International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 5, October 2021, pp. 4481~4488
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i5.pp4481-4488

Analyzing the instructions vulnerability of dense convolutional network on GPUS

Khalid Adam[1], Izzeldin I. Mohd[2], Younis Ibrahim[3]
[1,2]College of Engineering, University Malaysia Pahang, Malaysia
[3]College of IoT Engineering, Hohai University, China

Article Info

Article history:

Received Sep 4, 2020
Revised Mar 10, 2021
Accepted Mar 21, 2021

Keywords:

DenseNet201
GPUs
Healthcare
Reliability
Soft error

ABSTRACT

Recently, deep neural networks (DNNs) have been increasingly deployed in various healthcare applications, which are considered safety-critical applications. Thus, the reliability of these DNN models should be remarkably high, because even a small error in healthcare applications can lead to injury or death. Due to the high computations of the DNN models, DNNs are often executed on the graphics processing units (GPUs). However, the GPUs have been reportedly impacted by soft errors, which are extremely serious issues in the healthcare applications. In this paper, we show how the fault injection can provide a deeper understanding of DenseNet201 model instructions vulnerability on the GPU. Then, we analyze vulnerable instructions of the DenseNet201 on the GPU. Our results show that the most significant vulnerable instructions against soft errors PR, STORE, FADD, FFMA, SETP and LD can be reduced from 4.42% to 0.14% of injected faults, after we applied our mitigation strategy.

Corresponding Author:

Khalid Adam
Faculty of Electrical and Electronic Engineering Technology
University Malaysia Pahang
26300, Pahang, Malaysia
Email: khalidwsn15@gmail.com

1. INTRODUCTION

Recently, the success of deep neural network (DNN) in challenging perception tasks makes them a powerful tool for many applications, including safety critical system such as healthcare application [1]. For instance, DNN is used in surgical procedures where it gives a better understanding of surgical practices and automatic recognition of workflow [2], [3]. This and other potential DNN applications are associated with high risk of humam injury or death, especially if there is a malfunction. Hence, it should be exceptionally reliable [4]-[6]. However, with the growing complexity of modern digital hardware platforms (i.e., GPUs), it has become increasingly difficult to guarantee the reliability of hardware operations when DNN models are run on top of the hardware (i.e., GPUs) [7]-[9]. Notably, soft errors are caused by a transient signal. This is induced by a single energetic particle strike when the collected charge is greater than the critical charge required to cause a change in the state of a memory cell, register, latch, or flip-flops [10], [11]. As a result, this could eventually lead to misclassification of objects in DNNs, and the consequences would be disastrous. Therefore, when the DNNs is performed in GPUs, their reliability implication is not well understood in the healthcare applications, because the errors propagate from the GPUs to DNNs (i.e., DenseNet201) [7], [12].

Several techniques have been proposed to reduce the soft errors in the GPUs such as double modular redundancy (DMR), triple modular redundancy (TMR), and algorithm-based fault tolerance (ABFT) [13], [14]. Nevertheless, the main issue with these solutions is that they have runtime overhead and are not cost

effective. To address the above issues, we identified the soft errors propagated in the DNNs to mitigate the soft errors. To achieve this, we propose an analysis methodology to inject the DenseNet201 to identify the vulnerable instructions to soft errors. As the first aim, we proposed mitigation strategy to reduce the soft errors, and we implement it on DenseNet201.

The main contributions of this work are; i) A methodology to evaluate the probability of fault in specific parts of the source code due to errors at the output; ii) Experimental evaluation of the behavior of DenseNet201 executed on GPUs exposed to soft errors; iii) Identification of the most vulnerable instructions of the DenseNet201 through fault injection; iv) Concretely propose mitigation strategies on how to increase DenseNet201 instructions reliability for healthcare applications; v) Validation of DenseNet201 mitigation strategies through SASSIFI fault injection. This paper is structured as follows: Previous studies are presented in section 2, while our proposed strategies are presented in section 3. Section 4 contains the method, while section 5 is the analyses and explanation of findings. Furthermore, evaluation of strategies is presented in section 6, after which the conclusion is presented in section 7.

