C-SMART: A preprocessor for neural network performance and reliability under radiation

A. J. Rajappa [a,*], P. Reiter [a], P. Rech [b], S. Mercelis [a], J. Famaey [a]

[a] *IDLab, University of Antwerp - imec, Sint-Pietersvliet 7, 2000, Antwerp, Belgium*
[b] *Università di Trento, Via Calepina 14, Trento, 38122, Italy*

Abstract

Edge AI brings the benefits of AI, such as neural networks for computer vision analyses, to low-power edge computing platforms. However, application and resource constraints leading to inadequate protection can make edge devices vulnerable to environmental factors, such as cosmic rays. These factors can cause bit flips that affect the reliability of the neural network inferences computed using these edge devices. To address this issue, we developed the C-SMART preprocessor that adds a theoretically analysed condition on top of the SMART technique for obtaining both reliability and performance benefits. SMART is a reliability improvement technique introduced in our previous work, which involves skipping the multiply-accumulate operations performed on the zero-valued inputs to the layers of the neural network. We demonstrated C-SMART with a commercial bare-metal system containing an ARM microprocessor by exposing the system to real-world, atmospheric-like neutron radiation using the ChipIr facility in Oxfordshire, UK. We also conducted timing measurements for performance analysis. Our experiments with C-SMART for inference with a neural network revealed a reliability boost against soft errors by more than 26%, simultaneously improving performance by more than 35%. We foresee these benefits in various COTS devices by integrating C-SMART with compilers and NN generators.

1. Introduction

Edge AI, which combines the concepts of artificial intelligence (AI) and edge computing enables the execution of machine learning [1], specifically inferencing [2] algorithms closer to the edge. Compared to cloud-based AI, this concept offers several advantages, such as low latency, improved privacy and security, and reduced uplink/downlink requirements [2, 3]. Applicability of machine learning algorithms is increasing in health and medical instruments, autonomous vehicles [2], aerospace vehicles [3], aviation [4], nuclear power plants [5] and other similar applications, which are usually mission critical. Thus, any compromise in the reliability of these AI algorithms can lead to "critical consequences" [6, 7], such as total system failures and fatalities, which must be avoided at all costs. Unlike the training phase, the inference phase requires less computational resources and can be more readily deployed in application-and-resource-constrained edge devices such as microcontrollers [2].

The reliability of edge inference is affected by external factors, such as abnormal radiation levels, cosmic rays, radioisotopic impurities in the package and chip materials, and unstable or low power supplies, which are found in both conventional and hostile environments [6, 8]. These factors can cause different types of faults, including transient faults that manifest as single bit-flips [7], where a bit's state is flipped from logic 0 to logic 1 or vice versa. This is called a soft error [6]. The edge devices that execute inferences are usually situated near the data source to facilitate the associated applications [9]. Hence, they cannot always be ideally protected from the above factors and can experience such faults [10]. Thus, multiple techniques have been proposed to enhance the NN inference fault tolerance [11]. This includes Selective Multiply-Accumulate zeRo-opTimization (SMART) [12] software technique for fully connected neural networks (FC-NN), which is used in different types of Deep Neural Network such as Multilayer Perceptrons (MLP), Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN) along with its variants, namely, Long Short-Term Memory (LSTM) networks and Transformers [13].

In this article, we propose Conditional-SMART (C-SMART), a preprocessor designed to answer the question 'When to use SMART?'. The preprocessor takes the neural network and its test dataset as inputs

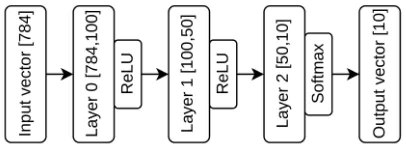


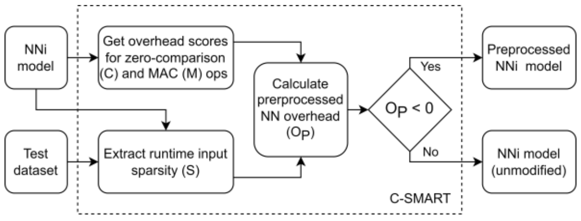
Fig. 1. FC-NN architecture.

to check and provide both reliability and performance benefits using SMART when certain conditions are met. Multiple versions of a proof-of-concept FC-NN, including the one preprocessed with C-SMART, were experimented with the bare-metal nRF52840 DK system, which is a commercial ARM-based platform with Cortex-M4 processor [14]. Experiments include timing measurements for performance analysis, and exposure to neutron radiation at the ChipIr facility, STFC, Rutherford Appleton Laboratory [15] in Oxfordshire, UK for reliability analysis. The radiation experiment used a real-world neutron spectrum [8]. The experimental data can be referenced at [16].

