On the Reliability of Convolutional Neural Network Implementation on SRAM-based FPGA

Boyang Du, Sarah Azimi, Corrado De Sio, Ludovica Bozzoli and Luca Sterpone
Department of Control and Computer Engineering
Politecnico di Torino
Torino, Italy
{boyang.du, sarah.azimi, corrado.desio, ludovica.bozzoli, luca.sterpone}@polito.it

***Abstract*—In recent years, topics around machine learning and artificial intelligence (AI) have (re-)gained a lot of interest due to high demand in industrial automation applications in various areas such as medical, automotive and space and the increasing computational power offered by technology advancements. One common task for these applications is object recognition/classification whose input is usually an image taken from camera and output is whether an object is present and the class of the object. In industrial pipeline, this task could be used to identify possible defects in products; in automotive application, such task could be deployed to detect pedestrians for Advanced Driver-Assistance Systems (ADAS). When the task is safety-critical as in automotive application, the reliability of the task implementation is crucial and has to be evaluated before final deployment. On the other hand, Field Programmable Gate Array (FPGA) devices are gaining increasing attention in the hardware acceleration part for machine learning applications due to their high flexibility and increasing computational power. When the SRAM-based FPGA is considered, Single Event Upset (SEU) in configuration memory induced by radiation particle is one of the major concerns even at sea level. In this paper, we present the fault injection results on a Convolutional Neural Network (CNN) implementation on Xilinx SRAM-based FPGA which demonstrate that though there exists built-in redundancy in CNN implementation one SEU in configuration memory can still impact the task execution results while the possibility of Single Event Multiple Upsets (SEMU) must also be taken into consideration.**

I. Introduction

Nowadays Convolutional Neural Network (CNN) has been employed in many applications to solve various complicated problems, such as object recognition from image or video input [1], [2], robot control [3], natural language processing [4], [5] and so on, thanks to its increasing performance [1] and accuracy [6], [7]. As one of the common tasks, object recognition has found popularity in many areas, even in safety-critical applications such as automotive, avionics and space applications. For example, in automotive applications, Advanced Driver-Assistance Systems (ADAS) takes input from various sensors including cameras, GPS and radars to detect the environment around car to make decisions on behalf of the human driver to allow faster and safer responses. One common feature implemented in ADAS is pedestrian recognition (and obstacle recognition in some autonomous driving cars), which involves taking continuous image stream from one or more cameras to detect whether pedestrian is present. When CNN is adopted in such applications, as constrained by respective standards (e.g. DO-254 for avionic applications and ISO26262 for automotive), the reliability of the implementation has to be evaluated, not only in the aspect of recognition accuracy but also the target hardware platform with consideration of external environment factors, among which is the radiation effects [8]–[10] even at sea-level.

On the other hand, Field Programmable Gate Array (FPGA) devices have drawn a lot of attention in recent years due to the ever-increasing computational power they offer and the high flexibility by (partial) reconfiguration. Among different types of FPGA devices, SRAM-based FPGA devices have the advantage of higher working frequency and larger configurable resources and have been applied in computational power demanding applications such as in data centers [11], [12]. However, as the configuration memory holding the configuration data of the design mapped to the device is composed of SRAM cells, which are highly susceptible against radiation effects, it is important to analyze the system reliability when SRAM-based FPGA is employed and possible fault tolerant strategies must be applied to meet the reliability constraints, especially in safety-critical applications.

When considering the hardware platform for implementing CNN, especially for the Quantized CNN where floating point data is avoided, FPGA devices pose as a big competitor against another popular platform, namely General Purpose Graphic Processing Unit (GPGPU), thanks to their highly configurable parallel architecture and low power consumption. In [13], authors presented a Quantized CNN implementation with 1 or 2 bits for weights and activations using Xilinx SRAM-based FPGA to be the fastest classification on the given datasets.

