Reliability of Google's Tensor Processing Units for Embedded Applications

Rubens Luiz Rech Junior* and Paolo Rech*†
*Institute of Informatics, UFRGS, Porto Alegre, Brazil
†DAUIN, Politecnico di Torino, and DII, Università di Trento, Italy

***Abstract*—Convolutional Neural Networks (CNNs) have become the most used and efficient way to identify and classify objects in a scene. CNNs are today fundamental not only for autonomous vehicles, but also for Internet of Things (IoT) and smart cities or smart homes. Vendors are developing low-power, efficient, and low-cost dedicated accelerators to allow the execution of the computational-demanding CNNs even in embedded applications with strict power and cost budgets.**

Google's Coral Tensor Processing Unit (TPU) is one of the latest low power accelerators for CNNs. In this paper we investigate the reliability of TPUs to atmospheric neutrons, reporting experimental data equivalent to more than 30 million years of natural irradiation. We analyze the behavior of TPUs executing atomic operations (standard or depthwise convolutions) with increasing input sizes as well as eight CNN designs typical of embedded applications, including transfer learning and reduced data-set configurations. We found that, despite the high error rate, most neutrons-induced errors only slightly modify the convolution output and do not change the CNNs detection or classification. By reporting details about the fault model and error rate, we provide valuable information on how to evaluate and improve the reliability of CNNs executed on a TPU.

I. Introduction

Convolutional Neural Networks (CNNs) are today the most effective (and efficient) way to detect an object in a scene [1]. By applying various filters to the input image, convolutional layers extract information (feature maps) that is then passed to the downstream layers to detect and/or classify objects. The number of layers, the kind of filter applied, and the structure of the CNN is engineered to achieve the desired accuracy and efficiency. To ensure very high accuracy along with real-time detection, both being fundamental for autonomous vehicles, it is necessary to execute CNNs on highly performant, costly, and power hungry devices, such as the latest GPUs or very big FPGAs. Nevertheless, many other applications, with less strict accuracy and timing constraints, can benefit from CNNs execution. This is the case of Internet of Things (IoT), smart homes, or smart cities, in which detecting or identifying a relatively low number of objects can significantly improve the user experience and the overall system features. In these embedded applications, the cost and power consumption need to be minimized, while still guaranteeing sufficient accuracy.

Lately, vendors have developed low-cost accelerators for CNNs execution, named *EdgeAI* devices, such as Google Coral Tensor Processing Units (TPU) or NeuroShield [2]. EdgeAI devices are only able to execute elementary operations (i.e., convolutions and some other matrices operations) in low precision (16-bit floating point or even 8-bit integer) [3]. Coupled with a good software framework (e.g., Tensor Flow) that runs on a host device, EdgeAI devices significantly reduce the time and power consumption of convolution. As EdgeAI devices are likely to be used at scale and in distributed systems, it is fundamental to investigate their reliability.

In this paper, we investigate the reliability of Google Coral TPU through accelerated neutron experiments that count for more than 30 million years of natural radiation. We define the fault model on the atomic operations (standard and depthwise convolutions) and compare the error rate and the prediction failures of eight CNNs configurations. We report the FIT rates of typical CNNs for embedded applications, such as SSD MobileNet v2, SSD MobileDet, Inception v4, and ResNet-50. We also evaluate the impact on the CNN reliability of *transfer learning*, which is adopted to have a quick re-training of the CNN for a specific application. Additionally, as in most embedded applications it is sufficient to identify/classify only few objects, we show that reducing the number of object classes is highly beneficial for the CNN reliability.

The rest of the paper is organized as follows. In Section II, we provide a background on CNNs and on the Coral TPU, useful to understand the experimentally observed behaviors. In Section III, we describe the experimental setup we developed and the software (convolutions and CNNs) we test. Experimental results are presented and discussed in Section IV, highlighting the implications for future hardening solutions for Coral TPU, while Section V concludes the paper.

II. Background

In this Section, we review the characteristics of CNNs, the Coral TPU architecture, and highlight our contributions.

A. EdgeAI accelerators and Transfer Learning

A CNN is a sequence of layers, each applying a specific function to the input frame (or feature map) [1]. One of the key, and most computing intense, steps of CNNs used for object detection is *convolution*. EdgeAI accelerators, like NeuroShield and Google Coral TPU, are low-power and low-cost devices designed to perform heavy machine learning computations in the context of embedded applications.

