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

Anuj Justus Rajappa[a,*] (Researcher), Philippe Reiter[a], Paolo Rech[b], Siegfried Mercelis[a] and Jeroen Famaey[a]

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

ARTICLE INFO

Keywords:
machine learning
neural network
reliability
preprocessor
radiation
bit-flips

ABSTRACT

Edge AI brings the benefits of AI, such as neural networks for computer vision analysis, 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 that continually shower on Earth. 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 Conditional-SMART (C-SMART) preprocessor designed to answer the question 'When to use SMART?', 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 and energy 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% while improving performance by more than 35%. We foresee these benefits in various COTS devices by integrating C-SMART with compilers and neural network generators.

1. Introduction

Edge AI, which combines the concepts of artificial intelligence (AI) and edge computing enables the execution of machine learning [37], specifically inferencing algorithms closer to the edge [48]. Compared to cloud-based AI, this concept offers several advantages, such as low latency, improved privacy and security, and reduced uplink/downlink requirements [48, 21]. Applicability of machine learning algorithms is increasing in health and medical instruments, autonomous vehicles [48], aviation [35], aerospace vehicles, interplanetary rovers [21], nuclear power plants [55] and other similar applications, which are usually mission-critical. Thus, any compromise in the reliability of these AI algorithms can lead to "critical consequences" [63, 18], 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 [48].

The reliability of edge inference is affected by 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 [63, 16]. These factors can cause different types of faults, including transient faults that manifest as single bit-flips [18], where a bit's state is flipped from logic 0 to logic 1 or vice-versa. This is called a soft error [63]. The edge devices that execute inferences are usually situated near the data source to facilitate the associated applications [59]. Hence, they cannot always be ideally protected from the above factors and can experience such faults [17]. Thus, multiple techniques have been proposed to enhance the neural network (NN) inference reliability [60]. This includes the Selective Multiply-Accumulate zeRo-opTimization (SMART) [54] software technique for fully connected neural networks (FC-NN), which is used in different types of Deep Neural Network (DNN) such as Multilayer Perceptrons (MLP), Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN), along with the variants of RNN such as Long Short-Term Memory (LSTM) networks and Transformers [61]. SMART involves skipping the Multiply ACcumulate (MAC) operations when its operand i.e., the input value to neurons is zero.

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

⋆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 №101008126.

*First and corresponding author

anuj.justusrajappa@uantwerpen.be (A.J. Rajappa); philippe.reiter@uantwerpen.be (P. Reiter); paolo.rech@unitn.it (P. Rech); siegfried.mercelis@uantwerpen.be (S. Mercelis); jeroen.famaey@uantwerpen.be (J. Famaey)

anujfalcon.com, anuj.justus@gmail.com (A.J. Rajappa)

ORCID(s): 0000-0001-8167-9171 (A.J. Rajappa); 0000-0002-2548-7172 (P. Reiter); 0000-0002-0821-1879 (P. Rech); 0000-0001-9355-6566 (S. Mercelis); 0000-0002-3587-1354 (J. Famaey)

https://www.linkedin.com/profile/view?id=anujfalcon (A.J. Rajappa)

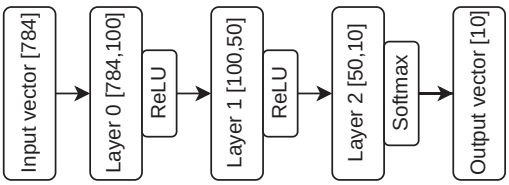

Figure 1: FC-NN architecture.

and performance benefits of SMART when certain conditions are met. Multiple versions of a proof of concept FC-NN with the architecture shown in Figure 1, including the ones preprocessed with C-SMART, were experimented on the bare-metal nRF52840 DK system, which is a commercial ARM-based platform with 64MHz Cortex-M4 processor [2]. Experiments include timing measurements for performance analysis, and exposure to neutron-based radiation at the ChipIr facility, STFC, Rutherford Appleton Laboratory [4] in Oxfordshire, UK for reliability analysis. The radiation experiment used a real-world neutron spectrum [16]. The experimental data can be referenced at [32].

The following section briefly details various works related to our study, Section 3 describes the algorithms used for the case study, Section 4 theoretically analyses the effect of preprocessing the algorithms with C-SMART and describes the associated process, Section 5 details various experiments conducted with the case study algorithms and discusses the results, and Section 6 concludes this article with a summary of the study, results and future works.

