bits are reserved for the mantissa, which means that most of the radiation-induced corruptions will not have a very significant impact on an FP32 word. Extending the comparison, if we were to use the same 1% tolerance interval, the error rate on the FP16 implementation would only improve by 8%, while the 8-bit integer version would not change at all. It can then be said that, by treating neural networks as the inherently approximate computing units that they are, the perceived/effective radiation sensitivity can be significantly reduced. The caveat is that TRE only helps reducing the tolerable portion of the cross section, as the image classification

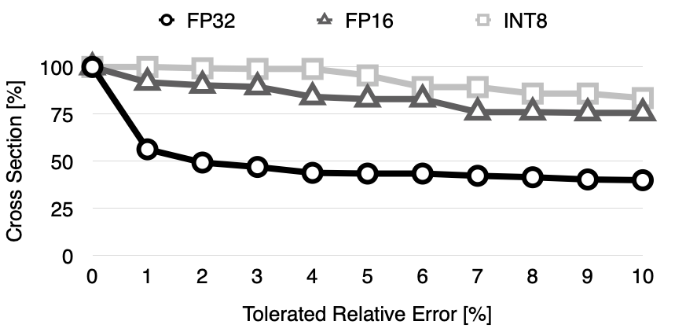

Fig. 6. Reduction in the cross sections of the FP32, FP16, and INT8 versions of the MNIST CNN, when establishing incremental tolerance intervals.

TABLE III
MEBRFs FOR THE FP32, FP16, AND INT8 VERSIONS OF THE MNIST CNN

	FP32	FP16	INT8
MEBRF [x 10^{15}]	0.61	1.11	4.08

depends on the comparison between all ten outputs, as opposed to their relationship with gold.

In order to get a realistic estimate of the failure rate of an algorithm in a radioactive environment, the mean executions between failures (MEBF) metric is commonly utilized [36], given that it takes into account both radiation sensitivity as well as performance. However, neural networks fall into a special category of applications, since they have an associated level of accuracy to them. Previous works have modeled the overall failure rate of neural networks (considering both inaccuracy and radiation as sources of faults) [25], as they analyzed binary quantization, which is rarely lossless. Since our CNN topology is simple, and our precision reduction was not so extreme, accuracy was maintained across all DUTs at 95%. Thus, we focus our analysis strictly on the effects of radiation, using the mean executions between radiation failures (MEBRF) metric. We show the MEBRF of our designs in Table III, considering the neutron flux at sea level [35] and the entire SDC cross section measurements (tolerable+critical errors), as legislation typically does not distinguish between error categories [37]. Evidently, the INT8 version of the MNIST CNN is able to complete a much higher number of failure-free executions than the floating-point implementations. To be exact, FP32 experiences over six times the failure rate of INT8, while FP16 fails over three times as much as the integer-based design. This is because reduced precision hardware not only occupies less area, but is also faster. The end result is lower radiation sensitivity and higher reliability, a true win–win situation (provided that accuracy remains stable).

B. Degree of Parallelism

Using the experimental setup and methodology described in Section III, we have tested two versions of the MNIST CNN with varying degrees of parallelism (Min PEs, Max PEs). As previously mentioned, the two designs in this analysis represent the two ends of the parallelism spectrum. On the one hand, we have a very small, iterative design, which uses few resources (therefore, occupies less area), and takes longer

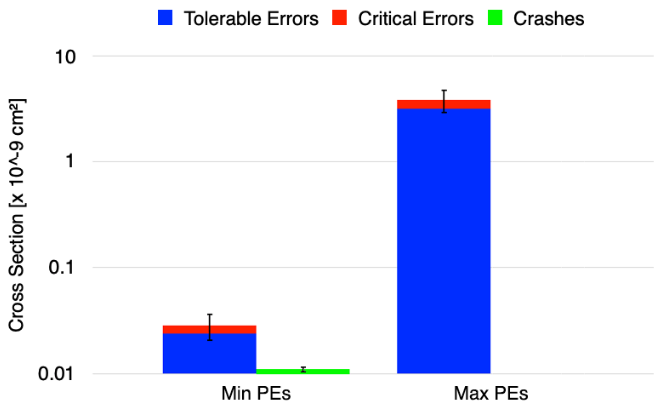

Fig. 7. Neutron cross section for the Min PE and Max PE versions of the MNIST CNN. Note that the y-axis is in logarithmic scale.

to complete the processing of input images (i.e., has lower performance). On the other side, we have a very large, fully parallel implementation, which occupies almost 100% of a state-of-the-art FPGA, but delivers extremely low latency. Such scenario was made evident in Section III-A (Table II and Fig. 3), but we believe that it is worth reiterating.

