Impact of Tensor Cores and Mixed-Precision on the Reliability of Matrix Multiplication in GPUs

Pedro Martins Basso, Fernando Fernandes dos Santos, and Paolo Rech

***Abstract*—Matrix multiplication is a cornerstone application for both High Performance Computing and safety-critical applications. Most of operations in Convolutional Neural Networks for objects detection, in fact, are matrix multiplication related. Chip designers are proposing novel solutions to improve the efficiency of the execution of matrix multiplication. In this paper, we investigate the impact of two novel architectures for matrix multiplication (i.e., tensor cores and mixed precision) in GPUs reliability. Additionally, we evaluate how effective the embedded Error Correcting Code is in reducing the matrix multiplication error rate. Our results show that low precision operations are more reliable and tensor core increases the amount of data correctly produced by the GPU. However, reducing precision and the use of tensor core significantly increase the impact of faults in the output correctness.**

I. Introduction

The high parallelism and computing efficiency of Graphics Processing Units (GPUs) make them extremely interesting for both High Performance Computing (HPC) and safety-critical applications such as autonomous vehicles. Convolutional Neural Networks (CNNs) on GPUs have been shown to detect and classify objects with high accuracy in almost real-time [1]. In CNNs, a kernel filter is convolved with a matrix to extract specific features of the image. As the kernel filter slides over the input matrix, every position of the input is multiplied and accumulated with every position of the kernel. As a result, about 80% of operations of modern object-detection frameworks are matrix-multiplication related.

The high-computational demand for object detection and the huge volumes involved with automotive market push vendors to find novel solutions to efficiently execute convolution and, then, matrix multiplication. The latest novelties introduced in the market are *tensor core* and *mixed-precision*. Tensor core is an architectural solution to speed up the execution of parallel matrix multiplication. Using tensor cores, it is possible to achieve more than a 9x performance advantage versus a traditional matrix multiplication on GPUs and up to 47x performance advantage versus a CPU-based system [2]. Mixed-precision allows the user to tune the data and operation precision (64 bits, 32 bits, 16 bits) with the application real needs. As convolution (and image processing in general) is intrinsically approximate, researchers have found unnecessary to execute CNNs and other applications in full precision. The higher the precision, the higher the circuit area, execution time, and power consumption. Executing CNNs with low precision has been shown to reduce significantly GPUs power consumption and execution time, with negligible impact on the object detection accuracy [3].

While the computer architecture and artificial intelligence communities succeeded in proposing solutions to improve the efficiency and performances of GPUs, we need to carefully evaluate their impact on the device and application reliability. Both tensor core and mixed-precision, in particular, were explicitly introduced to speed up CNN for object detection frameworks, a key feature for safety-critical applications such as self-driving vehicles. The reliability of both novel solutions, then, becomes paramount. We cannot trade off performances for reliability in safety-critical applications.

Tensor core and mixed-precision are likely to impact (positively or negatively) the reliability of the device. Tensor core is an entirely new functional unit, whose area (and then cross-section) is expected to be bigger than ordinary ADDs and MULs but could be more efficient. Reducing data precision reduces the exposed area and, thus, the error rate. However, a fault in low-precision data is likely to be more critical than a fault in high-precision data. A corruption of the least significant positions of the mantissa, in fact, may lead to (wrong) data sufficiently close to the correct value. Reducing precision reduces the margin for acceptable faults as well. In the paper, we compare not only the FIT rates of various implementations of matrix multiplications, but we also compare the impact of faults in the output. As we show, reducing precision and using tensor core as the drawback of increasing the magnitude of the error in the output.

In this paper we present, in Section II, the novel architectural solutions introduced in the market and investigate how they impact the reliability of modern GPUs. Then, in Section III, we present the experimental methodology adopted for our study. In Section IV, we discuss the obtained results, including the comparison of the FIT rate of tensor core and software implemented matrix multiplication, the impact of data and operation precision, and the benefit of using Error Correcting Codes (ECC) in reducing GPU error rate. Additionally, we study the impact of faults in the matrix multiply output values as well as the amount of data that can be correctly processed before experiencing a failure. Finally, Section V, concludes the paper.