2. Related works

Multiple optimization techniques including those that require preprocessing NN models such as Model Modification and Fault-Aware Training have been proposed for enhancing the fault tolerance of NN inference to improve the reliability [60]. General techniques such as error detection and correction codes [25, 68, 27]; interleaving bits of a word with optimal interleaving distance and single error correction codes (SEC) [56]; and monitoring the current consumption of SRAM using built-in current sensors (BICS) to detect abnormality associated with single event upset (SEU) and correcting them with a parity bit per RAM word [52, 62] can be applied to raise the fault tolerance of NN algorithms.

Classical methods are the N-Module Redundancy (NMR) [34] techniques, which include Dual Module Redundancy (DMR) [45, 19, 33], Triple Module Redundancy (TMR) [64, 43, 20] and Penta Module Redundancy (5MR) [66]. They use module duplication to get multiple outputs from the same kind of modules. These outputs are then fed to a voting mechanism to elect the final output. Two ways of duplication in NMR are spatial (executing the same algorithm in multiple hardware) and temporal (executing the same algorithm multiple times in single or multiple hardware). One major disadvantage of NMR is the resource overhead associated with this technique when applied to the entire NN algorithm. Several methods, such as approximate computing and selective hardening [50, 12], have been proposed to reduce this overhead.

Recently, researchers [42] have proposed two approaches, namely, selective kernel hardening and Symptom-based Duplication with Comparison (SDWC). The former approach statically protects selective parts of the NN through duplication and the latter protects the output of each layer at runtime using signature and output value checks against a pre-compiled threshold. The approaches were evaluated by injecting faults in an ARM platform using the On-Chip Debugger (OCD). Researchers [9] have also assessed the soft error reliability of the CNN executing on an ARM Cortex-M processor architecture using the Common Microcontroller Software Interface Standard-NN (CMSIS-NN) [36] library and Soft error Fault Injection Analysis (SOFIA) [14] tool. Later, the Register Allocation Technique (RAT) [8, 22] was introduced, where critical CNN parts are allocated to a pool of specific general purpose registers to improve soft error reliability. RAT along with precision bitwidth variation was assessed using Open Virtual Platforms simulator (OVPsim) [29] and SOFIA [8]. A fault injection experiment [10] to study the impact of precision bitwidth alone on the fault tolerance of CNN in the ARM platform with the CMix-NN [15] library and SOFIA was also conducted.

Again with SOFIA, the ARM platform was further subjected to fault injections and it was found that the CNN's susceptibility to soft errors increases with SIMD instructions due to the associated increase in the memory footprint [7]. The effect of thread parallelism on soft error reliability of the CNN in an ARM platform was also studied and found that multi-threaded versions positively impact CNN reliability [6]. Three kinds of algorithms, namely NN, Random Forest and Support Vector Machine (SVM) were tested under radiation by executing them on an ARM Cortex-M4 processor and they were found to exhibit intrinsic fault tolerance characteristics [53].

One technique that has been prevalent in enhancing the fault tolerance of NN algorithms is quantization, which has been shown through simulated fault injection and real-world radiation test campaigns [57, 41, 39, 65]. Another study [13] introduces Zero-Biased MNU-Aware SRAM Cell (ZBMA) that leverages the observation that parameters such as weights and feature maps of DNN have a strong tendency towards being zero and a soft-error that flips a bit zero to one is more likely to cause a failure. Works focusing on selective hardening alone have also reported improvement in fault tolerance of NNs [40]. For instance, researchers [46] have proposed the "feature-map level resilience technique (FLR)" and "inference level resilience technique (ILR)". The former identifies parts of a CNN that are most vulnerable and statically protects them through duplication, while the latter analyzes the output from the inferences to rerun the vulnerable ones. Another study [58] uses the error detection and mitigation network (EDMN) for detecting anomalies in the intermediate outputs of NNs and improving their fault tolerance, where EDMN is also an NN protected by TMR.

Table 1
Difference between state-of-the-art (SotA) techniques and C-SMART in enhancing the fault tolerance of NNs.