2. PREVIOUS WORKS

Several studies have been conducted on the performance and accuracy issues of DNN accelerators and healthcare applications. Jie *et al.* [15] proposes a novel method to predict heart diseases from electrocardiogram (ECG) signals with cardiology, signal processing methods and deep learning model (ResNet-34). But the authors did not consider the reliability of such model for the intended application, which is actually a safety-critical application, based on real-time convolutional neural network (CNN) model detection. Keno *et al.* [16] compared 15 different CNNs of five different architectures (ResNet, DenseNet, VGG, SqueezeNet, Inception v4, and AlexNet) on two chest radiograph classification datasets. This was done with the PyTorch and Fast AI libraries on a workstation, running on Ubuntu 18.04 with two Nvidia GeForce RTX 2080ti, and all training was done using the Python programming language. However, the authors did not consider the reliability of such models for the intended application. In another study by Zhiwen *et al.* [17], a lightweight hybrid neural network (DenseNet) for medical image classification was proposed. This consisted of a modified PCANet, cascaded with a simplified DenseNet. The updated PCANet has two phases in which at each point, the network produces successful feature maps by combining inputs with different learned kernels. With a small number of weights, the following simplified DenseNet will take all feature maps provided by the PCANet as inputs and use the dense shortcut links to achieve accurate classification of medical images. Experimental results showed that the proposed hybrid neural network is simple to train, and in terms of classification accuracy, sensitivity, and specificity, it outperforms conventional CNN models such as PCANet, ResNet and DenseNet. Both networks were made for acceleration with Python, based on TensorFlow 1.9.0 and Keras 2.2.4 on Ubuntu 16.04, and are run on the NVIDIA GTX 1080Ti GPU with CUDA 10.1. Nonetheless, the reliability of the models was not been considered.

Wenping *et al.* [18], conducted a study on the classification of five major trends of interstitial lung disease in healthy tissue, emphysema, ground glass, fibrosis and micronodules. The proper diagnosis of interstitial lung disease is helpful in improving the effect of treatment on patients. The paper introduces an improved DenseNet called the DenseNet small kernel (SK-DenseNet), to improve the efficiency of interstitial lung disease classification. The results of the experiment showed that the proposed SK-DenseNet achieves outstanding performance (~98.4 per cent), which increases its performance by 5 % compared to DenseNet. The proposed approach has been implemented using the Tensorflow framework and encoded in the Python programming language. All experiments were conducted on GPU NVIDIA under the Linux OS. However, the authors did not consider the reliability of such model for the intended use. Ahmet *et al.* [19] intended to diagnose brain tumours using a CNN model (Resnet 50). The last 5 layers of the Resnet50 model were removed, and 8 new layers were added. With this model, the accuracy value of 97.2 percent was obtained. The findings are also consistent with those obtained with the Alexnet, Resnet50, Densenet201, InceptionV3 and Googlenet models. Among these models, the most effective model was used to categorisze brain tumour images. The experiment was carried out in the MATLAB environment with GPU card. However, the authors did not consider the reliability of such model for safety-critical application applications.

There have been relatively few reports that addressed the reliability problems of DNNs through DNN accelerators (e.g. GPUs) [20], [21]. One of the major sources of unreliability in modern systems is soft errors [22]. This is typically caused by high-energy particles, striking electronic devices, and causing them to malfunction (e.g., a single bit flip) [23]. Therefore, most of the traditional applications that are based-on GPUs are fault tolerant. More recently however, GPUs are widely used to accelerate safety critical applications that are dominated by DNNs models as shown in Table 1. Therefore, it becomes essential to understand the behaviour of these applications in the presence of hardware faults [24]. A few studies have

evaluated the reliability of DNNs models on GPUs. Specifically, DNN models such as ResNet, VGGNet, and GoogLeNet have been examined [10], [25]. However, to the best of our knowledge, this is the first research to evaluate the soft error resilience of the DenseNet201 model from the perspective of vulnerability to instructions.