Figure 1 shows the high level schematic of the Coral TPU architecture which is mainly composed of a *systolic array* fed by a large set of input buffers (not protected by ECC). The array outputs the product of the model weights and

978-3-9819263-6-1/DATE22/©2022 EDAA

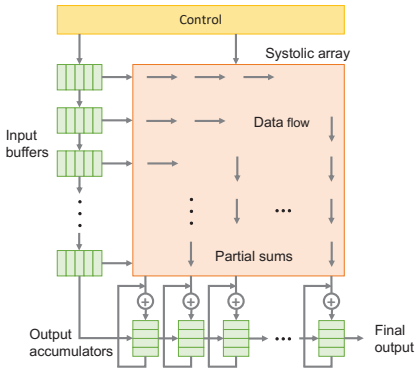

Fig. 1. Schematic of the Coral Edge TPU architecture. Adapted from [3].

each layer's input into the activation unit, where the partial sums are accumulated and the activation function is applied. Therefore, the TPU can perform convolutions, which are a fundamental block for machine learning applications, in an extremely power- and performance-efficient manner (the TPU delivers 2 TOPS per watt). For minimizing data transfers and speed up calculations, all data computed and stored within the TPU is represented as 8-bit unsigned integers (UINT8). The device performs the quantization and de-quantization steps for interfacing with the host floating-point representations.

Since Coral TPU is simply an accelerator, it must be connected to a host device through PCIe or USB 3.0 [2]. The software layer of the Coral TPU is based on TensorFlow Lite, which is a light version, optimized for embedded devices, of Google's TensorFlow framework [4].

B. CNNs Reliability

CNNs have already been shown to be particularly susceptible to transient faults, leading to a *Silent Data Corruption* (SDC, error at the output) or a *Detected Unrecoverable Error* (DUE, a crash or reboot) [5]–[7]. Through beam experiments and fault-injection it has been demonstrated that the corruption of each layer has a different probability of affecting the CNN output, being the convolution layer the responsible for most observed errors [5]–[7]. The corruption can (1) be masked without affecting the output, (2) can reach the output but keep the classification/detection unaltered, or (3) can spread and modify the output in a way that impacts the CNN functionality. Thanks to the intrinsic approximate nature of CNN computation, most of the errors do not turn into system failures, i.e., they modify the CNN output without affecting classification/detection. This has been proved for GPUs [5], FPGAs [8], and NeuroShield devices [9], [10]. Unfortunately, despite the intrinsic approximate nature of CNNs, their radiation-induced misdetection rate is still too high to be employed in safety-critical applications [5], [6].

C. Contributions

To the best of our knowledge, this is the first work that studies Coral TPU devices reliability to transient faults. While previous works have discussed the reliability of the Arduino NeuroShield [9], [10], showing that the error rate of the small EdgeAI accelerators is far from being negligible (higher than 10^2 Failure In Time - FIT), unlike previous publications, we engineered a setup to test also atomic operations performed by the accelerator (convolutions) with different sizes and depths (2D and 3D) and we distinguish between critical and tolerable errors in CNNs execution. This information is useful to deeply investigate the fault model induced by neutrons in the TPU.

An embedded application typically does not require to identify/classify all the thousands objects of the training datasets. Normally, the designer selects a subset of the dataset for the EdgeAI accelerator and trains the CNN accordingly. Transfer learning is a machine learning technique that allows to speed up the training process by reusing the knowledge coming from another machine learning model. As part of our contribution, we show that transfer learning does not impact the error rate of the CNN, while reducing the number of object classes significantly lowers the CNN error rate.

III. Reliability Evaluation Methodology

In this Section, we detail the software we test and the neutron experiments setup.

A. Atomic Operations and CNNs Configurations

To characterize the device fault model, we evaluate the reliability of the two types of atomic operations supported by Coral: *standard* and *depthwise* convolution. Standard convolutions are normal 2D convolutions while depthwise convolutions have an input composed by multiple channels and each one is convolved separately with its respective kernel. Since CNNs usually perform image prediction, in our experiments, the inputs of depthwise convolutions are always composed of three channels (3D), as for the RGB colors. We run experiments with square matrices of size ranging from 256 to 1,250 (INT8) as inputs and square kernels of fixed sizes: 40 for standard and 20 for depthwise convolutions.