SotA techniques	Preprocessing with C-SMART
Selective radiation hardening techniques [42, 46, 40] require profiling a NN for identifying vulnerable parts.	Identification of vulnerable NN parts is not required for implementation. However, profiling is required for overhead assessments.
Quantization can reduce model accuracy, depending on the resolution (or data type) and application (quantization-aware training and post-training quantization) [23, 67].	Can work with any data type, but could be beneficial only if that data type includes zero. Preprocessing with C-SMART does not impact the model's accuracy.
Additional processes such as calibration are required for obtaining quantization parameters such as clipping range [23, 67].	Additional processes such as calibration are not required.
Some techniques [46, 42] analyse the outputs of an NN layer for error detection and reruns inference upon detecting errors. They can be expensive in terms of execution time.	Neither error detection nor reruns of inference are required. Besides, in our case, the optimization also reduces the time taken for inference execution but does not completely eliminate critical errors.
The range restriction-based techniques [18, 26] might mitigate but not eliminate soft errors.	Reduces the probability of a soft error transpiration (elimination).
The range restriction-based techniques [18, 26] might clip large valid values that might arise on using the input data outside the validation dataset.	Output values are not restricted.
The range restriction [18, 26] and NMR [34, 19, 64, 43, 46, 40] based fault tolerance improvement techniques can incur additional overhead.	Reduces the overhead and, in our case, improves inference performance by leveraging input sparsity and skipping MAC operations.
Techniques that use error detection and correction codes [25, 68, 27, 56, 62] to improve fault tolerance require redundant bits.	Redundant bits are not required.
Techniques such as RAT [8, 22] improve fault tolerance by restricting the registers used for executing certain functions.	No explicit restrictions in the registers used for execution.
Techniques that use anomaly detecting NNs such as EDMN [58] with NMR based protection for detecting errors in another NN incurs additional overhead.	Anamoly detecting NN is not required. Besides, improves performance by reducing execution time and energy.
Special hardware implementations [47, 51] can improve the performance of MAC operations when an operand is zero but do not study its impact on fault tolerance.	Can improve performance pertaining to MAC operations when the input value is zero, through software changes; and improves fault tolerance.
Hardware implementations [13, 52, 47, 51] can be difficult to realize in existing and upcoming devices on various scales due to the cost and development overhead.	Can be relatively easy to implement through a software update in both existing and upcoming devices at various scales.

A technique called Ranger, which selectively restricts the ranges of values in DNNs, has also been found to improve fault tolerance [18]. Similar to Ranger, FT-ClipAct is another technique where the unbounded activation functions are clipped with a range to bound the outputs for raising the fault tolerance of NNs [26]. The clipping threshold or range was derived after analysis using a validation dataset.

The established relationship exists between the NN's fault tolerance and the sparsity of its weights, where the sparsity is found to enhance fault tolerance [57]. At the hardware level, architectures that take advantage of zero operands in MAC operations have been proposed to improve performance and energy efficiency [47, 51].

However, the above-described techniques differ from C-SMART as displayed in Table 1. Even though C-SMART differs from the above techniques, it is complementary and thus can be combined with other techniques to create a hybrid fault tolerance optimization technique.

3. Case study algorithms

Figure 1 shows the FC-NN architecture used for the experiments, which consists of three layers. This proof-of-concept NN architecture was used to train and evaluate an NN using the TensorFlow [5]. 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 Modified National Institute of Standards and Technology (MNIST) [38] 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) [44].

A total of 250 images were selected at random from the available test images, normalized and flattened to create a

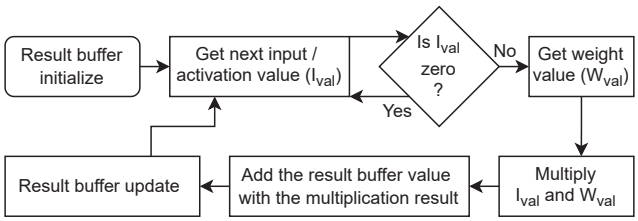

Figure 2: Process flow of the Preprocessed NNi implementation.

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 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 algorithm that perform 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 TMR version of Simple NNi (**TMR NNi**) and Temporal TMR version of Preprocessed NNi (**TMR Preprocessed NNi**). All the versions were individually compiled in the SEGGER embedded studio IDE using the SEGGER compiler [3]. The compilation targets the Cortex-M4 ARM core type present in the nRF52840 DK with soft type application binary interface for floating point operations.

3.1. Simple NNi

The Simple NNi is a custom algorithm implemented to perform NN inference, that takes in the NN parameters such as the number of layers, number of neurons per layer, activation functions [11], weights, and biases, which are obtained after training the NN. The experimental results based on Simple NNi were anticipated to provide a lower point of reference for fault tolerance when compared to the Preprocessed NNi.

3.2. Preprocessed NNi

The Preprocessed NNi is implemented similarly to the Simple NNi. The only crucial difference is in the part of the code that deals with MAC operations associated with an input value, whose program flow is displayed in Figure 2. The MAC operation is guarded by a piece of code responsible for the input value checking and branching operation, which branches to change the program flow and skip the MAC operation only when the input value is zero, i.e., SMART.

