Analyzing the Soft Error Reliability of Convolutional Neural Networks on Graphics Processing Unit

Khalid Adam¹, Izzeldin I. Mohd¹, Younis Ibrahim²

¹College of Engineering, University Malaysia Pahang, Malaysia
²College of IoT Engineering, Hohai University, China

Email: salimm@ieee.org*

Abstract. There has been extensive use of Convolutional Neural Networks (CNNs) in safety-critical applications. Presently, GPUs are the most prominent and dominated DNN accelerators to increase the execution speed of CNN models to improve their performance as well as the Latency. However, GPUs are prone to soft errors. These errors can impact the behaviors of the GPU dramatically. Thus, the generated fault may corrupt data values or logic operations and cause errors, such as Silent Data Corruption (SDC). Unfortunately, soft errors propagate from the physical level (GPUs) to the application level (CNN model). This paper analyzes the reliability of the AlexNet model to identify which part of the model more vulnerable to the soft error. To achieve this, we injected the AlexNet run on top of NVIDIA’s GPU, using the SASSIFI fault injector as the major evaluator tool. The experiments demonstrate a high reduction from 9.3 % to 0.00% SDCs errors in STORE and 5.0 % to 0.00% SDCs errors in GPR in Im2col. While Add bias kernel instructions STORE and GPR the errors reduced from 0.8 % to 0.00% and 1.2 % to 0.1% SDCs error respectively.

1. Introduction

Deep Neural Networks (DNNs) have been adopted in a variety of applications [1], such as computer vision, speech recognition, natural language processing, and others [2]. Computer vision, in turn, has plenty of applications ranging from entertainment to extremely safety-critical applications, for instance, healthcare application. A large number of specialized accelerators have been proposed to accelerate the execution of DNN models to improve their performance as well as the Latency. However, GPUs have been the dominant platform [3] [4] [5] [6], not only because it was the first co-processor has been invented to offload the computation-intensive from CPU, but also due to the computation power of the GPUs massively parallel architecture, and wide-range support of floating points, which perfectly match the requirements of DNN models use in safety-critical applications. This, in turn, has led to the importance of GPUs reliability.

However, one of the major sources of unreliability in GPUs is soft errors, typically caused by high-energy particles striking electronic devices and causing them misclassification (e.g., flip a single bit) [7][8]. These errors are able to impact the behaviors of the GPU dramatically. By generated fault may corrupt data values or logic operations and cause errors, such as Silent Data Corruption (SDC), application crash or system hang (i.e., Detected Unrecoverable Error, DUE), but it may also be Masked and cause no observable error [9]. Unfortunately, soft errors propagate from GPUs to the DNN model run on top of them and can eventually lead to misclassification of objects in DNN models [10], and the consequences would be disastrous as shown in Figure 1. For instance, Food and Drug Administration (FDA) reported that 1078 of the adverse events (10.1%) were unintended errors (soft
errors) that happened, including 52 injuries and two deaths. Therefore, becomes essential to understand the behaviour of these DNN models in the presence of GPUs faults [11].

Although many studies have been conducted on the performance and accuracy issues of DNN accelerators and applications, such as [12] [13], there have been relatively few works that addressed the reliability problems of DNN models through DNN accelerators (e.g., GPUs). Recent Nvidia GPUs [14] for instance, support Error-Correcting Code (ECC) to protect memory elements (i.e., VRAM, shared memory, L2 & L1 caches, and register files) from transient faults. Transient hardware faults, however, still can occur in the functional units, and then can propagate to memory elements. It can be concluded that transient-hardware faults still could occur. Evaluating and analyzing their resilience to soft errors caused by high-energy particle strikes has become a must. In this paper, the reliability of the AlexNet model on a GPU was analyzed by conducting series of fault-injection campaigns, using NVIDIA's SASSIFI. The first significant contribution of this paper is the determination of soft error reliability in AlexNet through comprehensive analysis and ranking of vulnerable model parts from the perspective of kernel. The second contribution is reduction of soft errors through selectively hardening only the vulnerable of the AlexNet model. The subsequent sections of this paper are including are section 2 literature review presents the related works. Section 3 contains the methodology. The experimental results and their analysis are presented in section 4. Finally, section 5 conclusion.