II. Background

In this section, we review previous findings on GPUs reliability and present background material on mixed-precision and tensor core features.

Pedro Martins Basso, Fernando Fernandes dos Santos, and Paolo Rech are with the Instituto de Informatica, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil, email: {pmbasso, ffsantos, prech}@inf.ufrgs.br

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2020.2977583, IEEE Transactions on Nuclear Science

A. GPU Vulnerability

A radiation-induced transient error in a computing device can be masked, produce a Silent Data Corruption (SDC) (i.e., the incorrect program output), or induce a Detected Unrecoverable Error (DUE) (i.e., program crash or device reboot). As GPUs were initially designed for graphics rendering, a task where high reliability is not a concern, their architecture has some inherent reliability weaknesses [4]. A single particle-induced fault in the GPU hardware scheduler, or in the shared resources, is likely to affect several parallel threads, leading to the corruption of multiple output values [5]. High-end GPUs protect their main storage structures with Single Error Correction Double Error Detection (SECDED) ECC, while some major GPU resources are left uncovered, e.g., flip-flops in pipeline queues, logic gates, block and warp schedulers, and instruction dispatch units.

B. Mixed-Precision

Approximate computing gained importance in the electronics market in the last years, demanding new architectural solutions. To be defined as mixed-precision an architecture must support at least two of the five floating point arithmetic specifications defined by the IEEE 754 (16, 32, 64, 128, and 256 bits). Mixed-precision architectures are not limited to a specific domain, they can be used in low-power devices, FPGAs, GPUs, general purpose CPUs, etc. [2], [6].

The interest in mixed-precision architectures raised as a range of applications, including neural networks, object detection, physical simulations, etc., does not require the full double or single precisions specified by IEEE754. The area to implement the floating point functional units and the power required to execute operations grows quadratically as the precision increase. Using double or single precisions even when lower precision could be sufficient, then, adds a non-negligible unnecessary overhead. It has been shown that using mixed precision has performance improvement, in fields such as HPC, Deep Learning, physics simulation, and approximate computing [3], [7], [8].

Nowadays, the biggest electronic devices brands offer support to half (16 bits), single (32 bits), and double (64 bits) precisions operations in their architectures [2]. Mixed-precision hardware is not limited to the standard x86 instructions (e.g., ADD, MUL, Fused Multiply-Add (FMA)), some architectures like NVIDIA Volta offers support to tensor operations (i.e., matrix multiply). A part of our work is to evaluate how mixed-precision arithmetics impact the application reliability.

C. Tensor Core

NVIDIA released, starting from the Volta microarchitecture, a specialized computing unit called *tensor core*. A tensor core is a specific hardware circuit designed to perform matrix multiplications, with significant advantage with respect to the software implemented matrix multiplication that uses a sequence of ADD and MUL. A tensor core is capable of performing a 4×4 matrix multiplication, in hardware, in one GPU clock cycle. Thanks to multithread programming it is also possible to execute up to 16×16 matrix multiplications, again in one clock cycle [9]. Bigger matrixes are divided, at compiler time, into 4×4 or 16×16 matrixes, that are multiplied in parallel or queued if all tensor cores are busy.

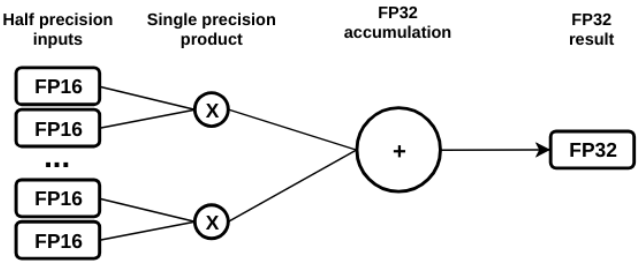

Fig. 1. Tensor cores Matrix-Multiply-Add operation. Adapted from [2]