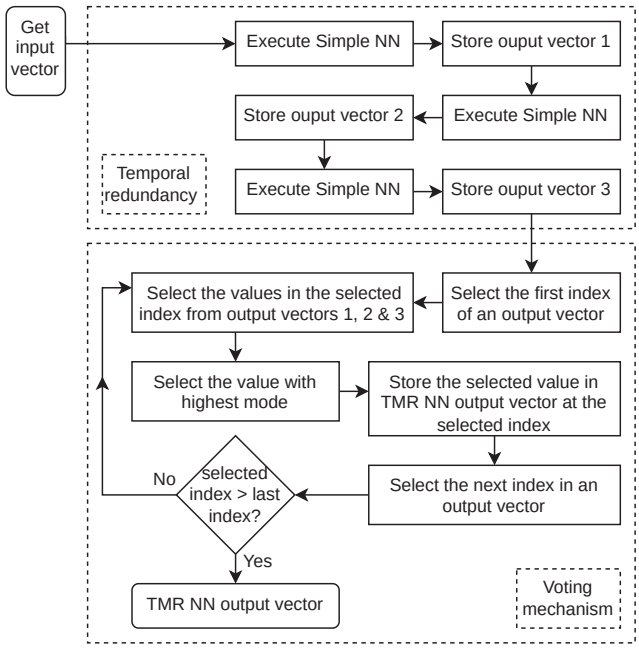

Figure 3: Process flow of a TMR NNi.

3.3. TMR NNi

The TMR NNi executes the Simple NNi thrice, one after the other, to take advantage of the temporal triple module redundancy, as portrayed in Figure 3. The probabilistic output of all three executions is stored. After the three executions, for a given index in the probabilistic output vector, the corresponding values from all three executions are gathered and compared. Out of the compared values, the value which appears most often (i.e., the value with the highest statistical mode) is selected and stored as the probabilistic output for that index in the output vector for the TMR NNi (voting). This approach is comparable to the TMR approach proposed by Esposito and colleagues [20] for time-critical tasks, but the executions are performed serially and a software-based voting algorithm is used. The experimental results based on TMR NNi were anticipated to provide a higher point of reference for fault tolerance when compared to the Preprocessed NNi.

3.4. TMR Preprocessed NNi

The TMR Preprocessed NNi is implemented in a manner similar to the TMR NNi, except that the Preprocessed NNi is executed thrice instead of the Simple NNi. The experimental results based on TMR Preprocessed NNi were expected to aid in analyzing the effect of TMR on Preprocessed NNi.

4. C-SMART preprocessor

Skipping a MAC operation when the input value is zero requires checking the value and conditionally branching (i.e., zero-comparison operation) before the MAC operation. Let us consider only the overhead of these two operations during

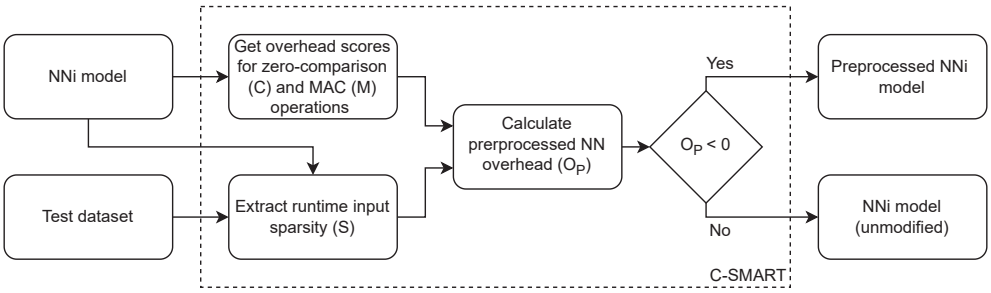

Figure 4: C-SMART preprocessor process flow.

Simple and Preprocessed NNi. Let the overhead of a zero-comparison operation associated with a single input value be assigned a score of C. Let the overhead of a MAC operation associated with a single input value be assigned a score of M. The overhead score is assumed to proportionally represent the quantity of overheads, such as energy or execution time, associated with an operation. The overhead score associated with different operations is assumed to be independent.