2. Literature Review

There are several studies [15][11][16][17][18] that evaluated and analyzed the reliability of CNN models. Hence, it has been established that there are varieties of CNN architectures, with each having different behavior and workflows. The different CNNs have been are implemented on various accelerators including GPUs, ASICs, and TPU, through their peculiar execution flow based on their varying components. This makes it difficult to directly generalize the case of a particular CNN to other architectures [10]. Notably, several approaches have been developed to reduce the occurrence of the soft error in GPUs, through software solutions. This includes Double Modular Redundancy (DMR) and Triple Modular Redundancy (TMR). However, the major drawback to the use of these solutions is the runtime overheads.

3. Methodology

A Maxwell architecture-based GPU GTX 750 Ti was used for this purpose, with a SASSIFI fault injector which was primarily used to assess the reliability of AlexNet model runs on GPUs. This was achieved through fault injection campaigns which made it easier to determine the possibility of a low-level corruption to propagate to the output. With this tool, it was possible to carry out fault injection through Instruction Output Address (IOA) mode to evaluate the Program Vulnerability Factor (PVF). In addition, used to investigate how a single error modifies the result of instruction and propagates to the program output (AlexNet). A total of 1000 faults was injected for IOA, this number of injections
was enough to guarantee that the worst-case statistical error bars at 95% confidence are at 1.96%.
Notably, various bit-flip models can be obtained from SASSIFI including zero value, single bit-flip, a
random value, and multiple bit-flip. Nevertheless, only the single bit-flip it selected for injection IOA
of this present study. We selected the single-bit flip because more suitable and realistic for memory
effects.

4. Result and Discussion
As a consequence of fault injection and comparison of program output with the golden output (i.e.,
the pure outcome), three categories are expected, Masked, DUE, or SDC. It should be noted that SDC
is the only error of interest to this study when studying the error propagation within the model. This is
because crash and hand errors (DUEs) are not being propagated to a subsequent layer. Similarly,
masked errors are instantaneously masked at the location of occurrence. The SDC errors and the
mechanism of their propagation through layers. Figure 2 presents the two most vulnerable kernels
(Im2col, Add-bias) and each instruction’s contribution percentage in the AlexNet model (using the
IOA mode). This should not be confused with the vulnerability analysis of the kernel or instruction
alone. Instead, we merge them together to make sense of the most vulnerable kernels and what
instructions are causing this vulnerability. First, the STORE and GPR instructions contribute 9.25%
and 4.95% IVF SDC (Malfunction and Light-Malfunction) errors to the Im2col kernel’s vulnerability,
whereas they generate 11.35% and 9.30% IVF No-Malfunction, respectively. Second, the same
instructions, STORE and GPR, contribute 0.75% and 1.15% IVF SDC (Malfunction and Light-
Malfunction) errors to the Add_bias kernel’s vulnerability, respectively, and they generate No-
Malfunction of 2.60 % and 10.55% IVF. These two points reflect the importance of protecting these
kernels.

![Figure 2](image_url)

Figure 2. The top vulnerable and resilient kernels and their corresponding instruction
groups for AlexNet (with IOA mode).

Figure 3 exhibits the instructions group in kernels after we protected (by duplicated them used TMR),
only the vulnerable kernels presented in Figure 2. Thus, our result demonstrates that STORE
instructions in both kernels produced 0.00% SDCs (Malfunction and Light-Malfunction) errors, and
GPR instructions yielded 0.00% and 0.10% SDCs (both) errors on Im2col and Add-bias, respectively.
Thus, STORE and GPR instructions were significantly improved, where No-Malfunction SDCs
increased to 26.7% and 18.1% in Im2col and 5.9% and 14.8% in Add-bias, respectively.
Figure 3. AlexNet’s kernels and their corresponding instruction groups after applying the proposed mitigation solution (IOA mode).

To conclude, since kernels are basically created from low-level instructions, within each kernel we can identify which instructions in our models are likely to generate more errors and directly contribute to the kernel’s vulnerability. Besides, GPU designers can properly select the instructions with higher Instruction Vulnerability Factor (IVF) to duplicate them in order to mitigate SDCs or DUE errors. For DUE errors, we suggest GPU architects who design specialized GPUs in deep learning to design watchdog circuits integrated into the GPU to detect crashes and to promote the reliability of DNN models in safety-critical applications.