Table 1. The summary of the related work

Ref	Addressed Issues	Characteristics	Technology	Limitations
[15]	This study focused on predicting heart diseases from ECG signals with cardiology.	CNNs model architecture (ResNet-34).	Not mentioned	The authors did not consider reliability. Fault injection to evaluate reliability. CNN model DenseNet201 was used GPUs with logic units more sensitive to soft errors was used.
[16]	This study focused on classification of chest radiographs by comparing 15 different convolutional neural networks (CNNs)	CNNs models ResNet, DenseNet, VGG, SqueezeNet, Inception v4 and AlexNet	Nvidia GeForce RTX 2080ti	The authors did not consider reliability. Fault injection was used to evaluate the reliability of GPUs where the logic units more sensitive to soft errors
[17]	This study focused on medical image classification in the case of a small amount of training image data	CNNs model DenseNet	NVIDIA GTX 1080Ti GPU	The authors did not consider reliability. We used fault injection to evaluate the reliability of GPUs where the logic units more sensitive to soft errors.
[18]	This study focused on classification of five main patterns of interstitial lung disease healthy tissue, emphysema, ground glass, fibrosis, and micronodules	CNNs model DenseNet	GPU NVIDIA	The authors did not consider reliability.
[19]	This study aimed to diagnose the brain tumor using MRI images.	CNNs model ResNet50	GPU NVIDIA	The authors did not consider reliability. Fault injection to evaluate reliability by used CNN model DenseNet201

3. PROPOSED MITIGATION STRATEGY

In this section, we propose our mitigation strategy by identifying the most vulnerable instructions for DenseNet201, via fault injection (soft errors). In order to identify the most vulnerable instructions in the DenseNet201 model, we present a method (in section 4.1). The key concept of our mitigation strategy is that it is based on the well-known triple modular redundancy (TMR), and it intertwines three copies of the instructions and adds majority voting. In short, this strategy mitigation consists of triple instructions, by means of majority voters. Based on this concept, our mitigation strategy is a selective solution that protects only the vulnerable instructions instead of duplicating the whole as in TMR, to reduce the overheads, and thereby offering more flexibility to designers.

4. METHOD

4.1. Fault injection settings

Fault injection experiments are very reliable method to measure the soft errors (SDCs, Masked and DUEs). However, to correlate soft errors to a particular resource or code region, it is important to employ fault injection. In this study, we used NVIDIA GPU, SASSIFI "fault injector" and DenseNet201 (see section 4.2). We injected errors at the GPU to test the program vulnerability factor (PVF) of the DenseNet201, which is the likelihood that the performance of a program will be propagated by a single fault that modifies the outcome of instruction. Thus, to measure PVF of the program, we inject errors with; i) instruction output address (IOA) mode to study the probability that the address instruction has errors, and ii) instruction output value (IOV) mode to study the probability that the value executed instruction has errors. Then, we injected 1000 injections at IOA and IOV, but we are interested in faults that are not masked and make their way to the application (e.g., Masked SDC).

4.2. DENSENET201

Convolutional neural networks (CNNs) are used for all state-of-the-art vision tasks such as image recognition, object detection and localization, and segmentation. The latest new architecture is from Facebook AI research (FAIR), and it won the best paper at the most prestigious conference on computer vision in 2017: computer vision and pattern recognition (CVPR) [26]. Their architecture was called DenseNet, which implemented a new block called dense block and stacked these blocks on top of each other, with some layers in between, to create a deep network. These dense blocks took the idea of residual networks a step further and linked each layer to other. In other words, for a dense block, we consider all the dense blocks before it to be input, and we generate the output that we feed into all the subsequent dense blocks. We apply convolutions and batch normalizations to render layers consistent with each other. The advantage of

this is that we promote the reuse of features, overcome the gradient problem, and have less overall parameters. This arrangement ensures the optimal flow of information between all layers on the network, and directly links all layers (matching the size of the function map). In order to maintain the characteristics of forward spread, each layer receives additional feedback from all the layers ahead and transfers its own character mapping to all subsequent layers Figure 1. The L layer receives the feature-maps of all preceding layers, $X_0 \ldots\ldots\ldots X_{L-1}$ as input:

$$x_1 = H_1(x_0, x_1, \ldots\ldots\ldots, x_{l-1})$$

where $X_0 \ldots\ldots\ldots X_{L-1}$ refers to the concatenation of the features-maps produced in 0,.(l-1) layers. We use densenet201, which consists of 4 dense blocks of 201 layers in total. Each layer involves the application of a convolution philtre followed by ReLU, activation and line-wise batch normalisation. For each dense block, a growth rate of 6, 12, 48, and 32 features-maps were used. Details of the network structure are shown in Figure 1.