Compared with the other GPU logic units, tensor core units are bigger and faster. The strategic choice of designing a novel hardware circuit to execute just matrix multiplication in mixed-precision is dictated by the importance of these operations for the training and execution of CNNs. Matrix multiplication, in fact, accounts for more than 80% of operations in a CNN for object detection [1]. Moreover, as CNNs are intrinsically approximated, half precision data and instructions in tensor cores contribute significantly to increase the throughput without affecting accuracy. Volta's mixed-precision tensor cores boost performance by more than 9x compared to others GPU's architecture [2].

The tensor core is based on Matrix-Multiply-Add (MMA) operation, as illustrated in Figure 1. Each tensor core performs a multiplication of two matrices and adds the result to an accumulator. The inputs and outputs may be half or single floating-point precision, however the latter are cast in hardware to half precision before execution. It is worth noting that, while values that are multiplied in CNN get smaller and smaller at each iteration (the exponents are negative), the result is then accumulated with another value that might be much larger. To avoid precision loss, accumulation is performed in single precision.

D. Matrix Multiplication on Convolutional Neural Networks

CNNs are one of the most efficient ways to perform image classification, segmentation and object detection. One of the key steps when using CNNs for object detection is convolution. A kernel filter is convolved with a matrix to extract specific features of the image. The kernel filter slides over the input matrix, multiplying and accumulating products at every position of the input with every position of the kernel. Each block is reorganized as a row of matrix A and the filter kernel is replicated as columns of a matrix B and then, convolution is then computed as $A \times B$.

As GPUs are highly efficient exploiting local memory, that has low latency and high bandwith, matrix multiplication can be easily accelerated and parallelized. The convolution and multiplication step are totally linked, previous works show that almost 80% of the operations executed in the most common object detection frameworks are matrix multiplication related, being one of the most important part of the CNNs execution.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2020.2977583, IEEE Transactions on Nuclear Science

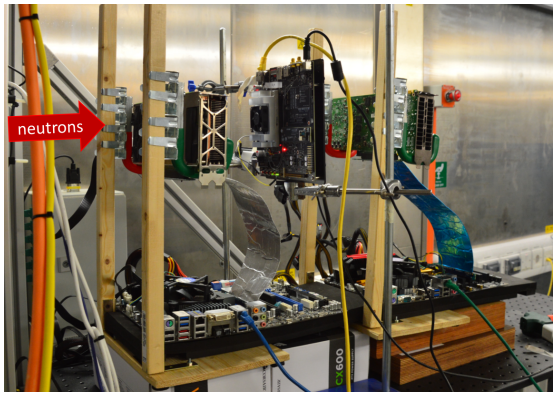

Fig. 2. Part of the experimental setup at ChipIR.

An error on the convolution step may lead to an incorrect output detection [10]. Therefore, it is strongly necessary to evaluate and study the reliability of matrix multiplication in GPUs.

III. Evaluation Methodology

To evaluate the reliability of tensor core and mixed-precision, we test a *Titan V* GPU and a *V100* GPU, both designed with a Volta architecture. The Titan V and the V100 share the same hardware core, the only difference between the GPUs is that only the V100 has ECC. Both GPUs are fabricated in a 12nm FinFET NVIDIA (FFN) technology and have 640 tensor cores split across 80 Streaming Multiprocessors (SMs). On both GPUs, the user can easily enable and disable the use of tensor cores. Each Volta SM has 32 double precision cores, 64 single precision floating point (Single) cores and each single-precision core can perform two half-precision floating point (Half) operations at the same time. Overall, the GPU can execute in parallel 5,120 operations in single precision or 2,560 operations in double precision. The register files, shared memory, L1 and L2 caches, and main memory are ECC protected on the V100.