Besides convolutions, we evaluate the reliability of *eight* neural network configurations, in which we vary the network architecture, the dataset and the training methodology. We consider CNNs that perform image classification and object detection. Object detection is a more complex task than classification, as multiple objects in the image have to be located and then classified. We consider two image classification CNNs, i.e. Inception V4 [11] and ResNet-50 [12], both trained with ILSVRC [13] dataset that supports 1,000 different object classes. We also consider two object detection CNNs, i.e., SSDLite MobileDet [14] and SSD MobileNet V2 [15], trained with COCO [16] dataset, which embraces 90 classes. The models for these CNNs are based on TensorFlow Lite.

In addition to these four models/configurations, we also retrain SSD MobileNet V2 with two other datasets: a subset of the COCO dataset (14 classes) and a subset of the Oxford-IIIT Pet [17] dataset (2 classes). Our goal is to evaluate whether and how a reduced number of objects to be detected impacts the device error rate. The retraining process is done with and without the application of *transfer learning* technique, in

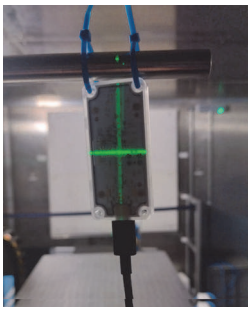

Fig. 2. The Coral TPU aligned with the high energy neutron beam at ChipIR. The host device, a Raspberry Pi 4, is connected with a 2 meters USB-3 cable and placed well out of the beam.

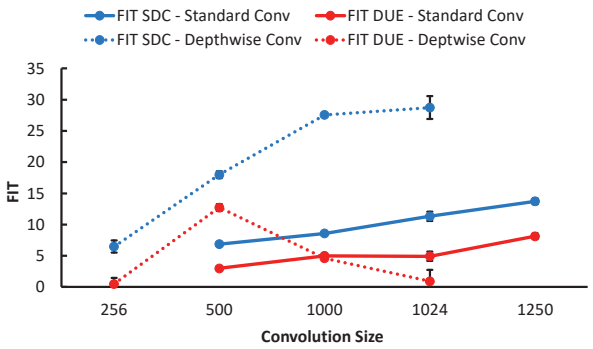

Fig. 3. SDCs and DUEs FIT rates for standard (2D) and depthwise (3D) convolutions, with increasing input sizes, exposed to high energy neutrons at Chip IR. Data is shown with 95% confidence intervals.

which the knowledge from another machine learning model is reused in order to speed up the learning process.

Considering both convolutions and the different CNNs configurations, we provide experimental data obtained on 16 benchmarks for evaluating the TPU reliability.

B. Neutron Experiment Setup

It is very challenging, if not impossible, to implement a software fault injection for the Coral TPU as its internal states are hidden to the user. The level of control allowed by the Coral TPU is restricted to send the input, operation to be performed and, then, read the results. The only possible software fault injection would, then, be the naive input modification, that provides little insights on the accelerator reliability. Thus, we have preferred to expose the device to accelerated neutron beams that (1) inject faults in the whole device, including the internal states, and (2) provides realistic error rates and fault models. Beam experiments, however, do not allow to identify the resource whose corruption generates the observed error.

Atmospheric neutrons experiments were performed at the ChipIR facility at the ISIS spallation neutron source of the Rutherford Appleton Laboratory (RAL), UK. ChipIR delivers a neutron spectrum as similar as possible to the atmospheric one. Thus, the experimental data can be used to estimate the error rate of the device in a realistic application, expressed in Failures In Time (FIT), i.e., errors per 10^9 hours of operation.

During the test, Silent Data Corruptions (SDCs) are detected comparing the experimental output with the pre-computed fault-free output. When a mismatch is detected, an SDC is counted and data is downloaded for further analysis. Detected Unrecoverable Errors (DUEs) are detected with a watchdog.