In this paper, we present the reliability analysis results of the CNN implementation reported in [13] on Pynq-Z1(-Z2) board using a fault injection platform utilizing multiple boards for acceleration. The main contributions of this paper are: first preliminary analysis regarding a Quantized CNN implementation with only 1 bit weights and activations against SEU and SEMU in the configuration memory; the difference between the fault injection result of SEU in configuration memory controlling other resources show different impacts regarding correctness and performance (though more detailed analysis requires further investigation). The rest of paper is

978-1-7281-2260-1/19/$31.00 ©2019 IEEE

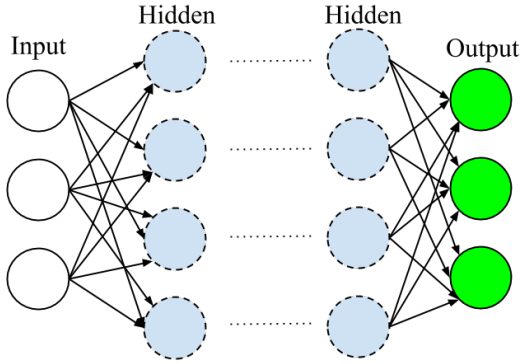
organized as following: Section II provides some background information regarding the CNN implementation under study and state-of-the-art researches presented in literatures; Section III presents the fault injection platform; Section IV presents results from three fault injection campaigns carried out with different fault lists with some preliminary analysis; Section V draws conclusions and talks about future works.

II. Background

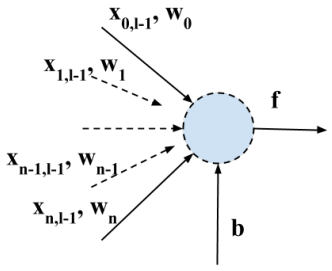
A *Multilayer Perceptron* (MLP) is a type of feedforward Artificial Neural Network (ANN) where its neurons are arranged into multiple layers with inputs of neurons in each layer connected to all the neurons output in previous layer as shown in Fig. 1a, while each neuron generates its output according to a non-linear function of its inputs as shown in Fig. 1b, which could be expressed as:

$$x_l = f(\sum_{n=0} w_n x_{l-1} + b) \quad (1)$$

where w_n is the weight for the output of n-th node in previous layer, x_{l-1} is the output of the n-th node in previous $(l-1)$ layer, b is the bias and f is the activation function. Most commonly used activation functions include $f = tanh(a)$ and Rectified Linear Unit (ReLU) $f = max(0, x)$.



(a) Fully connected MLP



(b) One node in the network

Fig. 1: Network structure of MLP.

CNN [14] as a variation of ANN, has its neurons connected only to a subset of neurons in previous layer to reduce computational demand and allow local features to be detected such as edges and corners. For example, to take an image of 28x28, the fully connected MLP implementation requires the node in the first hidden layer take outputs of all the input nodes as input parameters, which is 784 to calculate the output and propagate to next layer; while a CNN implementation could limit the node takes only a subset of 4x4 nodes from previous layer as parameters and considering the number of nodes in some complicated model, the reduction of computational requirements are enormous.

While large portion of CNN implementations use floating point as data type, researches in recent years have proven that CNN with reduced precision could still produce correct results with high accuracy. For example, in [15], [16] authors presented NN implementation using fixed-point data type; in [17]–[19] authors presented the extreme cases of reduced precision as weights and activations are constrained to single-bit values.

Though the Quantized CNN and the Binary Neural Network (BNN) implementations bring new possibilities of applications where hardware resources are limited, the redundancy induced by floating point data is removed and the impact against reliability of such implementations has to be studied especially when safety-critical applications are concerned.

Regarding the reliability analysis, there have been quite some studies presented in literatures. In [20], authors presented different methods to achieve fault tolerance in NN implementation. Radiation test results on NN-based object recognition implemented on GPU devices have been presented in [21]. In [22], authors presented the results on 3-layer Deep NN implementation by emulating SEU in the configuration memory. In [23], authors presented a case study on the reliability regarding permanent faults affecting Deep NN.

In this paper, we present the analysis results of the Quantized CNN implementation [13] on a Xilinx SRAM-based FPGA where weights and activations are limited to single bit values. The analysis has been done with fault injection platform emulating SEU in the configuration memory and is able to utilize multiple boards for accelerating fault injection campaigns. The main contribution of this paper lies on firstly the parallel scalable fault injection platform and secondly the preliminary analysis of BNN implementation regarding SEU and SEMU on FPGA.