Novel GPUs, such as NVIDIA Pascal, Volta, Turing and, most likely all future architectures, have High Bandwidth Memory (HBM2) memories stacked on top of the chip die. When performing radiation experiments, these memories are irradiated with the underlying chip. While HBM2 reliability is out of the scope of this paper, we can say that HBM2 seems very susceptible to neutrons, showing permanent and intermittent errors after a few seconds of beam time. Moreover, Double Data Rate (DDR) error rate is known to increase significantly with temperature [11]. As a result, when the 300W GPU is executing a code the number of errors exploded. To have a more detailed reliability evaluation of tensor core and mixed-precision we need to test the GPU both with ECC on (focusing on faults in logic resources and unprotected resources) and with ECC off (to understand the contribution of memory errors). When ECC is enabled HBM2 is protected, and we have not seen any uncorrectable error during our experiments. When ECC is disabled, we need to engineer a solution to avoid errors in HBM2 to bias our results. We decide to triplicate and vote data in HBM2. While the voting process slows down the access to HBM2, we ensure that the computation inside the GPU core remains unaltered. The HBM2 triplication should, then, not affect the measured error rates.

We run $8k \times 8k$ Matrix Multiplication (**MxM**) to evaluate the reliability of tensor core and mixed-precision. We implement MxM both in software (as a sequence of ADD, MUL, and control loop variables) and using tensor core. Both implementations follow NVIDIA CUDA examples to take full advantage of GPU performances. We limit our analysis to MxM as it is the only code that takes full advantage of tensor core. When executed in half, the execution time of our MxM implemented with tensor core is less than half than the software one. Because of the casting from float to half, the float MxM with tensor core execution time is 60% the software MxM. The novel architectural improvements (and our reliability evaluation) have been dedicated to MxM for its importance in both neural network and HPC applications execution. It is worth noting that the triplication of data in the HBM2 when ECC is off required some code modifications, eventually reducing the memory access performances. While the algorithm and operations are the same, we highlight that with ECC on MxM is an optimized version of a matrix multiplication, also named General Matrix Multiply (GEMM) while with ECC off MxM has lower performance.

Experiments were performed at the Los Alamos Neutron Science Center (LANSCE) facility, Los Alamos, New Mexico and at the ChipIR facility, Didcot, United Kingdom. The available neutron flux was approximately $2.5 \times 10^6 neutrons/(cm^2/s)$.

Figure 2 shows a portion of the experimental setup mounted at ChipIR. A host computer initializes the test, sending a pre-selected input to the GPU and gathers results, comparing them with a pre-computed golden output. When a mismatch is detected, the execution is marked as affected by an SDC. The output data is then downloaded for post-processing. A software watchdog monitors a timestamp to identify application crashes. An ethernet controlled switch acts as a hardware watchdog and performs a power cycle of the system in the event of a hang.

IV. Experimental Results

In this section, we will present the results obtained from our radiation experiments. We compare the FIT rate of the various matrix multiplication implementations, we evaluate the effectiveness of ECC in reducing the error rate of GEMM and MxM algorithms with and without using tensor cores, evaluate the impact of faults in the output, and the Mean Executions Between Failure (MEBF).

A. Failure in Time

Figures 3 and 4 show and tables I and II list the normalized FIT rates for SDCs and DUEs of the Titan V and V100. To normalize data we have divided all reported values by the lowest FIT rate measured (DUE of tensor core executed in half for the ECC OFF version. FIT rates then are shown in arbitrary units (a.u.) to allow comparison across devices

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2020.2977583, IEEE Transactions on Nuclear Science

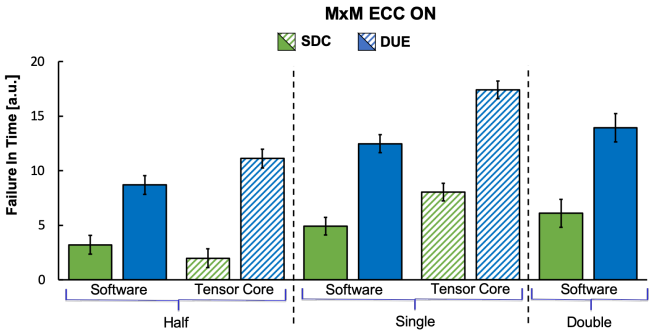

Fig. 3. Normalized FIT rates on optimized MxM algorithm with ECC enabled.