A picture of the Coral TPU at ChipIR is shown in Figure 2. At the position of the Coral, the average flux was of about $3.9 \times 10^6 n/cm^2/s$ and the plastic package of the TPU is transparent to neutron. We test the device for more than 241 effective hours (without considering the setup, load input, download output, and reboot time), resulting in a fluence higher than $3.41 \times 10^{12} n/cm^2$. When scaled to the terrestrial flux ($13 n/cm^2/h$), this fluence corresponds to more than 30 million years of natural irradiation. As the natural flux is much lower than the experimental one, to avoid artifacts, we have engineered the experiment (exposure time, flux, computation) to make it highly unlikely for two different neutrons to generate a fault during one single execution. The error rate at ChipIR was lower than 1 error every 1,000 executions.

IV. Experimental Results

In this Section we present and discuss the neutron experiments data on convolutions and CNNs.

A. Atomic Operations

Aiming to measure the error rate and to identify the fault model of the simplest and most light-weighted operations the TPU can execute, we run two different types of convolutions: standard (2D) and depthwise (3D). Tests are performed with squared matrices of varying sizes as inputs and fixed kernel size. We recall that Coral TPU only executes INT8 operations.

Figure 3 plots the FIT rates (SDCs in blue and DUEs in red) for the tested sizes of both convolution types. The results for size 256 of the standard convolution are not shown as only few SDCs were observed during a long period of tests (the amount of processed data is very small) while depthwise convolution for 1,250 input cannot be executed on the TPU since it exceeds the device computing capabilities.

As shown in Figure 3, the **SDC FIT rate** increases with the size of the convolution input. Intuitively, the systolic array becomes more occupied due to the increasing amount of data that needs to be processed. The use of additional resources increases the hardware area that, if corrupted, can lead to a SDC. However, the SDCs increase with the input size for *all* possible sizes is *not* obvious. Typically, when the software saturates the hardware computing capabilities, the SDC rate reaches a maximum as the same resource is used twice and, thus, the area that can be corrupted remains unaltered. The exposure time is doubled, doubling the number of errors but also the number of impinging particles (the error rate is unaltered). Additional operations, then, increase the execution time but not the error rate (assuming that at most one neutron can induce an error in one execution). The trend in Figure 3 for SDCs shows that, for the TPU, the computing capabilities

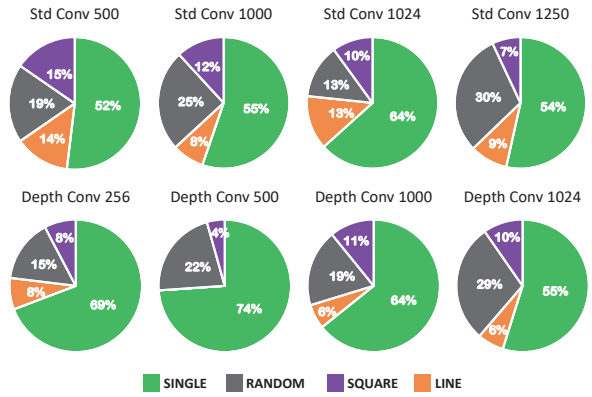

Fig. 4. Geometric distribution of the corrupted elements in the output of convolutions observed at ChipIR.

(and the error rate) do not saturate until reaching the maximum allowed input size. Smaller sizes underutilize the Coral TPU.

Our data in Figure 3 shows that, for a given input size, depthwise convolutions have higher SDCs error rate when compared to standard convolutions (179% higher, on average). Also, the error rate of 3D convolutions increases with the input size at a 10 times higher rate than the 2D convolutions. This is, again, related to the fact that more area is used when processing 3D convolutions.

The **DUEs FIT rate** does not follow the SDCs trend. This should not surprise since, as shown in previous work [18], DUEs have a dominant component that depend exclusively on the hardware and not on the software layer. DUE FIT rate, then, is biased by the sensitivity of resources that are independent of the executed code (or input size).

Figure 4 shows the **geometric distribution of the convolutions output elements** that were found corrupted. When an SDC is detected, we download the whole output matrix and identify how many elements in the output are corrupted and categorize the corruption based on their spatial distribution. When multiple corrupted elements sit in the same row or column, we count a *Line* error, when they are distributed in a square (a whole portion of the output matrix is corrupted), we count a *Square* error. When we see multiple corrupted elements that are neither on a Line nor on a Square, we count a *Random* error. We recall that we engineered the experiments (input sizes and flux) not to have more than one neutron generating an error in one execution. Thus, multiple errors are caused by the spread of the single neutron corruption to multiple operations and not by multiple neutrons corruptions.