III. Fault Injection Platform

To evaluate the impact of SEU in configuration memory against the CNN implementation on SRAM-based FPGA, a fault injection platform has been built. The main idea is to emulate SEU in configuration memory by generating faulty bitstream with selected bit(s) in frame data flipped, which has been around for several years as a low cost analysis methods before performing radiation test for more realistic result considering radiation induced effects. However, the platform presented in this paper has the advantage of the possibility of utilizing multiple boards, namely Pynq-Z1(Z2), in the platform to speed up the fault injection campaign and the platform has been further extended to support SEMU fault injection in the configuration memory. The overall architecture of the platform is shown in Fig. 2.

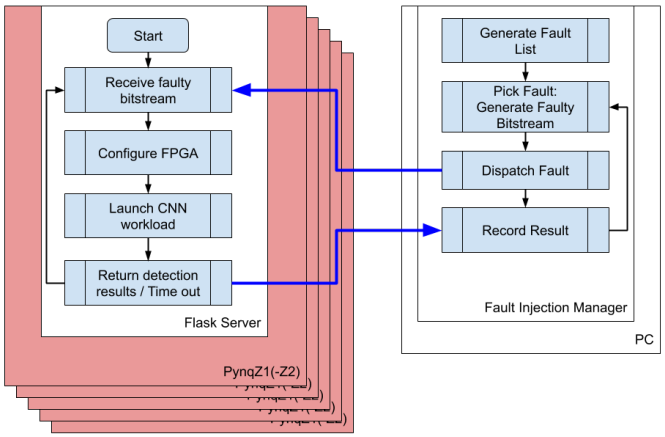


Fig. 2: Overall architecture of fault injection platform.

The Pynq-Z1(Z2) board used in the platform is equipped with a Xilinx Zynq-7000 SRAM-based FPGA device with a hardwired ARM dual-core processor embedded. The Pynq environment provides PetaLinux as operating system with Python environment integrated which makes the platform implementation much easier especially for communicating with host PC. On Pynq board side, a RESTFul server has been implemented using Python3 and Flask to accept fault injection requests from host PC side. The server performs several tasks upon requests:

- Receive faulty bitstream: the faulty bitstream is generated on host PC and uploaded through request to the server on Pynq as it is much faster to generate faulty bitstream on a much more powerful PC and easier to manage since the fault list is also managed by host PC;
- Configure FPGA: in order to launch the fault injection run with uploaded faulty bitstream, the BNN Python library provided by the team of [13] has been modified to force download the bitstream every time the object recognition workload is launched;
- Launch CNN workload: the server will create another process where the object recognition is performed with the CNN implementation presented in [13] and a small image from the SD card will be loaded and used as input for the recognition. Please note that for all the fault injection runs the same image is used though the input image could be configured via request from host PC;
- Return results: once the CNN workload process exit (or failed due to time out), the server will send back the result when host PC requests.

Furthermore, in some cases, the fault injected could cause functional interruption of the server (which could be complicated effects involving also the driver software) that server would stop responding to host PC. In order to avoid "dead" board during the fault injection campaign, the PetaLinux device tree along with the boot image has been recompiled to allow watch-dog reset-on-timeout. So the server when launched will also start the watch-dog count down and periodically "kick" the watch-dog to keep the server alive, and in case of functional interruption, the watch-dog will reset the board after 10 seconds and the server will be online again after the rebooting.

On the host PC side, the Fault Injection Manager (FIM) is in charge of following tasks:

- Generate fault list: depends on the fault model selected, FIM generates a list of faults and stores them in a database;
- Pick fault: FIM will launch another thread to pick the fault from the list which is not executed yet, and generate the faulty bitstream. The number of threads running in parallel is determined by the number of Pynq boards utilized in the platform;
- Dispatch fault: once the faulty bitstream is generated, an "alive" Pynq board (checking the fault injection server running on Pynq board) is searched and if found, the faulty bitstream will be uploaded along with the fault injection request;
- Record result: after sending the fault injection request, the thread will wait for the result from the server with a timeout set to 5 secs (which is much longer than the duration of the used object recognition workload). In case of no response from the server, the thread will report the server as "dead" and wait for its resurrection via watch-dog reset;
- Repeat the above steps until the fault list is exhausted.