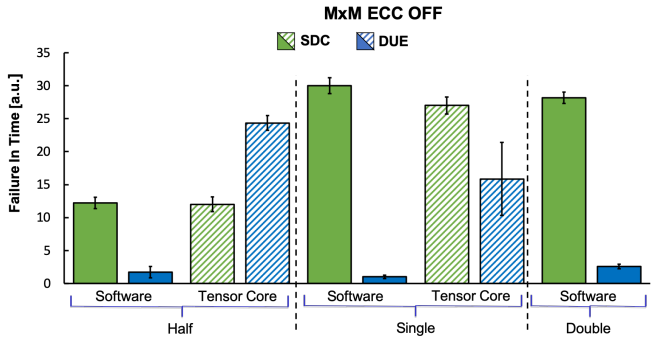

Fig. 4. Normalized FIT rates on MxM algorithm with ECC disabled.

without revealing business sensitive data. Data is shown with 95% confidence intervals, according to Poisson distribution.

The main insights we want to highlight and discuss from our data are: (1) the reliability of MxM implemented in software versus using tensor core; (2) the influence of data precision on the reliability of software and tensor core MxM (double precision is not supported by tensor core); (3) the impact of ECC in both (1) and (2).

From Figures 3 and 4 it is clear that, while effective in lowering SDC rate, ECC increases the DUE FIT rate. This is in accordance with previous studies that show that the GPU's ECC triggers an application crash when an unrecoverable double bit error is detected [12]. We can also notice that, independently on precision, the use of tensor cores increases DUE FIT. The DUE increase caused by the use of tensor core is exacerbated when ECC is turned off, suggesting that most of the faults causing DUEs in tensor core comes from errors in memory elements. It is worth noting that fewer accesses to HBM2 are performed on a tensor core implemented MxM versus a software implemented MxM. Accesses to an external device, that have been proved to cause DUE [13], are not the reason for the increased DUE FIT when tensor cores are used. Our data suggests that the corruption of internal resources of the tensor core is, then, particularly prone to generate a DUE.

TABLE I
NORMALIZED FIT RATE FOR MXM EXECUTED WITH ECC ENABLED

Precision	Version	FIT SDC [a.u.]	FIT DUE [a.u.]
Half	Tensor	3.21 ± 0.85	8.69 ± 0.85
	Software	1.98 ± 0.87	11.10 ± 0.87
Float	Tensor	4.81 ± 0.82	12.46 ± 0.82
	Software	8.02 ± 0.80	17.41 ± 0.80
Double	Software	6.09 ± 1.29	13.93 ± 1.29

TABLE II
NORMALIZED FIT RATE FOR MXM EXECUTED WITH ECC DISABLED

Precision	Version	FIT SDC [a.u.]	FIT DUE [a.u.]
Half	Tensor	12.21 ± 0.85	1.71 ± 0.85
	Software	12.00 ± 1.13	24.34 ± 1.13
Float	Tensor	30.00 ± 1.22	1 ± 0.25
	Software	26.99 ± 1.31	15.85 ± 5.51
Double	Software	28.18 ± 0.87	2.56 ± 0.34

Figure 3 shows that, when ECC is on, tensor cores MxM has a lower FIT rate when executed in half precision. This attests that the tensor core circuit is slightly more reliable than the combination of ADD, MUL, and the loop control variables needed to implement MxM in software. When MxM is executed in single precision, the tensor core SDC FIT rate increases significantly, being 20% higher than the software MxM one. This is only in apparent contrast with the claim that tensor core circuit is more reliable than the combination of software MxM. In fact, as stated in Section II, tensor core executes physical operations only in half precision. Single precision inputs require a hardware casting to half precision, that increases the GPU's occupation and, thus, the SDCs (and DUEs) FIT rate.