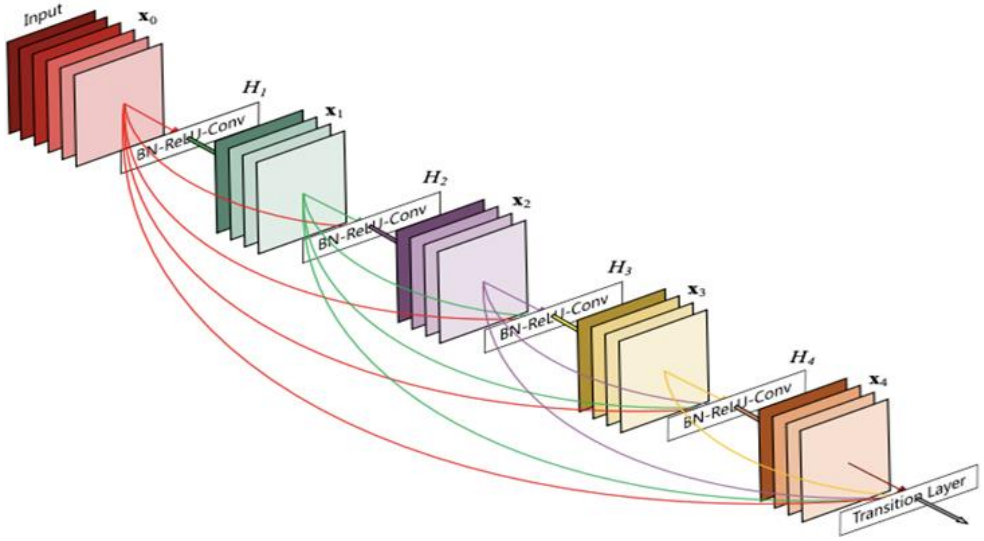

Figure 1. Five layers of a DenseNet block with a growth rate of 4 feature-maps per layer [26]

5. RESULTS AND ANALYSIS DISCUSSION

Based on the method discussed in section 4, in this section we show our result dissection by analysing the DenseNet201 instructions vulnerability.

5.1. Instruction group's vulnerability analysis

In this subsection, we evaluate the resilience of the DenseNet201 models from low-level instructions perspective. It is worth mentioning that this metric will be achieved by injecting errors into addresses and values of the executing instructions. Here, we intended to evaluate the sensitivity of various instructions by measuring the instruction vulnerability factor (IVF), which is the probability that a single soft error that modifies the result of an instruction will propagate to a program output. SASSIFI provides many predefined instruction groups, and based on our application (DenseNet201), we evaluate and analyze the IVF of the following instruction groups:

- Instructions that write to general-purpose registers (GPR).
- Instructions that write to condition-code registers (CC).
- Instructions that write to predicate registers (PR).
- Store instructions (STORE).
- Integer add and multiply instructions (IADD_IMUL_OP).
- Single-precision floating-point add and multiply instructions (FADD_FMUL_OP).
- Integer fused multiply and add instructions (MAD_OP).
- Single-precision floating-point fused multiply-add instructions (FFMA_OP).
- Load instructions (LD_OP).
- Register (SETP_OP).

We performed 1000 injections using the single bit-flip model in the IOA and IOV modes and show the results in Figures 2 and 3, respectively. We noticed that IOA mode is based on only two predefined instruction groups-GPR and STORE, which are enough to disturb the address of the destination register. On the other hand, IOV mode is based on many instruction groups. We can easily observe the amount of SDC and DUE errors that are produced by each instruction group in both injection modes (i.e. IOA and IOV).

In the Figure 2, injections into both instructions GPR and STORE were masked (32.65%) and (35.05%) respectively, while producing the same amount of SDCs (9.80%). In contrast, GPR producing DUEs (7.45%) and STORE producing (5.5%) DUE errors. Thus, both instructions seem to have the same vulnerability in IOA mode.