Let the input sparsity, i.e., the number of zero-valued inputs divided by the total number of inputs, be S, where $S \in \mathbb{R}$ and $0 \leq S \leq 1$. In the Simple NNi, the total overhead score per input value $\left(TO_{Simple}\right)$ is also equal to M as the MAC operations associated with all the inputs are assumed to be executed. With preprocessing, the total overhead score per input value $\left(TO_P\right)$ is calculated as shown in Equation 2.

$$TO_P = C \times S + (C + M)(1 - S) \tag{1}$$

$$TO_P = C + M(1 - S) \tag{2}$$

Then, the overhead associated with preprocessing alone $\left(O_P\right)$ can be calculated as shown in Equations 3 and 5.

$$O_P = TO_P - TO_{Simple} \tag{3}$$

$$O_P = C + M(1 - S) - M \tag{4}$$

$$O_P = C - MS \tag{5}$$

If C and M are constants in Equation (5), then the overhead associated with preprocessing alone $\left(O_P\right)$ decreases as sparsity increases. If $O_P > 0$, then the Preprocessed NNi is said to incur higher overhead when compared to the Simple NNi. If $O_P = 0$, then the Preprocessed NNi is said to incur the same overhead as the Simple NNi. If $O_P < 0$, then the Preprocessed NNi is said to incur lower overhead when compared to the Simple NNi. When $C < M$, which is the most common scenario in general-purpose computing implementations, if $S = 1$ or $S > \frac{C}{M}$, then the overhead of the Preprocessed NNi is not positive i.e., $O_P < 0$. Under this condition, preprocessing the NNi model is more beneficial and the benefits would triple for the TMR version. The C-SMART preprocessor uses this condition in its process flow shown in Figure 4. Algorithm 1 shows the pseudocode of the C-SMART implemented in our study.

Algorithm 1 C-SMART implementation pseudocode

```
Require: C, M ∈ ℝ and C, M > 0
  C ← Single zero-comparison operation overhead;
  M ← Single MAC operation overhead;
  NN ← Fully connected neural network model;
  Test_images ← Test dataset from MNIST;
  Total_inputs = 0; ▷ Initializing the variable for counting
    the total number of inputs during inferences with all
    Test_images
  Zero_inputs = 0; ▷ Initializing the variable for counting
    the total number of zero-valued inputs during inferences
    with all Test_images
  procedure PREPROCESS(NN_model)
      ▷ Can be implemented using the macros in C lan-
        guage ◁
      Replace:
      Result_buffer += input_value × weight_value;
      ▷ MAC operation ◁
      with:
      if input_value ≠ 0 then
          Result_buffer += input_value×weight_value;
      ▷ Execute MAC operation if input value is non-zero ◁
      in: NN_model
      return NN_model
  for an (image) in the (Test_images) do
      NNi ← (NN) inference with (image);
      Input_Array ← All input_value in NNi;
      ▷ Get the inputs to all neurons during this inference ◁
      Total_inputs += size of Input_Array;
      ▷ Count and accumulate the total number of
        input_value in this inference ◁
      Zero_inputs += number of 0's in Input_Array;
      ▷ Count and accumulate the total number of
        input_value = 0 in this inference ◁
  ▷ Calculating sparsity (S) for the given NN model ◁
Ensure: S ∈ ℝ and 0 ≤ S ≤ 1
  S = Zero_inputs/Total_inputs;
  if (C < M) and (S > C/M) then
      output PREPROCESS(NN);
  else
      output NN;
```

Table 2
Input sparsity throughout the different NN layers.

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

Input sparsity for various layers of the NN employed during experiments was calculated using all the 10,000 MNIST test images and shown in Table 2. The overall input sparsity for input values throughout all the layers (S) is 0.774, which means that around 77.4% of all the input values in our NN are zero. Let us define the overhead scores C and M, of the zero-comparison and MAC operations, respectively, as the number of assembly instructions associated with each. They are obtained by analyzing the corresponding executables with Objdump [24].

In our case, $S = 0.774$, $C = 16$ and $M = 175$. Then $O_P = -119$ according to Equation (5) i.e., the overhead of the Preprocessed NNi is not positive. The quantity 119 for O_P is 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. Hence, more than 10 million instructions are skipped while computing an inference with the preprocessed NN model. Therefore, C-SMART converts the input NNi into Preprocessed NNi for added performance and reliability benefits.

5. Experiments, results and discussion

Execution time measurements, energy measurements and radiation experiments were conducted with all four case study algorithms and the results are shown in Tables 3 to 5.

The execution time consumed by the nRF52840 DK for performing inferences was measured using the Data Watchpoint and Trace Unit (DWT) [1] available in the SoC of the nRF52840 DK. The DWT consists of a counter that counts the CPU clock cycles, and the counter is read afore and after the region of interest in the code (such as the inference function) whose execution time needs to be measured. The difference between the values read afore and after the region of interest provides the execution time in clock cycles, which was measured for all four case study algorithms with 250 input images in the input dataset and the mean of those 250 measurements are displayed in Table 3. Due to skipped instructions in Preprocessed NNi, the execution time was reduced by 4.5 million cycles, about 70 ms per inference in nRF52840 DK operating at 64 MHz, compared to Simple NNi. This effect is tripled in the TMR version as expected.