As shown in Figure 3 , when executed in software with ECC on, MxM FIT rate increases as the precision increases. It is worth noting that the FIT increase from single to double is less remarkable (about 10%) than the increase from half to single (about $2\times$). This is due to the fact that Volta architecture has 5,120 cores for single and half precision operations but only 2,560 cores for double precision operations. The number of parallel resources involved in the computation as a 64 bits MxM is $1/2$ the number of resources involved in the computation of a 32 bits MxM. While the FIT rate of a double operation is probably higher than the FIT rate of a single precision operation, then, overall the GPU is executing fewer 64 bits operations than 32 bits operations. Actually, knowing from Figure 3 that MxM FIT rates for double and single precision is almost the same and that there are twice as many 32 bits cores than 64 bits cores, we can deduce that the FIT rate of one double precision core is about $2\times$ the FIT rate of a single precision core.

Disabling ECC, as shown in Figure 4, increases significantly (at least $5\times$) the SDC rate, for all data precisions and for both MxM implementations. When ECC is disabled, errors in caches and register file can impact MxM executions. As caches and registers are the biggest and most vulnerable resources (we recall that data in HBM2 is triplicated), when ECC is off they are expected to dominate the GPU error rate [12].

From Figure 4 we can see that when ECC is turned off the software and tensor core MxM implementations have similar FIT rates. Moreover, the SDC rate of half precision is about $1/2$ the SDC rate of single precision. Both the tensor core and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2020.2977583, IEEE Transactions on Nuclear Science

software implementation of MxM multiplies matrixes of the same size and, then, uses the same amount of data in caches and registers (thus their error rate is similar when ECC is off). Moreover, the size of 16 bits matrixes is $1/2$ the size of 32 bits matrixes, justifying the FIT rate of single vs. half precision. Finally, double and single precision MxM have similar SDC FIT rates. As mentioned, there are 5,120 cores for single precision and 2,560 cores for double precision operations. In a given instant, then, the GPU maintains in caches and registers $2\times$ more single precision elements than double precision elements. As a 64 bits element has $2\times$ the size of a 32 bits element the overall memory footprint (and, then, the FIT rate when ECC is off) will be the same for double and single precision MxM.

B. Impact of Faults in the Output Values

An additional aspect of a GPU's reliability, that can be gleaned from inspecting the output values of neutron beam experiments, is the degree of mismatch in the output values (i.e., how different the corrupted output is from the expected one). Small variations in an output value could have little or no impact at all in terms of CNN detection or classification (e.g., small changes in the shape of the object would not be considered critical, or small probability variations). Here we evaluate the impact of faults on matrix multiplication's output correctness by measuring how the FIT rate is reduced when we increase the Tolerated Relative Error (TRE). A TRE of 0% considers any mismatch between the corrupted output and the expected value as a critical error. We then relax the correctness constraint accepting (corrupted) values in a given range as tolerable (corrected). As an example, if we consider a TRE of 10%, any outputs value between 90% and 100% of the expected value will be considered as correct or, at least, tolerable.

Figure 5 and 6 illustrate how varying the acceptable error margins (the TRE) affects the SDC FIT rate of software and tensor MxM versions, respectively. We plot how much the FIT rate changes (vertical axis) when we increase the TRE from 0% up to 10% (horizontal axis). The different FIT rate reductions are symptomatic of how physical-level faults propagate through the GPU microarchitecture.

From Figure 5 we can see that the increasing of the acceptable error margin and the precision reduces the FIT rate. This is in accordance with previous studies that show that lower precision leads higher difference between expected and corrupted values [14]. For instance, a 5% TRE would reduce of 40% the FIT rate of the double precision matrix multiplication, but only of 25% the float one and of 20% the half one. The fact that the double precision version has a higher FIT reduction compared to single and half means that the corrupted outputs are closer to the expected values for the double precision.