5. Conclusion

In this paper, we analyzed and evaluated the SDCs of soft errors for the AlexNet model on the GPU. We identify the most vulnerable kernels, by the analyzed the reliability of the model’s bit sensitivity, the vulnerable bits can be selectively protected. Our result shows a high reduction rate of errors in the top vulnerable kernels (Im2col and Add_bias). Besides, the model achieved high reduction in masked from 9.3 % to 0.00% SDCs errors in STORE and 5.0 % to 0.00% SDCs errors in GPR in Im2col. While Add_bias kernel instructions STORE and GPR the errors reduced from 0.8 % to 0.00% and 1.2 % to 0.1% SDCs error respectively. Moreover, the performance overhead of our solution is low compared with the well-known protection techniques such as the Double Modular Redundancy (DMR), and Triple Modular Redundancy (TMR), because we protected only the venerable part of the model instead of whole the model.

Acknowledgement

This research was supported by Universiti Malaysia Pahang, through the UMP internal grant (RDU1903149)

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, ‘A survey of deep neural network architectures and their applications’, Neurocomputing, vol. 234, pp. 11–26, 2017.
[2] D. Yu and L. Deng, ‘Deep learning and its applications to signal and information processing [exploratory dsp]’, IEEE Signal Processing Magazine, vol. 28, no. 1, pp. 145–154, 2010.
[3] J. Tang et al., ‘Bridging Biological and Artificial Neural Networks with Emerging Neuromorphic Devices: Fundamentals, Progress, and Challenges’, Advanced Materials, 2019.
[4] M. Nazemi, G. Pasandi, and M. Pedram, ‘NullaNet: Training deep neural networks for reduced-memory-access inference’, arXiv preprint arXiv:1807.08716, 2018.
[5] T. V. Huynh, ‘Deep neural network accelerator based on FPGA’, in 2017 4th NAFOSTED Conference on Information and Computer Science, 2017, pp. 254–257.
[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, ‘Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks’, IEEE Journal of Solid-State Circuits, vol. 52, no. 1, pp. 127–138, 2016.

[7] Y. Ibrahim and H. Wang, ‘Analyzing the Reliability of Convolutional Neural Networks on GPUs: GoogLeNet as a Case Study’.

[8] N. Jouppi, ‘Google supercharges machine learning tasks with TPU custom chip. 18 May 2016’. 2019.

[9] Y. Ibrahim et al., ‘Microelectronics Reliability Soft errors in DNN accelerators: A comprehensive review DNN (Predictive) Model Input Label = Truck Soft Error (Bit Flip) Global Failure Soft Error Fault Error’, Microelectronics Reliability, vol. 115, no. September, p. 113969, 2020.

[10] Y. Ibrahim et al., ‘Soft Error Resilience of Deep Residual Networks for Object Recognition’, IEEE Access, vol. 8, pp. 19490–19503, 2020, doi: 10.1109/ACCESS.2020.2968129.

[11] 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, SC 2017, 2017, p. 8, doi: 10.1145/3126908.3126964.

[12] E. O. Aboagye, G. C. James, and R. Kumar, ‘Evaluating The Performance Of Deep Neural Networks For Health Decision Making’, Procedia computer science, vol. 131, pp. 866–872, 2018.

[13] M. G. Amin and B. Erol, ‘Understanding deep neural networks performance for radar-based human motion recognition’, in 2018 IEEE Radar Conference (RadarConf18), 2018, pp. 1461–1465.

[14] P. N. Glaskowsky, ‘NVIDIA’s Fermi: the first complete GPU computing architecture’, White paper, vol. 18, 2009.

[15] F. F. Dos 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, doi: 10.1109/TR.2018.2878387.

[16] L. Weigel, F. Fernandes, P. Navaux, and P. Rech, ‘Kernel vulnerability factor and efficient hardening for histogram of oriented gradients’, in 2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2017, pp. 1–6.

[17] C. Lunardi, F. Prevolon, 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, 2018.

[18] F. F. 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 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2017, pp. 169–176.