We observe in Figure 4 that the geometric distribution is very similar across all sizes and convolution types. Most of the time, just a single element of the output matrix is corrupted. The second most frequent SDC geometry is Square, followed by Random and Line distributions.

The fact that simple corrupted elements is the most common distribution for the TPU is in contrast with what has been observed for GPUs, CPUs, and FPGAs, for which the majority of the corrupted matrices have multiple corrupted elements [5], [19]. This is due to the different matrix multiplication implementation. On other devices, matrix multiplication is executed as a code, with a sequence of instructions, and it is based on *block tiling*. So, a fault that corrupts the computation of a given block B_i propagates spreading as the (corrupted) result of the block is merged with the result of another block B_j. On the contrary, on TPUs, matrix multiply is a single instruction executed in a *systolic array* [3]. Due to the higher level of concurrency delivered by the systolic array, a corrupted element has a lower chance to be merged with other elements and, therefore, the propagation of errors is reduced. As it has been shown that multiple corrupted elements in the output matrix are the main cause for missdetections or missclassifications in CNNs [5], the fact that the TPU is less prone to have multiple output errors is a promising results for its reliability in executing CNNs (we confirm this observation in the next subsection).

Additionally, we have observed that **the magnitude of the errors** (i.e., how much the corrupted value is different from the expected one) is, overall, very small. The absolute difference between the expected and the corrupt element value is, in fact, exactly *one* (e.g., the expected value is 80 and the corrupted one is 81 or 79) in 91% of the observed SDCs. Please recall that only INT8 operations can be performed on the TPU. Also, when the error magnitude is greater than one, the difference between the corrupted and expected value is a power of 2 (i.e., a single bit flip usually occurs). This happens regardless of the convolution type, as this is a consequence of the device's intrinsic fault model. Again, this is in contrast with data observed for other devices, for which the magnitude of the error can be significantly higher (orders of magnitude) [5], [7], [19]. This is another promising result for the TPU reliability in executing CNNs, as a smaller error magnitude has a lower impact on the final detection/classification.

B. Convolutional Neural Networks

With regards to neural networks, we test the reliability of eight different configurations by varying the network architecture, dataset and training procedure, with or without transfer learning, in which the knowledge learned by other model is reused especially to reduce the training time and, in most cases, to have better prediction performance/accuracy.

We leveraged on four CNNs models that were trained and made available by Google (Inception V4, ResNet-50, SSD MobileDet, and SSD MobileNet V2). We also retrained MobileNet using two different datasets (a subset of the COCO dataset and a subset of the Oxford-IIIT Pet dataset) with and without applying transfer learning. By retraining the machine learning models, we want to evaluate: (1) how the number of object classes supported by the CNN impacts its reliability, since the datasets used for retraining have much less classes than the original COCO dataset and (2) whether transfer learning has an effect in the detection resilience.

Figure 5 plots SDC (left Y-axis) and DUE (right Y-axis) FIT rates for the eight CNN configurations. Please note that **SDCs are about 2 orders of magnitude more likely than**