In Figure 3 starting PR, STORE, FADD, FFMA, SETP, and LD appears to be the most vulnerable instructions at all with average of SDCs 4.42%. This is an extremely high percentage, and it is unacceptable for safety-compliant systems. Injections with these six instructions PR, STORE, FADD, FFMA, SETP, and LD in DenseNet201 masked only (6.66%, 5.29%, 5.17%, 5.42, 6.83% and 3.97%) of the errors, respectively. On the other hand, PR and SETP produced small DUEs. These findings indicates that PR, STORE, FADD, FFMA, SETP in DenseNet201 are the most vulnerable instructions against soft errors. There are two probable reasons for this: i) as CNN models are basically built of many layers (i.e., computationally hungry), they load data (i.e., with LD instructions) as many times as the number of layers in the model; ii) most of the CNN architectures including DenseNet201 greatly relies on instructions (i.e., SETP DenseNet201) in their operations. These require the attention of researchers who work on GPU architecture designing or CNN model building to take the reliability issue into considerations. This is particularly important when the CNN model is intended to be used in some safety-critical environments. Meanwhile, GPR, IADD, and MAD instructions have moderate error resilience, where each of these instructions produced number of SDC errors on the average of 1.32% and masked 4.14%, 3.08% and 4.12% respectively. Injecting fault into the value (IOV) of the STORE instruction is not same as injecting into its address (IOA). Significantly, at least 4.71% of the injected IOV STORE faults became SDCs, while masking about 5.29%. However, it is surprising that no DUE has been generated. It is worth mentioning that CC instruction is a one-bit operations. In other words, they only perform toggle between zero and one, thus, we can only inject single-bit-flip faults. From a reliability perspective, every single fault has been injected to CC instruction was masked, therefore, no SDC or DUE has been observed.

Thus, we can identify the vulnerable instruction groups in DenseNet201, and some key observations include: i) in our application (DenseNet201), injecting errors in GPR and STORE in IOA mode often results in higher SDC and DUE probabilities when compared to respective IOV; ii) injecting errors in STORE in IOV does not result DUE errors; ii) injecting errors into CC registers did not have any effect on the program output; iv) PR, STORE, FADD, FFMA, SETP, and LD are the top-6 vulnerable instruction groups in models.

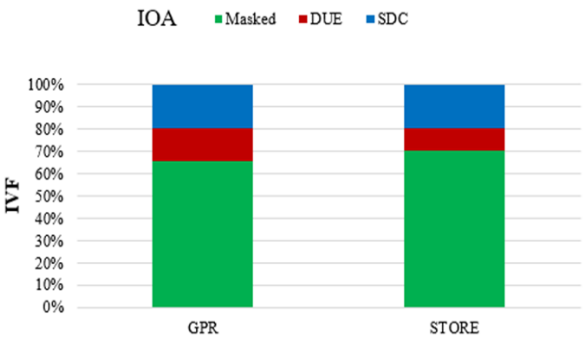

Figure 2. Outcomes of injections in instruction groups (IOA mode) DenseNet201

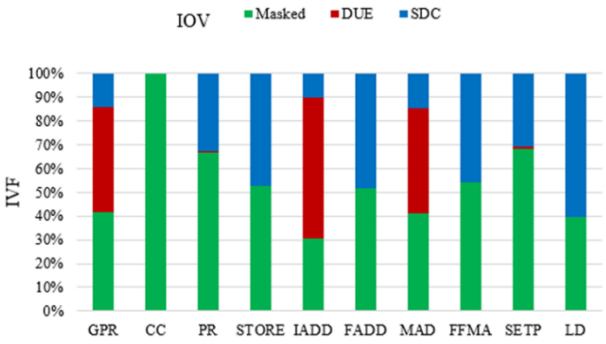

Figure 3. Outcomes of injections in instruction groups (IOV mode) DenseNet201

6. EVALUATION OF THE MITIGATION STRATEGY

In this section we duplicated only the vulnerable instruction groups in the model, by applying our mitigation strategy in the DenseNet201 via executing the instruction three times and voting afterwards. Figures 4 and 5 shows the instructions group after applying the mitigation strategy. As can be seen, PR, STORE, FADD, FFMA, SETP, and LD are the most vulnerable instructions in DenseNet201, and most of the SDCs errors transformed to mask. In Figure 4, both instructions GPR and STORE was masked on the average of 42.38% from the errors that were injected. On another hand, both instructions produced small SDCs errors on the average of about 0.8% errors. However, in Figure 5, instructions PR, STORE, FADD,

FFMA, SETP, and LD masked errors that was injected, with an average SDCs of about 0.14% for all. This is a significant improvement after we applied our mitigation strategy solution. In contrast, all the instructions masked errors on an average of about 9.84% from the errors that was injected.