Fig. 7 plots the neutron cross section of the two designs. We classify output errors as tolerable or critical, depending on whether or not the image classification was affected. Noting that the y-axis is in logarithmic scale, and the difference in radiation sensitivity between the two implementations is striking: the fully parallel design is 133 times more likely to experience radiation-induced errors. Interestingly, it uses 130 times more resources than the version with far less PEs, confirming the direct relationship between resource utilization and cross section in SRAM-based FPGAs.

Furthermore, we should point out that the percentage of critical errors observed in our experiments was roughly the same in the two design variations (16% and 17%, for Min PEs and Max PEs, respectively). As there is no variation in data precision here, the impact of data corruption during computation is about the same in both cases, so this was an expected outcome. Finally, we can see that the more parallel version of the MNIST CNN did not experience any crashes. This is because, by being a fully pipelined, streaming architecture, it does not have/need any sort of FSM for control logic and therefore it never gets stuck.

Again, we should emphasize that the cross section only measures the level of radiation sensitivity of an algorithm/circuit/device. In order to get an estimation of failure rate, one must also consider a performance metric (execution time) as a variable. Thus, we show the MEBRF of our DUTs in Table IV, considering the neutron flux at sea level [35]. Surprisingly, despite having a much higher cross section, the fully parallel implementation of the network is able to complete over seven times more error-free executions than the smaller design. This is because, as reported in Section III-A (Fig. 3), the more parallel design is over 1000× faster.

Different than the analysis in Section IV-A, where the precision reduction led to smaller and faster hardware, we are seeing that the degree of parallelism in an architecture is

TABLE IV
MEBRFs FOR THE MIN PE AND MAX PE VERSIONS OF THE MNIST CNN

	Min PEs	Max PEs
MEBRF [x 10^{17}]	0.65	5.00

always a tradeoff between area and performance, which both directly impact the overall reliability of the system. From our experimental data, it seems that a faster (more parallel) design is the one that will deliver the lower failure rate, since the performance gains are outweighing the increased area and radiation sensitivity. However, as we said before, our case study CNN was specifically chosen to enable this analysis, and for most state-of-the-art neural network architectures, instantiating all of the necessary PEs in the FPGA would be unfeasible. Therefore, it follows that the optimal neural network hardware accelerator architecture (from the reliability standpoint) should really just be as parallel as possible.

C. Technology Node

Even though showcasing the difference between FPGAs in different technology nodes was not one of the main goals of this article, the INT8 circuit that was tested on the Zynq-7000 happens to be exactly the same as the Min PE design that was tested on the Zynq UltraScale+. Therefore, we have decided to plot and report their dynamic neutron cross sections in the same graph (Fig. 8). We can see that the older 28-nm CMOS technology is around one order of magnitude more sensitive to the radiation of the newer 16-nm FinFET technology, which, considering statistical errors, is in line with the static cross section numbers reported by Xilinx [38].

A number of previous works have dived into the exploration of differences in radiation sensitivity across technology nodes, through a mixture of charge collection simulations, and real-world beam experiments with neutrons/heavy ions [39], [40]. As such, our experimental data merely corroborates.

V. CONCLUSION

We have seen that reducing the data precision representation in CNNs, through state-of-the-art machine-learning frameworks that allow for little to no accuracy loss on their quantization processes, can significantly improve the overall reliability of safety-critical applications that rely on image processing. In summary, using a lower precision simplifies the hardware implemented on the FPGA, lowering its resource utilization, which means that it becomes less likely to get hit by impinging particles. At the same time, reduced precision hardware is usually faster, which further contributes to lowering the failure rate of neural networks. In addition to that, we have explored how the concept of error criticality affects the reliability analysis of a CNN intended for a classification task and how the adoption of