For the tensor cores implementation, Figure 6 shows that the acceptance of the error margin only slightly decreases the FIT rate for single precision and remains basically constant for the half precision. The very high impact of faults in the output correctness of tensor core is justified by the fact that tensor core is a huge core whose output is the result matrix.

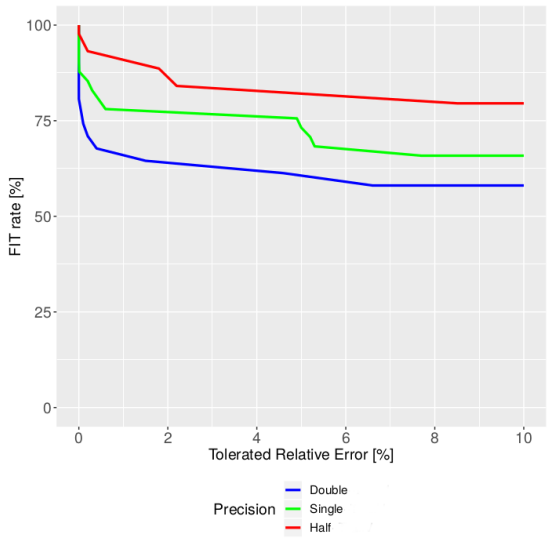

Fig. 5. FIT reduction as a function of relative error tolerance for the three precisions of matrix multiplication.

In other words, the atomic operation a tensor core executes is matrix multiplication (e.g., operations are performed in a specific hardware). On the contrary, the software implemented matrix multiplication uses a sequence of atomic ADD and MUL operations. For the software implemented version, a fault in an ADD or MUL can significantly impact the atomic operation output but then this value is processed with several others while propagating to the output. We believe that an error during the execution of a tensor core may cause a corruption that spreads in all parts of the core, highly affecting the result.

The shown results on the impact of faults in the output correctness of matrix multiplication should be considered when choosing which implementation to adopt in the execution of CNNs. As shown in [10], in fact, the faults affecting matrix multiplications are the ones that are more likely to impact the object detection. A higher difference between the corrupted output and the expected output is more likely to lead the CNN to wrongly detect or classify objects.

C. Mean Executions Between Failure

As tensor cores were introduced to improve MxM performances, while the FIT rate of MxM in software vs. tensor core are similar, the tensor core implementation could be more reliable in the sense that it could complete more correct executions before experiencing a failure. To consider also execution time in our reliability evaluation we consider, in Figure 7, the MEBF of MxM executed with ECC on. MEBF is the number of correct executions correctly completed before experiencing a failure (the higher, the better). MEBF is calculated dividing the Mean Time Between Failures ($1/FIT$) by the execution time.

As shown in Figure 7, tensor cores allow to complete $2\times$ as many half precision MxM executions as the software implementation. When MxM is executed in single precision the benefit of the tensor core in the MEBF is attenuated by the casting of input that increases both the FIT rate and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2020.2977583, IEEE Transactions on Nuclear Science

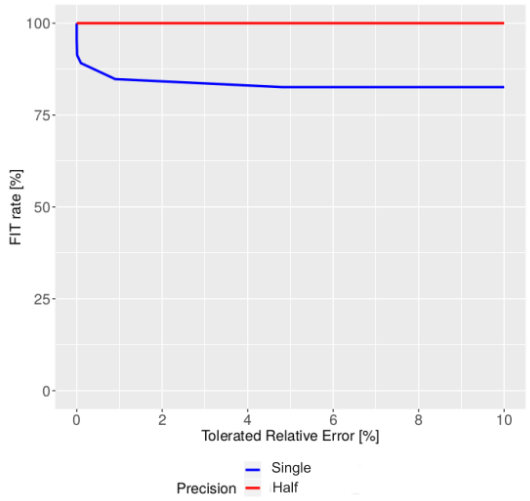

Fig. 6. FIT reduction as a function of relative error tolerance for the tensor core.