The energy consumed by the nRF52840 DK (operating at 3.0 V) for executing inferences using all four of the case study algorithms was measured with the help of the Joulescope (model JS110) [31], and the block diagram of the experimental setup is portrayed in Figure 5. The experiment was conducted with an nRF52840 DK in the DEFAULT mode to facilitate running the experiment in an automated manner. Automation is achieved by using a control script

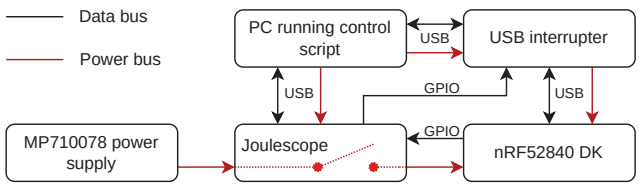

Figure 5: Block diagram of the energy overhead experimental setup.

Table 3
Average number of CPU cycles (execution time) in millions and energy in millijoules required for executing an inference with different NN versions.

NN Versions	mean Mcycles	mean mJ
Simple	12.7	5.1
Preprocessed	8.2	3.3
TMR	37.5	15.8
TMR Preprocessed	23.5	9.8

written in Python, which creates one executable for each of the four case study algorithms with each of the 250 images of the input dataset, resulting in a total of 1000 executables.

Each compiled executable is then used by the control script to program the nRF52840 DK using command line tools. Once the nRF52840 DK is programmed, the USB power to the board is disconnected using a USB interrupter module, which is controlled using the general-purpose output of the Joulescope. Then the Joulescope allows the power flow from the Multicomp pro MP710078 power supply [49] to the nRF52840 DK, through the Joulescope. Now, the Joulescope starts collecting samples at 2MHz. The samples collected by the Joulescope consist of electrical current and voltage data at the power input port to which the power supply is connected, along with the state of the Joulescope's general purpose inputs.

A general purpose output in nRF52840 DK is configured to create a pulse before the first inference and after the last inference, which is connected to one of the Joulescope's general purpose inputs, which is also sampled at 2MHz. The falling edge of the former pulse and the rising edge of the latter pulse are detected by analyzing the collected samples. The samples between those edges are used to calculate the energy consumed for 50 consecutive inferences rather than just one inference to increase the task length and make the sampling error insignificant (cf., Section 3.4.3 in [28]). This value is averaged to get the average energy consumption for a single inference. About 250 such energy consumption values are generated for each case study algorithm and the mean of those 250 values are displayed in Table 3. The table suggests that the Preprocessed NNi consumes 1.8 mJ less execution energy than Simple NNi. This advantage tripled in the TMR version altogether due to preprocessing as expected. Overall, the preprocessing reduced energy and timing consumption for executing inference by more than 35%.

Table 4
Experimental results for Simple NNi and Preprocessed NNi.

NN Versions	Neutron fluence $10^{10} neutrons/cm^2$	Irradiation time h	Error count			Soft error cross-section $(10^{-10} cm^2)$			FIT
			Tolerable	Critical	Output	Tolerable	Critical	Output	Output
Simple	07.9212	04.9	25	1	26	3.1561	0.1262	3.2823	4.249
Preprocessed	26.1172	15.8	61	2	63	2.3356	0.0766	2.4122	3.123

Table 5
Experimental results for TMR NNi and TMR Preprocessed NNi

NN Versions	Neutron fluence $10^{10} neutrons/cm^2$	Irradiation time h	Masked error		Error count (Tolerable)	Soft error cross-section $(10^{-10} cm^2)$	FIT Output
			Probabilistic	Inference			
TMR	07.9212	04.9	25	1	1	0.1262	0.163
TMR Preprocessed	26.1172	15.8	62	2	1	0.0383	0.050

Figure 6: Aligning the nRF52840 DK (indicated in red) with the projected path of the neutron beam.

The radiation chamber and the test nRF52840 DK hardware are shown in Figure 6. The radiation experiment results are shown in Tables 4 and 5. The raw data from the radiation experiment was analysed using Python scripts. The analysis involves generating golden results; i.e., probabilistic and inference outputs obtained by executing the case study algorithms in an environment outside the radiation chamber and comparing them with the corresponding results obtained during the experiment under radiation. Different categories of errors observed during analysis are the following:

- **Tolerable error**: Probabilistic output mismatches without a mismatch in inference output within an inference are considered as one tolerable error.
- **Critical error**: Probabilistic output mismatches that also lead to a mismatch in inference output within an inference are considered as one critical error.
- **Output error**: The sum of all tolerable and critical errors is considered as the Output error count.
- **Masked error**: Successfully handled errors within the TMR versions due to the voting mechanism.