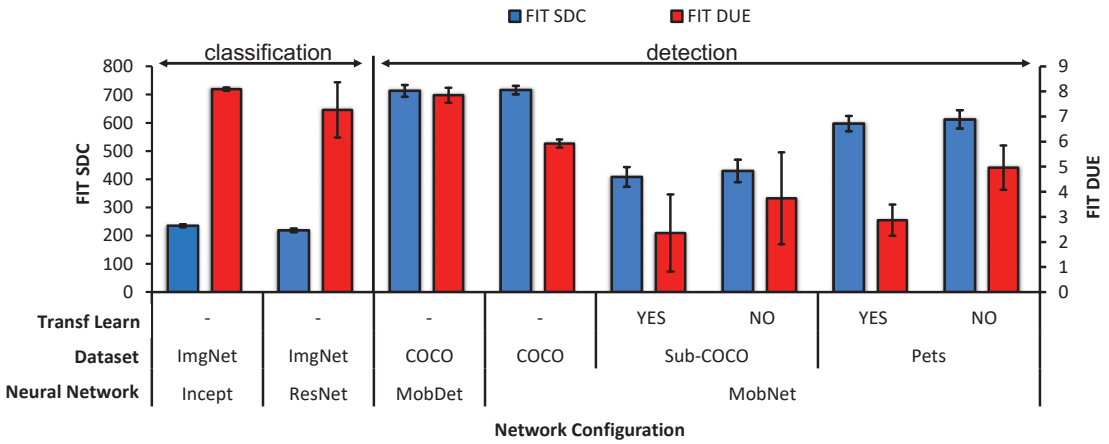

Fig. 5. FIT rates for the eight neural network configurations that were exposed to high energy neutrons at ChipIR. The four configurations on the left side are the original models that Google provide on the Coral Edge TPU official website. The ones disposed on the right side are retrained versions, with and without transfer learning, of the MobileNet with different datasets. The values for SDC FIT rates are ploted on the left Y-axis and DUEs on the right.

DUEs. This is in contrast with GPUs, for which the continuous host-device synchronizations exacerbate the DUE rate [5] and in accordance with FPGAs and NeuroShield, that do not require synchronizations, for which almost no DUE has been observed [8], [9]. The TPU, then, implements a very reliable interface with the host device.

ResNet and Inception, which perform image classification (not detection), have the highest DUEs FIT rates. This is related to their model size, which is 5 to 7 times larger than the MobileNet model. Apart from the retrained networks, which have the lowest DUE FIT, the overall DUE rate is similar among the CNNs which enforces the fact that DUEs are mostly related to the hardware attributes rather than the algorithm.

In Figure 5 we label as SDC any mismatch between the experimental and expected output of the CNN. We will discuss whether SDCs modify detection/classification next. The lowest SDC FIT rate (270 FIT) is measured for Inception, a similar neural network as the one tested for the NeuroShield in [9], that showed a FIT rate of 10^2. To put values in Figure 5 in perspective, MobNet trained with the COCO dataset has an SDC rate of 716 FIT. As FIT are errors per a billion hours of operation and the number of devices used in IoT applications is approaching 10 billion, if each included a Coral TPU running MobNet, we would see a total of 7,160 SDCs per *hour*.

From the results plotted in Figure 5, we observe that **detection networks have higher SDC rates** than the classification ones. This is because, although the classification models are larger and, possibly execute more operations, the detection output is much more complex. For the classification task, the output values simply represent the probability of each object class while, in the detection task, the output is composed of six values for each possibly detected object: its class, its probability and its position (x, y, width, height). The position elements are much more sensitive to the effects of faults and, thus, detection CNNs will have higher error rates.

Transfer learning (TL) does not seem to have a significant impact on the FIT rate of neural networks. This technique has shown to decrease the SDC error rate of just 2-5% when compared to the analogous configuration without TL. However, the training process tend to converge much faster with this strategy and, in our case, it reduces the CNNs learning time of of about 50%. TL is then a good solution when a quick CNN re-training is necessary, as it is fast and does not impact the error rate negatively.

Our results also show that the **retraining** of MobileNet with the COCO subset (14 classes) reduces the SDC error rate by 43% when compared to the original model trained using the 90 classes of the original COCO dataset. The same network but trained with the Pets dataset (2 classes) have higher error rate than the one trained with Sub-COCO, but still smaller than the one obtained for the original with the entire COCO dataset. This trend evinces that, with less classes to be considered, the detection process gets simpler and the error rate is reduced. Therefore, the training of CNNs should target the real application needs and include classes of object that are really relevant to the context of the application.

It is well known that not all SDCs are critical for neural networks execution. Figure 6 shows, for the configurations presented in Figure 5, the percentage of **SDCs that critically affect the classification/detection outcome**, i.e, SDCs that (1) change the number of detected objects or (2) their classes or (3) significantly alter their position (less than 50% of intersection between the expected and the corrupted result).

As shown in Figure 6, the great majority of SDCs does not modify classification/detection. The percentage of critical errors is much lower for the TPU than for other devices for which between 8% and 60% of errors have been shown to be critical for the CNN [5], [8]. The less critical impact of neutrons in the CNN output for the TPU is directly related to the fault model of convolutions (single element corrupted, small difference with the correct result), shown earlier.