With this fault injection platform, it is possible to perform fault injection campaign of large number of faults in a relatively short time depending on how many boards are actually utilized in the platform. In our case, with 4 boards used, the average fault injection time is 1 second per run.

IV. Experimental Analysis

A. Experimental Setup

In this experiment, we selected the *cnvW1A1* network from the available networks from the CNN implementation in [13], which uses only 1 bit as weight and 1 bit as activation. The hardware utilization of *cnvW1A1* network is reported in Table I when implemented on Pynq board. For this network, the *road-signs* classifier as one of the three already provided classifiers is selected to recognize the road signs from input image as workload executed by the fault injection server on Pynq board. The workload reports 2 results: *classification index* and *recognition duration*.

TABLE I: Hardware resource utilization of *cnvW1A1* network

Resources	Available	Used	Utilization(%)
LUT	53200	25447	47.83
Register	106400	42090	39.56
Block RAM	140	124	88.57
DSP	220	24	10.91

Before the fault injection campaign, 10000 runs have been executed with fault-free bitstream to gather the information

regarding the recognition duration as it depends on the timing of the internal implementation which could cause the recognition duration varies within a certain margin. When the fault injection campaign is launched, this margin is used to determine whether a run has been affected by the injected fault leading to performance degradation. Though such margin is not 100% precise, but it gives the idea of how the performance of CNN could be affected negatively by the SEU in configuration memory as presented in later section.

Fig. 3 shows the distribution of the recognition duration without fault in the configuration memory. As can be observed from Fig. 3, most of the runs report the recognition duration around 1580 μs, while a few runs are slightly slower with duration lower than 1700 μs, however, there is some cases the duration could go above 1800 μs.

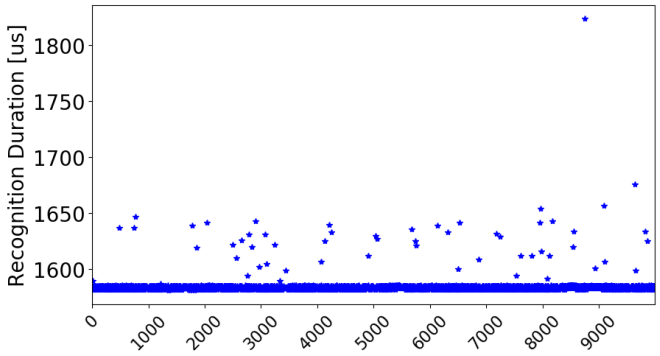


Fig. 3: Recognition duration distribution without fault in configuration memory.

For the fault injection campaign, three different fault lists are generated:

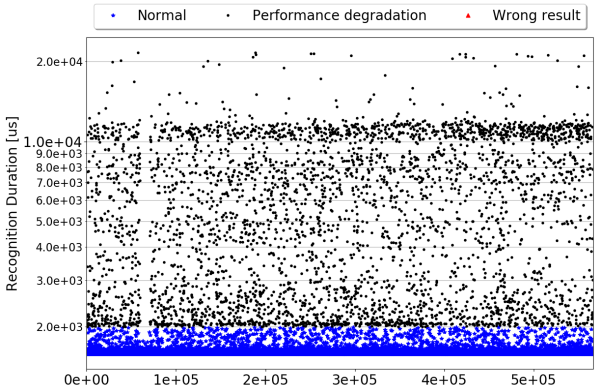
- bits from *Logic Location* file generated along with the bitstream;
- random bits selected in the configuration memory;
- multiple bits, to be precise, 4 bits randomly selected as a cluster to emulate the SEMU as reported in recent radiation test [24].