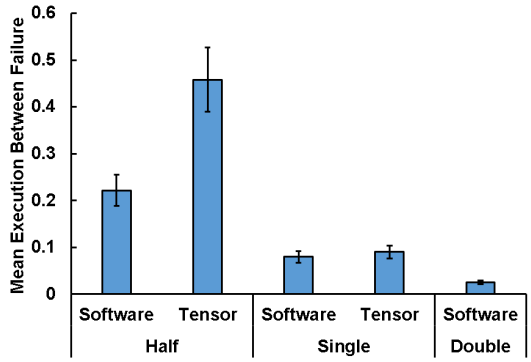

Fig. 7. Mean Execution Between Failures of MxM ECC.

execution time. Finally, as we increase precision from 16, 32, and 64 bits the MEBF of software MxM is gradually reduced. The combination of higher FIT rates and lower performances of executing operations with higher precision, then, have the drawback of reducing significantly the GPU reliability.

It is worth noting that different data and operation precision, besides the shown impact on SDC and DUE FIT rates, can also impact the criticality of errors. An error in a half precision element, in fact, can generate a corrupted value much more different from the expected value than an error in a double precision element.

V. Conclusion

We have presented a reliability analysis of Matrix Multiplication, implemented using tensor cores and in software, executed with different precision on Volta GPUs. We have seen that tensor cores slightly increases the GPU FIT rate for both SDC and DUE. However, thanks to the improved performances, tensor cores allow the GPU to complete a significantly higher number of correct matrix multiplication executions than the software implementation of matrix multiplication. Additionally, we have seen that reducing precision has the benefit of improving not only the GPU performances and efficiency but also reliability. Finally, the impact of faults in the output correctness strongly depends on the implementation of matrix multiplication (i.e., software or using tensor core) as well as on the precision.

References

[1] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[2] NVIDIA, "NVIDIA Tesla V100 GPU Architecture - Whitepaper," NVIDIA, Tech. Rep. 1.1, aug 2017.

[3] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating Deep Convolutional Networks Using Low-precision and Sparsity," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2861–2865.

[4] D. A. G. de Oliveira, L. L. Pilla, T. Santini, and P. Rech, "Evaluation and mitigation of radiation-induced soft errors in graphics processing units," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 791–804, March 2016.

[5] P. Rech, T. Fairbanks, H. Quinn, and L. Carro, "Threads Distribution Effects on Graphics Processing Units Neutron Sensitivity," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 6, pp. 4220–4225, Dec 2013.

[6] C. Lomont, "Introduction to Intel Advanced Vector Extensions," *Intel White Paper*, pp. 1–21, 2011.

[7] M. Clark, R. Babich, K. Barros, R. Brower, and C. Rebbi, "Solving Lattice QCD Systems of Equations Using Mixed Precision Solvers on GPUs," *Computer Physics Communications*, vol. 181, no. 9, pp. 1517 – 1528, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465510001426

[8] N. Ho, E. Manogaran, W. Wong, and A. Anoosheh, "Efficient Ploating Point Precision Tuning for Approximate Computing," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2017, pp. 63–68.

[9] Z. Jia, M. Maggioni, B. Staiger, and D. P. Scarpazza, "Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking," *CoRR*, vol. abs/1804.06826, 2018. [Online]. Available: http://arxiv.org/abs/1804.06826

[10] F. F. d. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, June 2019.

[11] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: A Large-Scale Field Study," in *SIGMETRICS*, 2009.

[12] C. Lunardi, F. Previlon, D. Kaeli, and P. Rech, "On the Efficacy of ECC and the Benefits of FinFET Transistor Layout for GPU Reliability," *IEEE Transactions on Nuclear Science*, vol. 65, no. 8, pp. 1843–1850, Aug 2018.

[13] V. Fratin, D. Oliveira, C. Lunardi, F. , G. Rodrigues, and P. Rech, "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.

[14] F. Fernandes dos Santos, C. Lunardi, D. Oliveira, F. Libano, and P. Rech, "Reliability Evaluation of Mixed-Precision Architectures," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2019, pp. 238–249.