2. Case study algorithms

Figure 1 shows the FC-NN architecture used for the experiments. This proof-of-concept NN architecture was used to train and evaluate an NN using the TensorFlow [17]. The NN parameters obtained after training are transferred to a C language based custom NN inference algorithm. For demonstration purposes, the FC-NN was trained and tested using the MNIST [18] dataset, with a testing accuracy of 97.98%. MNIST dataset was chosen due to the small size of its images (hence, manageable input image transfer times to inference models during the radiation experiments), widespread applicability in AI and provides baseline results with NNs which can later be used by emerging AI algorithms, such as hyperdimensional computing (HDC) [19].

A total of 250 images were selected at random from the available test images, normalized and flattened to create a one-dimensional array (input vector) of size 784 to form the input dataset, with their elements in single precision floating-point format (FP32). This input dataset was used throughout the experiments. The size 250 for the input dataset was chosen for achieving manageable runtime periods



NNi model
Test dataset
Get overhead scores for zero-comparison (C) and MAC (M) ops
Extract runtime input sparsity (S)
Calculate prerprocessed NN overhead (Op)
Op < 0
Yes
No
Preprocessed NNi model
NNi model (unmodified)
C-SMART

Fig. 2. C-SMART preprocessor process flow.

Table 1
Input sparsity throughout the different NN layers

Layer 0	Layer 1	Layer 2	All the layers
0.807	0.666	0.473	0.774

during the experiment. After each inference computed with an input image, ten FP32 values are generated as the output vector, representing the **probabilistic output**, the input image's probability of being a digit from zero to nine. The digit whose corresponding probability is the highest, and thus closest to the input image, is considered the **inference output**.

Four distinct versions of the FC-NN algorithm that performs the NN inference (NNi) were executed and evaluated during the experiments for comparative analysis. They are the NN inference model with none of the proposed optimizations, **Simple NNi**; NN inference model preprocessed by C-SMART, **Preprocessed NNi**; temporal Triple Module Redundancy (TMR) [12] version of Simple NNi, **TMR NNi**; and temporal TMR version of Preprocessed NNi, **TMR Preprocessed NNi**. All the versions were individually compiled using the SEGGER compiler [20] with soft type application binary interface for floating point operations.

3. C-SMART preprocessor

Figure 2 shows the C-SMART's process flow. The trained NNi model and its test dataset are fed as input to the preprocessor. The inputs are used to extract: the runtime input sparsity (S), and the overhead associated with zero-comparison (C) and the multiply-accumulate (MAC) operation (M) [12]. S, C and M are then used to calculate the overhead of preprocessed NN (O_P) using the equation $O_P = C - MS$. When $C < M$, which is the most common scenario in general-purpose computing implementations, if $S = 1$ or $S > C/M$, then the overhead of the Preprocessed NNi is not positive i.e., $O_P < 0$. Under this condition, preprocessing the input NNi can be more beneficial.

Input sparsity for various layers of the NN employed during experiments was calculated using all the 10,000 MNIST test images and it is displayed in Table 1. As shown, the overall input sparsity for input values throughout all the layers (S) is 0.774, which means that around 77.4% of the input values are zero. Let us define the overhead scores C and M, of the checking and MAC operations, respectively, as the number of assembly instructions associated with each. They are obtained by analyzing the corresponding executables with Objdump [21].

In our case, $C = 16$ and $M = 175$. Then $O_P = -119.4$ i.e., the Preprocessed NNi overhead is not positive. The quantity 119.4 for O_P can be interpreted as the average number of skipped instructions due to preprocessing for each input value. There are 83,900 input values associated with each inference for the NN architecture considered. More than 10 million instructions are skipped while computing an inference with the preprocessed NN model; hence, C-SMART converts the input NNi into Preprocessed NNi for added performance and reliability benefits.

4. Experiments, results and discussion

Execution time measurement with DWT [22] and radiation experiments were conducted on the case study algorithms. The results are shown in Tables 2 and 3. Due to skipped instructions in Preprocessed NNi, the execution time reduced by 4.5 million cycles, about 70 ms per inference in nRF52840 DK (64 MHz), compared to Simple NNi. This reduction effect between the TMR versions tripled as expected.