For each fault injection run, depends on the two values reported by the workload, the result is classified as following:

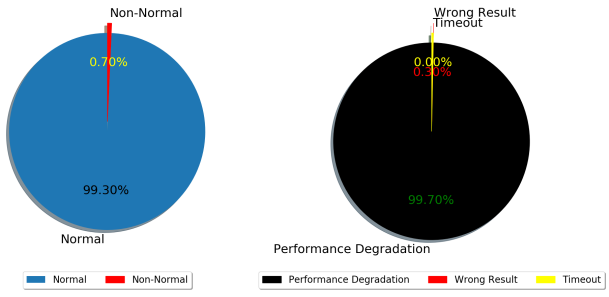
- *Normal*: the object classification is right and the recognition duration is lower than 2000μs;
- *Performance Degradation*: the object classification is right, but the recognition duration is over 2000μs;
- *Wrong Result*: the object classification is wrong;
- *Timeout*: the object recognition did not finish within 5 seconds. This also includes the cases where the functional interruption of the fault injectin server is observed.

B. Bits from Logic Location file

With the bits extracted from *Logic Location* file, which indicates the bits controlling the resources, such as latches, FFs, LUTs and Block RAMs used in the design, more than 500,000 faults are generated and injected. The fault injection results are shown in Fig. 4. As could be observed from Fig. 4b, 99.30% of the faults result in *Normal*, while in the rest of 0.7% of runs, no *Wrong Results* has been observed (though the fault injection campaign is not covering all the bits from *Logic Location* file) and 99.7% of runs suffer from *Performance Degradation* which could go up to more than 21 ms as shown in Fig. 4a. Please note the reason that there is no *Wrong Result* observed could be the fact that the bits from *Logic Location* file are mainly bits related to registers and Block RAMs used in the CNN implementation and due to the fact CNN implementation has certain level of robustness built-in by the multilayer nature and its training process [25], the probability of one corrupted node in the network causing wrong final classification is quite low considering the fact that the weights and activations are limited to single bit values in the selected *cnvW1A1* network. However, the *Performance Degradation* effects observed in the results are also crucial in safety-critical applications, as it may compromise correct execution of some real-time task, besides there exists a few runs with *Timeout*.



(a) Recognition durations



(b) Result distribution

Fig. 4: Fault injection results of faults generated from *Logic Location* file.

C. Random SEU in configuration memory

When SRAM-based FPGA is concerned, the design implemented and mapped to the FPGA usually employs several different types of resources, not only the registers and Block RAMs as in the *Logic Location* file but also the logic functions or Look-Up-Table (LUTs), Digital Signal Processing (DSP)

units, routing resources and so on. In the case of CNN implementation used in this paper, bits in the configuration memory controlling the DSP units, LUTs and configurable routing resources connecting the nodes in the network could also be affected. So, the second fault injection campaign has been performed with randomly selected bits in the configuration memory (without overlapping with the *Logic Location* file).

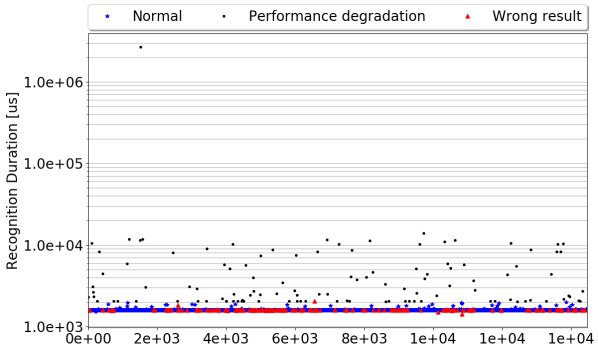
The fault injection results are shown in Fig. 5 of more than 14,000 fault injections. As could be observed from Fig. 5c, comparing to the results in Fig. 4, the *Non Normal* percentage is much higher. More interestingly in the three *Non Normal* categories that there are more than 80% of *Wrong Result* and *Timeout* combined which could be due to the structure of CNN itself been corrupted by the SEU injected in the configuration memory. Furthermore, from Fig. 5b, the *Recognition Duration* of those *Wrong Result* mostly within the range of the *Normal* range, which makes it difficult to detect, similar to the Silent Data Corruption (SDC) errors. This result indicates the necessity of proper fault tolerant technique to improve the overall reliability of CNN implementation not only just for the user register and Block RAMs but more importantly in the configuration memory for protecting the integrity of the CNN itself.