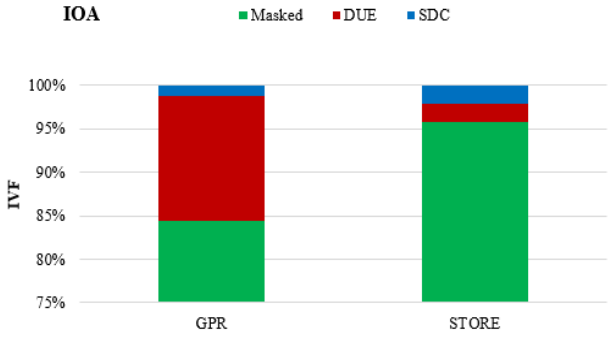

Figure 4. Outcomes of hardened instruction groups (IOA mode) DenseNet201

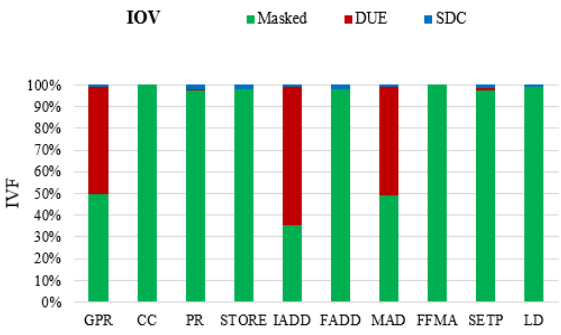

Figure 5. Outcomes of hardened instruction groups (IOV mode) DenseNet201

6.1. Mitigation strategy before vs N-modular redundancy

N-Modular Redundancy (i.e., Triple Modular Redundancy) has been widely used for soft errors masking for high reliablity GPUs application (DenseNet). It is implemented by instantiating 3 copies of all the given DNNs model kernels (i.e., instructions) and by performing a majority voting on the outputs of all instructions. The output given by the majority of the DNNs model will propagate through the voter and if the majority gives the correct output, faults in the other modules are being masked. Nevertheless, the main issue with these solutions is that they have runtime overhead and are not cost effective. Therefore, it is important to better identify the vulnerable instruction portions of the DNNs application (DenseNet) and to reduce the SDC rate with cost-effective and low overhead (see section 3). Tables 2 and 3 summarizes the errors injection, and the instruction groups after applying our mitigation strategy solution.

Table 2. IOA mode instructions group before and after mitigation strategy

	IOA			**IOA***		
IGID	Masked	DUE	SDC	Masked	DUE	SDC
GPR	32.65%	7.45%	9.80%	42.30%	7.15%	0.65%
STORE	35.05%	5.15%	9.80%	42.45%	0.95%	0.95%

IOA = With fault injection (soft errors)
IOA* = After applying Mitigation Strategy

Table 3. IOV mode instructions group before and after mitigation strategy

	IOV			**IOV***		
IGID	Masked	DUE	SDC	Masked	DUE	SDC
GPR	4.14%	4.42%	1.44%	4.98%	4.96%	0.06%
CC	10.00%	0.00%	0.00%	10.00%	0.00%	0.00%
PR	6.66%	0.05%	3.29%	9.74%	0.04%	0.22%
STORE	5.29%	0.00%	4.71%	9.82%	0.00%	0.18%
IADD	3.08%	5.90%	1.02%	3.53%	6.39%	0.08%
FADD	5.17%	0.00%	4.83%	9.81%	0.00%	0.19%
MAD	4.12%	4.39%	1.49%	4.90%	5.02%	0.08%
FFMA	5.42%	0.00%	4.58%	10.00%	0.00%	0.00%
SETP	6.83%	0.07%	3.10%	9.77%	0.07%	0.16%
LD	3.97%	0.00%	6.03%	9.89%	0.00%	0.11%

IOV = With fault injection (soft errors)
IOV* = After applying Mitigation Strategy