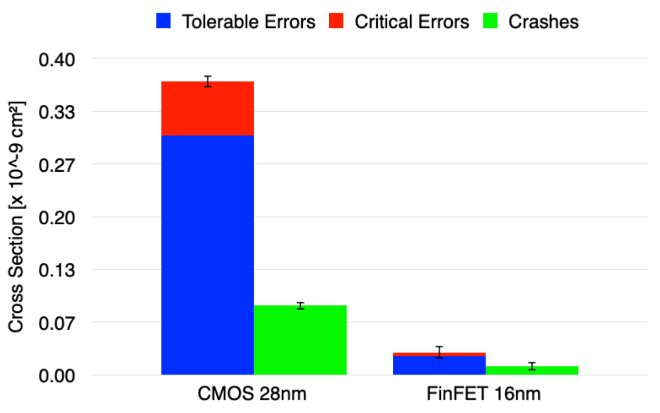

Fig. 8. Neutron cross section for the MNIST CNN on the 28-nm Zynq-7000 and the 16-nm Zynq UltraScale+.

tolerance intervals can significantly impact the overall reliability of an application. In general, we have concluded that, as long as the accuracy level remains the same after the quantization, the designer of an efficient hardware accelerator should opt for using 8-bit integer as opposed to floating-point representations.

Furthermore, we have analyzed how different degrees of parallelism (and the associated tradeoff between area and performance) affect the reliability of neural network accelerators on FPGAs. We have shown that, in general, the performance gains obtained through higher parallelism overshadow the cost paid in increased circuit area. Although the studied maximum degree of parallelism is often out of reach for most state-of-the-art CNN topologies, our results point to the fact that hardware accelerators for neural networks should be as parallel as possible, for improved reliability.

We believe that our findings regarding data precision and parallelism can offer some guidance for designing efficient and reliable hardware accelerators for neural networks in the future. As we have stated before, the machine learning field is in constant evolution, which means that the computing requirements for accelerating today's CNNs might not be the same in a few years' time. Our work can help novel accelerator architectures to be built with reliability in mind, ultimately enabling the deployment of neural networks in safety-critical applications. Finally, it is worth mentioning that we did not see any differences in error likelihood across classes in our 100-image input set. As a by-product of our experiments, we were also able to compare the radiation sensitivities of FPGAs in different technology nodes. Our experimental data corroborate with prior studies.

As future work, we intend to evaluate the reliability of Xilinx's proprietary deep-processing unit (DPU) [41], along with their recently released set of tools Vitis AI, which enable for quantization and compilation of neural network models straight out of industry standard frameworks such as TensorFlow. We also intend to analyze and improve (through adoption of efficient hardening techniques) the reliability of systolic array structures for matrix multiplication, which seems to be the current choice of architectural core for both Xilinx's DPU and Google's tensor-processing unit (TPU) [42].

References

[1] McKinsey. (Dec. 25, 2020). *Automotive Revolution—Perspective Towards 2030*. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insight%s/disruptive-trends-that-will-transform-the-auto-industry/de-de#

[2] Mercedes-Benz. (Dec. 25, 2020). *Mercedes-Benz Innovation: Autonomous*. [Online]. Available: https://www.mercedes-benz.com/en/innovation/autonomous/

[3] Volvo. (Dec. 25, 2020). *Autonomous Driving|Intellisafe*. [Online]. Available: https://www.volvocars.com/en-kw/own/own-and-enjoy/autonomous-driving

[4] Tesla. (Dec. 25, 2020). *Autopilot*. [Online]. Available: https://www.tesla.com/autopilot

[5] T. Samad, J. S. Bay, and D. Godbole, "Network-centric systems for military operations in urban terrain: The role of UAVs," *Proc. IEEE*, vol. 95, no. 1, pp. 92–107, Jan. 2007.

[6] NASA. (Dec. 25, 2020). *Mars 2020 Mission Perseverance Rover*. [Online]. Available: https://mars.nasa.gov/mars2020/

[7] NASA. (Dec. 25, 2020). *Mars Helicopter*. [Online]. Available: https://mars.nasa.gov/technology/helicopter/

[8] H.-C. Shin *et al.*, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1285–1298, May 2016.

[9] 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.

[10] M. Zhang, H. Li, G. Xia, W. Zhao, S. Ren, and C. Wang, "Research on the application of deep learning target detection of engineering vehicles in the patrol and inspection for military optical cable lines by UAV," in *Proc. 11th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2018, pp. 97–101.