D. Random SEMU (4-bits) in configuration memory

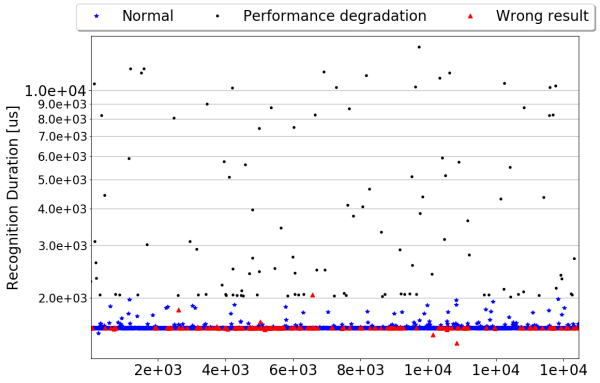
Furthermore, with technology scaling the transistor device is getting smaller and the threshold voltage is getting lower and lower, which means the device is becoming more susceptible against radiation particles even at lower energy level. While considering in space environment, the energy level of possible radiation particles striking the device could reach to GeV scale, SEU in the configuration memory may become insufficient to model the radiation effects. In [24], authors have presented the radiation test results using Ultra High Energy radiation beam on a Xilinx Kintex-7 device, arguing presence of SEMU in the configuration memory. So, another fault injection campaign has been carried out with randomly selected 4-bits SEMU following the patterns reported in [24]. The fault injection results are shown in Fig. 6 of more than 14,000 fault injections. As could be observed from Fig. 6b, part of *Performance Degradation* has worsened into *Wrong Result* and *Timeout* while the proportion of *Non Normal*, comparing to SEU scenario, has increased. The reason for this difference is quite trivial, more bits means more resources could be corrupted while the built-in resiliency against SEU becomes less effective against SEMU.

E. The Verdict

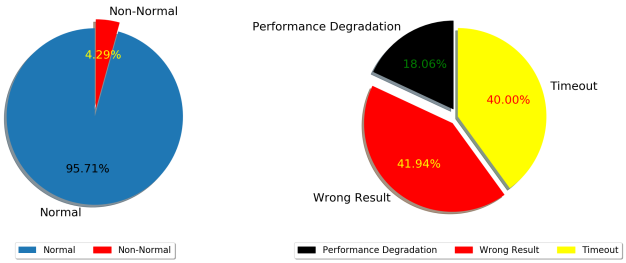
With the results from three fault injection campaigns presented above, it is clear to see that due to the built-in resiliency against single soft error in the data of CNN implementation, even when the weights and activations are limited to single bit values, the implementation still proves good reliability against SEU in configuration memory affecting user register and Block RAMs, however, suffers from some performance degradation. But when bits controlling other resources are affected in the configuration memory, *Wrong Result* (SDC) and *Timeout* start to show and the portion increases when considering SEMU in the configuration memory which makes it important to have fault tolerant strategies to be applied when considering to deploy CNN implementation on SRAM-based FPGA in safe-critical applications where reliability is one of the major constraints.



(a) Recognition durations



(b) Recognition durations (zoom)

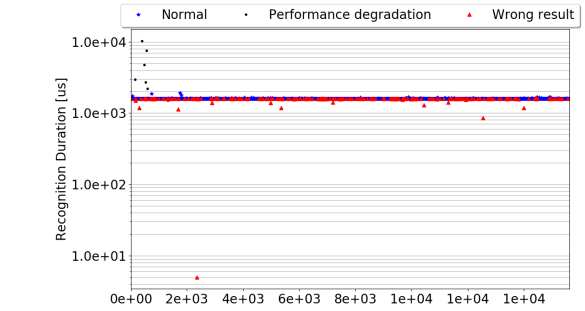


(c) Result distribution

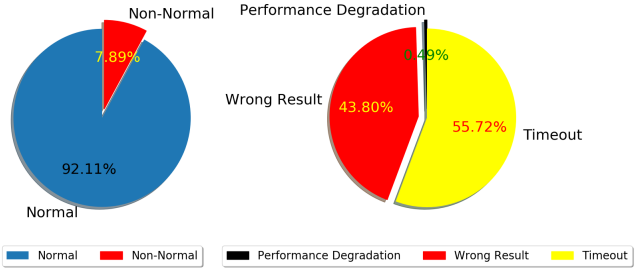
Fig. 5: Fault injection results of faults generated from random bits.