Combining the radiation facility log with the raw data timestamp provides the neutron fluence for each 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 [30].

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.

Failure In Time (FIT) represents the number of failures expected within a billion operating hours of a device. Let us assume that each Output error causes a failure. Then the FIT values for test hardware running each of the four case study algorithms were calculated by multiplying their corresponding soft error cross-section value with billion hours ($10^9\,h$) and the reference terrestrial neutron flux of $12.946\,cm^{-2}h^{-1}$ [30]. As expected, FIT values for the preprocessed algorithms are lower than their counterparts as shown in Tables 4 and 5.

6. Conclusion and future work

C-SMART was used to preprocess our proof-of-concept fully connected 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 compared to the (non-optimized) Simple NNi and TMR NNi algorithms. According to the analysis of the experimental results made publicly available [32], preprocessing the NN inference model with C-SMART:

- Improved reliability via decreasing the soft error cross-section of NN inference by 26% and TMR NN inference by 70%.

- Improved performance via reducing the average execution time and energy consumption 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. Analysis of the execution energy of the case study algorithms reveals the affinity of C-SMART towards lowering power consumption. Further experiments can improve the statistical significance of the results, and investigate the NNs with various sparsity and overhead conditions. 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.

CRediT authorship contribution statement

Anuj Justus Rajappa: Conceptualization; methodology; software; validation; formal analysis; investigation; data curation; writing – original draft; writing - review & editing; visualization. **Philippe Reiter:** Conceptualization; writing - review & editing; supervision; project administration; funding acquisition. **Paolo Rech:** Resources; funding acquisition; supervision; data curation. **Siegfried Mercelis:** Supervision. **Jeroen Famaey:** Supervision; funding acquisition; writing - review & editing.

Data availability

The experimental data is publicly available at https://doi.org/10.5281/zenodo.7962582

References

[1] ARM , (accessed on 14-Nov-2024). Chapter 9. data watchpoint and trace unit. Available online: https://developer.arm.com/documentation/ddi0439/b/Data-Watchpoint-and-Trace-Unit?lang=en.

[2] Nordic semiconductors , (accessed on 14-Nov-2024). nrf52840 dk. Available online: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk.

[3] SEGGER , (accessed on 14-Nov-2024). Segger compiler. Available online: https://wiki.segger.com/SEGGER_compiler.

[4] UK Research and Innovation (UKRI), Science and Technology Facilities Council (STFC) , (accessed on 14-Nov-2024). Chipir. Available online: https://www.isis.stfc.ac.uk/Pages/Chipir.aspx. URL: https://www.isis.stfc.ac.uk/Pages/Chipir.aspx.

[5] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. software available from tensorflow.org.

[6] Abich, G., Garibotti, R., Gava, J., Reis, R., Ost, L., 2022a. Impact of thread parallelism on the soft error reliability of convolution neural networks, in: 2022 IEEE 13th Latin America Symposium on Circuits and System (LASCAS), pp. 1–4. doi:10.1109/LASCAS53948.2022.9789088.

[7] Abich, G., Garibotti, R., Reis, R., Ost, L., 2022b. The impact of soft errors in memory units of edge devices executing convolutional neural networks. IEEE Transactions on Circuits and Systems II: Express Briefs 69, 679–683. doi:10.1109/TCSII.2022.3141243.

[8] Abich, G., Gava, J., Garibotti, R., Reis, R., Ost, L., 2021a. Applying lightweight soft error mitigation techniques to embedded mixed precision deep neural networks. IEEE Transactions on Circuits and Systems I: Regular Papers 68, 4772–4782. doi:10.1109/TCSI.2021.3097981.

[9] Abich, G., Gava, J., Reis, R., Ost, L., 2020. Soft error reliability assessment of neural networks on resource-constrained iot devices, in: 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 1–4. doi:10.1109/ICECS49266.2020.9294951.

[10] Abich, G., Reis, R., Ost, L., 2021b. The impact of precision bitwidth on the soft error reliability of the mobilenet network, in: 2021 IEEE 12th Latin America Symposium on Circuits and System (LASCAS), pp. 1–4. doi:10.1109/LASCAS51355.2021.9667153.

[11] Apicella, A., Donnarumma, F., Isgrò, F., Prevete, R., 2021. A survey on modern trainable activation functions. Neural Networks 138, 14–32. doi:https://doi.org/10.1016/j.neunet.2021.01.026.

[12] Aponte-Moreno, A., Restrepo-Calle, F., Pedraza, C., 2019. Using approximate computing and selective hardening for the reduction of overheads in the design of radiation-induced fault-tolerant systems. Electronics 8, 1539. doi:10.3390/electronics8121539.