7. CONCLUSION

In this paper, we have analyzed the error resilience of DenseNet201, a well-known DNNs model, from the perspective of instructions level. Our analysis showed that DenseNet201 is more prone to SDC than DUE errors, which are more crucial because they modify th emodel's final output. Accordingly, we found

that PR, STORE, FADD, FFMA, SETP and LD are the top vulnerable instructions against soft errors for IOV mode. Specifically, these instructions generate 3.29%, 4.41%, 4.83%, 4.58, 3.10% and 6.03% SDC errors respectively in the injection mode. In contrast, these instructions generated only 0.22%, 018%, 0.19%, 0.00%, 0.16% and 0.11% SDC errors respectively, after applying our mitigation strategy. ThereforeHence, our mitigation strategy solution shows high capacity to mitigate soft errors in the instruction levels of the DenseNet201 model. 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 mission-critical environments

ACKNOWLEDGEMENTS

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

REFERENCES

[1] C. S. Vidya and B. P. Vijaya Kumar, "Reliability Analysis in Healthcare Imaging Applications," *Indian Journal of Science and Technology*, vol. 9, no. 34, pp. 1-5, 2016, doi: 10.17485/ijst/2016/v9i34/100988.
[2] S. S. Janney and S. Chakravarty, "Deep learning in medical and surgical instruments," *Bioelectronics and Medical Devices*, pp. 833-855, 2019, doi: 10.1016/B978-0-08-102420-1.00040-6.
[3] J. J. Rassweiler, R. Autorino, J. Klein, A. Mottrie, A. S. Goezen, J.-Uwe Stolzenburg *et al.,* 'Future of robotic surgery in urology," *BJU Internationa*l, vol. 120, no. 6, pp. 822-841, 2017, doi: 10.1111/bju.13851.
[4] H. Alemzadeh, J. Raman, N. Leveson, Z. Kalbarczyk, and R. K. Iyer, "Adverse events in robotic surgery: A retrospective study of 14 years of fda data," *PLoS ONE*, vol. 11, no. 4, pp. 1-20, 2016, doi: 10.1371/journal.pone.0151470.
[5] E. Rajih, C. Tholomier, B. Cormier, V. Samouëlian, T. Warkus, M. Liberman *et al.,* "Error reporting from the da Vinci surgical system in robotic surgery: A Canadian multispecialty experience at a single academic centre," *Canadian Urological Association Journal*, vol. 11, no. 5, pp. E197-E202, 2017, doi: 10.5489/cuaj.4116.
[6] A. Ferrarese, G. Pozzi, F. Borghi, A. Marano, D. Paola, B. Amato *et al.,* "Malfunctions of robotic system in surgery: Role and responsibility of surgeon in legal point of view," *Open Medicine*, vol. 11, no. 1, pp. 286-291, 2016, doi: 10.1515/med-2016-0055.
[7] F. F. D. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli *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.
[8] 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," *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Denver, CO, USA, 2017, pp. 169-176, doi: 10.1109/DSN-W.2017.47.
[9] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5355-5516, 2019.
[10] Y. Ibrahim, H. Wang, M. Bai, Z. Liu, J. Wang, Z. Yang *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] T. Dhanushya and T. Latha, "Error Correction for Soft Errors," *Journal of Electrical and Electronic Systems*, vol. 07, no. 04, pp. 1-5, 2019, doi: 10.4172/2332-0796.1000276.
[12] F. F. dos Santos, L. Carro, and P. Rech, "Kernel and layer vulnerability factor to evaluate object detection reliability in GPUs," *IET Computers and Digital Techniques*, vol. 13, no. 3, pp. 178-186, 2018, doi: 10.1049/iet-cdt.2018.5026.
[13] Y. Ibrahim, H. Wang and K. Adam, "Analyzing the Reliability of Convolutional Neural Networks on GPUs : GoogLeNet as a Case Study," *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, Tabuk, Saudi Arabia, 2020, pp. 1-6, doi: 10.1109/ICCIT-144147971.2020.9213804
[14] C. Braun, S. Halder, and H. J. Wunderlich, "A-ABFT: Autonomous algorithm-based fault tolerance for matrix multiplications on graphics processing units," *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Atlanta, GA, USA, 2014, pp. 443-454, doi: 10.1109/DSN.2014.48.
[15] J. Zhang, B. Li, K. Xiang, and X. Shi, "Method of diagnosing heart disease based on deep learning ECG signal," *arXiv:1907.0151*, 2017.
[16] K. K. Bressem, L. C. Adams, C. Erxleben, B. Hamm, S. M. Niehues, J. L. Vahldiek, "Comparing different deep learning architectures for classification of chest radiographs," *Scientific Reports*, vol. 10, pp. 1-16, 2020, Art. No. 13590, doi: 10.1038/s41598-020-70479-z.
[17] Z. Huang, X. Zhu, M. Ding, and X. Zhang, "Medical Image Classification Using a Light-Weighted Hybrid Neural Network Based on PCANet and DenseNet," *IEEE Access*, vol. 8, pp. 24697-24712, 2020, doi: 10.1109/ACCESS.2020.2971225.
[18] W. Guo, Z. Xu, and H. Zhang, "Interstitial lung disease classification using improved DenseNet," *Multimedia Tools and Application*s, vol. 78, no. 21, pp. 30615-30626, 2019, doi: 10.1007/s11042-018-6535-y.