V. Conclusion

In this paper, we presented some preliminary results regarding reliability of CNN implementation on a Xilinx SRAM-based FPGA against SEU and SEMU in the configuration memory using a fault injection platform utilizing multiple FPGA boards. The results demonstrate that the resiliency of CNN itself against single fault in the network leads to



(a) Recognition durations



(b) Result distribution

Fig. 6: Fault injection results of (4 bits) SEMU faults generated from random bits.

high reliability against SEU in the configuration memory when only user register and Block RAMs are affected though suffered from performance degradation; however, when bits controlling other resources such as DSP units, LUTs and interconnections are affected, the CNN implementation starts to generate wrong object classification output; finally SEMU fault injection shows situation worsened comparing to SEU as more resources could be corrupted by SEMU at the same time. So, for CNN implementation to be integrated into safety-critical applications, fault tolerant technique has to be applied such as ECC to protect user register and Block RAMs, TMR and/or configuration memory scrubbing against SEU and SEMU in the configuration memory. More in detail analysis with fault tolerant techniques has been planned as further study of this work.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[3] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

[4] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams *et al.*, "Recent advances in deep learning for speech research at microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8604–8608.

[5] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[7] T. Weyand, I. Kostrikov, and J. Philbin, "Planet-photo geolocation with convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 37–55.

[8] E. Normand, "Single event upset at ground level," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2742–2750, Dec 1996.

[9] J. Leray, "Effects of atmospheric neutrons on devices, at sea level and in avionics embedded systems," *Microelectronics Reliability*, vol. 47, no. 9-11, pp. 1827–1835, 2007.

[10] P. Rech, C. Aguiar, R. Ferreira, C. Frost, and L. Carro, "Neutron radiation test of graphic processing units," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, June 2012, pp. 55–60.

[11] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Enabling fpgas in hyperscale data centers," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*. IEEE, 2015, pp. 1078–1086.

[12] P. Gupta, "Accelerating datacenter workloads," in *26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016.

[13] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. ACM, 2017, pp. 65–74.

[14] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2014, pp. 1–6.

[16] J. Kim, K. Hwang, and W. Sung, "X1000 real-time phoneme recognition vlsi using feed-forward deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 7510–7514.

[17] M. Kim and P. Smaragdis, "Bitwise neural networks," *arXiv preprint arXiv:1601.06071*, 2016.

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[19] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[20] E. B. Tchernev, R. G. Mulvaney, and D. S. Phatak, "Investigating the fault tolerance of neural networks," *Neural Computation*, vol. 17, no. 7, pp. 1646–1664, 2005.

[21] 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)*. IEEE, 2017, pp. 169–176.

[22] I. C. Lopes, F. L. Kastensmidt, and A. A. Susin, "Seu susceptibility analysis of a feedforward neural network implemented in a sram-based fpga," in *2017 18th IEEE Latin American Test Symposium (LATS)*. IEEE, 2017, pp. 1–6.

[23] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez, "A reliability analysis of a deep neural network," in *2019 IEEE Latin American Test Symposium (LATS)*, March 2019, pp. 1–6.

[24] B. Du, L. Sterpone, S. Azimi, D. M. Codinachs, V. Ferlet-Cavrois, C. B. Polo, R. G. Alía, M. Kastriotou, and P. Fernández-Martmnez, "Ultra high energy heavy ion test beam on xilinx kintex-7 sram-based fpga," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2019.

[25] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *arXiv preprint arXiv:1511.06488*, 2015.