Figure 3 shows the radiation chamber and the test hardware. The raw data from the radiation experiment was analyzed by comparing it with the golden results, i.e., probabilistic and inference outputs obtained by executing the case study algorithms in a conventional environment. Various types of errors observed during analysis are: Probabilistic output mismatches without a mismatch in inference output within an inference are considered as one **tolerable error**; probabilistic output mismatches that also lead to a mismatch in inference output within an inference are considered as one **critical error**; the sum of all tolerable and critical errors is considered as the **output error** count; and errors successfully handled within the TMR versions are considered as **masked errors**.

Combining the radiation facility log with the raw data timestamp provides the neutron fluence for each

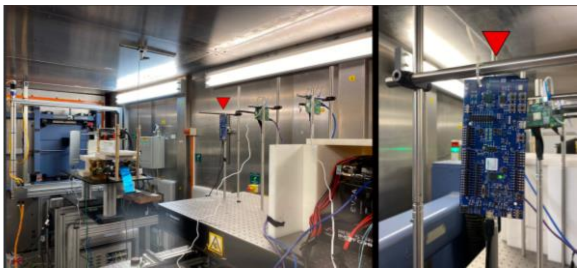


Fig. 3. Aligning the nRF52840 DK (indicated in red) with the projected path of the neutron beam.

of the case study algorithms. Dividing the number of observed errors in a case study algorithm by the corresponding neutron fluence gives soft error cross section values, which is a probabilistic measure of a neutron to cause an error in that case study algorithm running in our bare-metal system [23].

Due to the extended irradiation period, the neutron fluence for the preprocessed versions is higher than the other versions. However, the soft error cross-section values suggest that if the same number of neutrons were to pass through the bare-metal system while executing each of the case study algorithms, preprocessed versions would have a lower chance of facing a tolerable or critical error due to those neutrons when compared to the associated non-preprocessed versions. In TMR versions, we observed errors successfully masked by the voting mechanism and only observed tolerable errors in the output.

5. Conclusion and future work

C-SMART preprocessed our proof-of-concept FC-NN, bringing in both performance and reliability benefits. This is demonstrated with the Preprocessed NNi and TMR Preprocessed NNi algorithms whose performance and reliability metrics were better

Table 2
Experimental results for Simple NNi and Preprocessed NNi

NN Versions	Error count			Neutron fluence	Soft error cross-section (10^{-10} cm^2)			Avg. exec. time
	Tolerable	Critical	Output	10^{10} neutrons/cm^2	Tolerable	Critical	Output	million-cycles
Simple	25	1	26	07.9212	3.1561	0.1262	3.2823	12.7
Preprocessed	61	2	63	26.1172	2.3356	0.0766	2.4122	8.2

Table 3
Experimental results for TMR NNi and TMR Preprocessed NNi

NN Versions	Masked error		Error count	Neutron fluence	Soft error cross-section	Avg. exec. time
	Probabilistic	Inference	(Tolerable)	10^{10} neutrons/cm^2	(10^{-10} cm^2)	million-cycles
TMR	25	1	1	07.9212	0.1262	37.5
TMR Preprocessed	62	2	1	26.1172	0.0383	23.5

compared to the (non-optimized) Simple NNi and TMR NNi algorithms. According to the analysis of experimental results, preprocessing the NN inference model with C-SMART decreased the soft error cross-section of NN inference by **26%** and TMR NN inference by **70%**, and reduced the average execution time per inference by more than **35%**.

C-SMART can be integrated into compilers and NN model generators to bring these benefits on a wide range of low-power, commercially available off-the-shelf (COTS) devices to run fully connected networks and layers in their intelligent algorithms more reliably and efficiently. Further analysis of the execution energy of the case study algorithms will reveal the affinity of C-SMART towards lowering power consumption. In future, the overhead scoring mechanism in C-SMART can be further improved by including the effects of special hardware used for zero-comparisons and MAC operations, such as the Floating-Point Unit (FPU)s and other accelerators.

Acknowledgements

Beam experiments were provided by the ChipIr team thanks to Christopher Frost, Carlo Cazzaniga, and Maria Kastriotou (DOI: 10.5286/ISIS.E.RB2200-004-1). This work has been partially supported by the MOVIQ (Mastering Onboard Vision Intelligence and Quality) project funded by Flanders Innovation & Entrepreneurship (VLAIO) and Flanders Space (VRI); and the European Union's Horizon 2020 research and innovation programme under the grant agreement No 101008126.