[19] A. Çinar and M. Yildirim, "Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture," *Medical Hypotheses*, vol. 139, 2020, Art. no. 109684 doi: 10.1016/j.mehy.2020.109684.
[20] A. Azizimazreah, Y. Gu, X. Gu, and L. Chen, "Tolerating Soft Errors in Deep Learning Accelerators with Reliable On-Chip Memory Designs," *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*, Chongqing, China, 2018, pp. 1-10, doi: 10.1109/NAS.2018.8515692.
[21] F. Libano B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost *et al.*, "Selective hardening for neural networks in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 66, no. 1, pp. 216-222, 2019, doi: 10.1109/TNS.2018.2884460.
[22] F. Kastensmidt and P. Rech, "FPGAs and parallel architectures for aerospace applications: Soft errors and fault-tolerant design," *FPGAs and Parallel Architectures for Aerospace Applications: Soft Errors and Fault-Tolerant Design*, pp. 1-325, 2015, doi: 10.1007/978-3-319-14352-1.
[23] N. Jouppi, "Google supercharges machine learning tasks with TPU custom chip. 18 May 2016," 2019, [Online]. https://cloud.google.com/blog/products/ai-machine-learning/google-supercharges-machine-learning-tasks-with-custom-chip.
[24] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer *et al.*, "Understanding error propagation in Deep Learning Neural Network (DNN) accelerators and applications," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analys*is, SC 2017, 2017, pp. 1-12, Art., No. 8, doi: 10.1145/3126908.3126964.
[25] J. Wei, Y. Ibrahim, S. Qian, H. Wang, G. Liu, Qi. Yu *et al.*, "Analyzing the impact of soft errors in VGG networks implemented on GPUs," *Microelectronics Reliability*, vol. 110, 2020, Art. No. 113648, doi: 10.1016/j.microrel.2020.113648.
[26] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional Networks with Dense Connectivity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1-1, 2019, doi: 10.1109/tpami.2019.2918284.

BIOGRAPHIES OF AUTHORS

Khalid Adam Received the B.Sc. degree (Hons.) in Computer Engineering Faculty of Electrical Engineering, Alzaiem Alazhari University, Sudan in 2013, and the M.Sc. degree from University Malaysia Pahang, Malaysia, in 2016. He is currently pursuing the Ph.D. degree with the College of Engineering, Universiti Malaysia Pahang (UMP), Malaysia. His research interests include reliability of autonomous healthcare application, Deep learning, Big Data Analytics, and Internet of things.

Izzeldin Ibrahim received the B.S. degree from Faculty of Electronics Engineering, Sudan University of Science & Technology in 1994, Sudan. M.S., and Ph.D. degrees from Faculty of Electrical Engineering Universiti Teknologi Malaysia (UTM) in 2000 and 2006 Malaysia respectively. He is currently a senior lecture of faculty of electrical and electronics engineering at University Malaysia Pahang, Malaysia. His research interests include Computer Networking, internet traffic classification, deep learning and FPGA Programming.

Younis Ibrahim received the bachelor's and master's degrees from Sudan and China, in 2013and 2017, respectively. He is currently pursuing the Ph.D. degree with Hohai University, China. His research interests include reliability of AI models through AI accelerators in safety-critical systems and accelerating deep learning models, basically, convolutional neural networks (CNN) models.