[11] H. R. Kerner, K. L. Wagstaff, B. D. Bue, P. C. Gray, J. F. Bell, and H. Ben Amor, "Toward generalized change detection on planetary surfaces with convolutional autoencoders and transfer learning," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 10, pp. 3900–3918, Oct. 2019.

[12] M. Andersch, J. Lucas, M. A. LvLvarez-Mesa, and B. Juurlink, "On latency in GPU throughput microarchitectures," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2015, pp. 169–170.

[13] A. Aldahlawi, Y.-B. Kim, and K. K. Kim, "GPU architecture optimization for mobile computing," in *Proc. Int. SoC Design Conf. (ISOCC)*, Oct. 2019, pp. 247–248.

[14] M. Qasaimeh, J. Zambreno, P. H. Jones, K. Denolf, J. Lo, and K. Vissers, "Analyzing the energy-efficiency of vision kernels on embedded CPU, GPU and FPGA platforms," in *Proc. IEEE 27th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2019, p. 336.

[15] T. Piovesan, H. C. Sartori, J. E. Baggio, and J. R. Pinheiro, "CubeSat electrical power supplies optimization—Comparison between conventional and optimal design methodology," in *Proc. 12th IEEE Int. Conf. Ind. Appl. (INDUSCON)*, Nov. 2016, pp. 72–78.

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

[17] Xilinx. (Dec. 25, 2020). *7 Series FPGAs Configuration User Guide*. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_%Config.pdf

[18] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2018, p. 17.

[19] 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.

[20] F. F. dos Santos *et al.*, "Analyzing and increasing the reliability of convolutional neural networks on GPUs," *IEEE Trans. Rel.*, vol. 68, no. 2, pp. 663–677, Jun. 2019.

[21] F. Libano, P. Rech, L. Tambara, J. Tonfat, and F. Kastensmidt, "On the reliability of linear regression and pattern recognition feedforward artificial neural networks in FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 65, no. 1, pp. 288–295, Jan. 2018.

[22] 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.

[23] F. Fernandes dos Santos, C. Lunardi, D. Oliveira, F. Libano, and P. Rech, "Reliability evaluation of mixed-precision architectures," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 238–249.

[24] F. F. dos Santos, P. Navaux, L. Carro, and P. Rech, "Impact of reduced precision in the reliability of deep neural networks for object detection," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2019, pp. 127–132.

[25] F. Libano, B. Wilson, M. Wirthlin, P. Rech, and J. Brunhaver, "Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 67, no. 7, pp. 1478–1484, Jul. 2020.

[26] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[27] TensorFlow. (Dec. 25, 2020). *TensorFlow Model Optimization Toolkit—Post-Training Integer Quantization*. [Online]. Available: https://blog.tensorflow.org/2019/06/tensorflow-integer-quantization.htm%l

[28] 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.

[29] TensorFlow. (Dec. 25, 2020). *TensorFlow Lite*. [Online]. Available: https://www.tensorflow.org/lite

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

[31] 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. 149–154.

[32] 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.

[33] Xilinx. (Dec. 25, 2020). *Zynq-7000 SoC*. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[34] Xilinx. (Dec. 25, 2020). *Zynq UltraScale+ MPSoC*. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mps%oc.html

[35] JEDEC. (Dec. 25, 2020). *Jesd89a, JEDEC Standard*. Accessed: 2006. [Online]. Available: https://www.jedec.org/sites/default/files/docs/jesd89a.pdf

[36] 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.

[37] ISO. (Dec. 25, 2020). *ISO 26262*. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:26262:-2:ed-2:v1:en

[38] Xilinx. (Dec. 25, 2020). *Device Reliability Report*. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug116.pdf

[39] Y.-P. Fang and A. S. Oates, "Neutron-induced charge collection simulation of bulk FinFET SRAMs compared with conventional planar SRAMs," *IEEE Trans. Device Mater. Rel.*, vol. 11, no. 4, pp. 551–554, Dec. 2011.

[40] P. Nsengiyumva *et al.*, "A comparison of the SEU response of planar and FinFET d flip-flops at advanced technology nodes," *IEEE Trans. Nucl. Sci.*, vol. 63, no. 1, pp. 266–272, Feb. 2016.

[41] Xilinx. (Dec. 25, 2020). *Zynq DPU*. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3%_1/pg338-dpu.pdf

[42] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.