From Figure 6, it is clear that SDCs in MobileDet is way

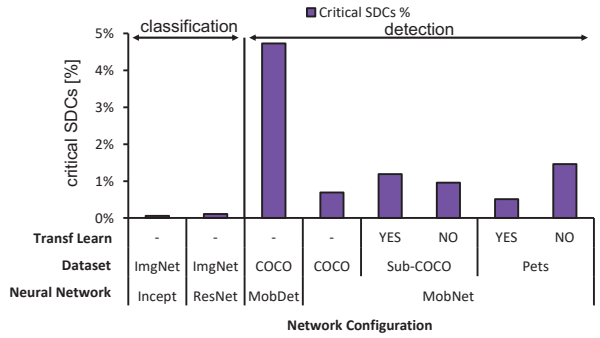

Fig. 6. Percentage of SDCs that critically affect the classification/detection outcome of the CNN configurations that were tested at ChipIR.

more critical than the other network architectures. Comparing it to the MobileNet, which also performs detection, MobileDet has less model parameters, a 13% larger input size and a 50% smaller output. The fact that MobileDet has half of the number of output elements makes each one of them twice more significant for the detection outcome and, therefore, the corruption of a single output value tends to be more critical in MobileDet than MobileNet.

Transfer learning does not seem to have a consistent impact on the criticality of the SDCs. In the case where MobileNet is retrained with the Pets dataset, transfer learning decreases the number of critical errors by almost 3 times. On the other hand, when trained with COCO subset, it makes the CNN 20% more susceptible to critical SDCs. Further studies are necessary to understand the reasons for this opposite trend. The differences, though, are not very high, confirming that transfer learning does not impact the CNN FIT rate.

Naturally, SDCs in classification CNNs (Inception and ResNet) are considerably less critical since only a few values, the highest ones out of 1,000 output values, are indeed relevant to the outcome of the classification process. Therefore, although the SDCs are propagated to the network raw output, most of them do not influence the classification result, as confirmed by our data in Figure 6.

V. Conclusions and Future Work

In this paper we have evaluated the reliability of Google Tensor Processing Units to high energy neutrons. First, we have understood how neutrons impact the execution of 2D and 3D convolutions, which are the atomic operations for the TPU, of increasing input size. Besides the linear dependence between the input size and the SDC rate, we have seen that most neutrons corrupt only one element of the output matrix and the corrupted value is very close to the expected value. Then, we have executed eight different configurations of neural networks on the irradiated TPU. We have seen that detection networks have a much higher error rate than classification networks and that transfer learning does not significantly modify the error rate. Finally, the great majority of errors are not critical for the neural network execution, which is strictly related to the fault model observed for convolutions.

VI. Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 886202 and from The Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. Neutron beam time was provided by ChipIR (DOI: 10.5286/ISIS.E.RB2000161).

References

[1] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[2] Google, "Google Coral," Accessed: 2021-09-14. [Online]. Available: https://coral.ai

[3] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080246

[4] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

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

[6] A. Bosio *et al.*, "A reliability analysis of a deep neural network," in *2019 IEEE Latin American Test Symposium (LATS)*, 2019, pp. 1–6.

[7] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3126908.3126964

[8] F. Libano *et al.*, "On the reliability of linear regression and pattern recognition feedforward artificial neural networks in fpgas," *IEEE Transactions on Nuclear Science*, vol. 65, no. 1, pp. 288–295, 2018.

[9] S. Blower *et al.*, "Evaluating and mitigating neutrons effects on cots edgeai accelerators," *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 1719–1726, 2021.

[10] R. M. Brewer *et al.*, "The impact of proton-induced single events on image classification in a neuromorphic computing architecture," *IEEE Transactions on Nuclear Science*, vol. 67, no. 1, pp. 108–115, 2020.

[11] C. Szegedy *et al.*, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[12] K. He *et al.*, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[13] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[14] Y. Xiong *et al.*, "Mobiledets: Searching for object detection architectures for mobile accelerators," *CoRR*, vol. abs/2004.14525, 2020. [Online]. Available: https://arxiv.org/abs/2004.14525

[15] M. Sandler *et al.*, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[16] T. Lin *et al.*, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[17] O. M Parkhi *et al.*, "The Oxford-IIIT PET DATASET," Accessed: 2021-08-27. [Online]. Available: https://www.robots.ox.ac.uk/ vgg/data/pets/

[18] V. Fratin *et al.*, "Code-dependent and architecture-dependent reliability behaviors," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 13–26.

[19] D. Oliveira *et al.*, "Experimental and analytical study of xeon phi reliability," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 28:1–28:12. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126960