References

[1] S. Lakrouni, M. Sebgui, and S. Bah, "Using ai and iot at the edge of the network," in 8th International Conference on Optimization and Applications (ICOA), pp. 1–6, 2022.
[2] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for ai enabled iot devices: A review," Sensors, vol. 20, no. 9, 2020.
[3] G. Furano, G. Meoni, A. et. al., "Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities," IEEE Aerospace and Electronic Systems Magazine, vol. 35, pp. 44–56, 2020
[4] E. Kulida and V. Lebedev, "About the use of artificial intelligence methods in aviation," in 2020 13th International Conference "Management of large-scale system development" (MLSD), pp. 1–5, 2020.
[5] A. Ramos, A. Carrasco, et. al., "Artificial intelligence and machine learning applications in the spanish nuclear field," Nuclear Engineering and Design, vol. 417, p. 112842, 2024.
[6] A. Vega, P. Bose, and A. Buyuktosunoglu, "Chapter 1 and chapter 2" in Rugged Embedded Systems, pp. 1–37, Boston: Morgan Kaufmann, 2017.
[7] Z. Chen, G. Li, and K. Pattabiraman, "A low-cost fault corrector for deep neural networks through range restriction," in 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 1–13, 2021.
[8] C. Cazzaniga and C. D. Frost, "Progress of the scientific commissioning of a fast neutron beamline for chip irradiation," Journal of Physics: Conference Series, vol. 1021, p. 012037, may 2018.
[9] W. Shi and S. Dustdar, "The promise of edge computing," Computer, vol. 49, no. 5, pp. 78–81, 2016.
[10] J. Chen, J. Klein, Y. et. al., "A thermoelectric energy harvesting system for powering wireless sensors in nuclear power plants," IEEE Transactions on Nuclear Science, vol. 63, no. 5, pp. 2738–2746, 2016.
[11] F. Su, C. Liu, and H.-G. Stratigopoulos, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives," IEEE Design & Test, vol. 40, no. 2, pp. 8–58, 2023.
[12] A. J. Rajappa, P. Reiter, et. al., "Smart: Selective mac zero-optimization for neural network reliability under radiation," Microelectronics Reliability, p. 115092, 2023.
[13] W. Su, L. Li, F. Liu, M. He, and X. Liang, "Ai on the edge: a comprehensive review," Artificial Intelligence Review, vol. 55, no. 8, pp. 6125–6183, 2022.
[14] Nordic semiconductors, "nrf52840 dk." Available online: https://www.nordicsemi.com/Products/Developme-nt-hardware/nrf52840-dk, (accessed on 28 Jan 2024).
[15] UK Research and Innovation (UKRI), "Chipir." Available online: https://www.isis.stfc.ac.uk/Pages/Chipir-.aspx, (accessed on 23 Mar 2024).
[16] A. Justus, "Experimental dataset: C-SMART for reliable and efficient NN inference in a neutron-irradiated bare-metal system." Available online: https://doi.org/10.-5281/zenodo.7962582, May 2023.
[17] M. Abadi, A. Agarwal, et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.
[18] Y. Lecun, L. Bottou, et. al., "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
[19] D. Ma, S. Zhang, and X. Jiao, "Robust hyperdimensional computing against cyber attacks and hardware errors: A survey," in Proceedings of the 28th Asia and South Pacific Design Automation Conference, ASPDAC'23, (New York, NY, USA), p. 598–605, Association for Computing Machinery, 2023.
[20] SEGGER, "Segger compiler." Available online: https://wiki.segger.com/SEGGER_compiler, (accessed on 28 Jan 2024).
[21] GNU, "Objdump." Available online: https://ftp.gnu-.org/old-gnu/Manuals/binutils-2.12/html_node/binutils_6. html#SEC6, (accessed on 28 Jan 2024).
[22] ARM, "Chapter 9. data watchpoint and trace unit." Available online: https://developer.arm.com/docume-ntation/ddi0439/b/Data-Watchpoint-and-Trace-Unit?lang=en, (accessed on 28 Jan 2024).
[23] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, Std. JESD89B, Sep. 2021." Available online: https://www.jedec.org/system/files/docs-/JESD89B.pdf, (accessed on 28 Jan 2024).