[13] Azizimazreah, A., Gu, Y., Gu, X., Chen, L., 2018. Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs, in: 2018 IEEE International Conference on Networking, Architecture and Storage (NAS), pp. 1–10. doi:10.1109/NAS.2018.8515692.

[14] Bandeira, V., Rosa, F., Reis, R., Ost, L., 2019. Non-intrusive fault injection techniques for efficient soft error vulnerability analysis, in: 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC), pp. 123–128. doi:10.1109/VLSI-SoC.2019.8920378.

[15] Capotondi, A., Rusci, M., Fariselli, M., Benini, L., 2020. Cmix-nn: Mixed low-precision cnn library for memory-constrained edge devices. IEEE Transactions on Circuits and Systems II: Express Briefs 67, 871–875. doi:10.1109/TCSII.2020.2983648.

[16] Cazzaniga, C., Frost, C.D., 2018. Progress of the scientific commissioning of a fast neutron beamline for chip irradiation. Journal of Physics: Conference Series 1021, 012037. URL: https://www.doi.org/10.1088/1742-6596/1021/1/012037, doi:10.1088/1742-6596/1021/1/012037.

[17] Chen, J., Klein, J., Wu, Y., Xing, S., Flammang, R., Heibel, M., Zuo, L., 2016. A thermoelectric energy harvesting system for powering wireless sensors in nuclear power plants. IEEE Transactions on Nuclear Science 63, 2738–2746. doi:10.1109/TNS.2016.2606090.

[18] Chen, Z., Li, G., Pattabiraman, K., 2021. 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. doi:10.1109/DSN48987.2021.00018.

[19] CNET, (accessed on 14-Nov-2024). Meet tesla's self-driving car computer and its two ai brains. Available online: https://www.cnet.com/tech/computing/meet-tesla-self-driving-car-computer-and-its-two-ai-brains/.

[20] Esposito, S., Avramenko, S., Violante, M., 2016. On the consolidation of mixed criticalities applications on multicore architectures, in: 2016 17th Latin-American Test Symposium (LATS), pp. 57–62. doi:10.1109/LATW.2016.7483340.

[21] Furano, G., Meoni, G., Dunne, A., Moloney, D., Ferlet-Cavrois, V., Tavoularis, A., Byrne, J., Buckley, L., Psarakis, M., Voss, K.O., Fanucci, L., 2020. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. IEEE Aerospace and Electronic Systems Magazine 35, 44–56. doi:10.1109/MAES.2020.3008468.

[22] Gava, J., Hanneman, A., Abich, G., Garibotti, R., Cuenca-Asensi, S., Bastos, R.P., Reis, R., Ost, L., 2023. A lightweight mitigation technique for resource-constrained devices executing dnn inference models under neutron radiation. IEEE Transactions on Nuclear Science , 1–1doi:10.1109/TNS.2023.3262448.

[23] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K., 2022. A survey of quantization methods for efficient neural network inference. Low-Power Computer Vision , 291–326doi:10.1201/9781003162810-13.

[24] GNU, (accessed on 14-Nov-2024). Objdump. Available online: https://ftp.gnu.org/old-gnu/Manuals/binutils-2.12/html_node/binutils_6.html#SEC6.

[25] Hamming, R.W., 1950. Error detecting and error correcting codes. The Bell System Technical Journal 29, 147–160. doi:10.1002/j.1538-7305.1950.tb00463.x.

[26] Hoang, L.H., Hanif, M.A., Shafique, M., 2020. Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation, in: 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1241–1246. doi:10.23919/DATE48585.2020.9116571.

[27] Hsiao, M.Y., 1970. A class of optimal minimum odd-weight-column sec-ded codes. IBM Journal of Research and Development 14, 395–401. doi:10.1147/rd.144.0395.

[28] Huybrechts, T., Reiter, P., Mercelis, S., Famaey, J., Latré, S., Hellinckx, P., 2021. Automated testbench for hybrid machine learning-based worst-case energy consumption analysis on batteryless iot devices. Energies 14. doi:10.3390/en14133914.

[29] Imperas Software, (accessed on 14-Nov-2024). Open Virtual Platforms - the source of Fast Processor Models and Platforms. Available online: https://www.ovpworld.org/technology_ovpsim.

[30] JEDEC, (accessed on 14-Nov-2024). 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.

[31] Joulescope, (accessed on 14-Nov-2024). Joulescope™ js110 user's guide. Available online: https://download.joulescope.com/docs/JoulescopeUsersGuide/JoulescopeUsersGuide_v1_1.pdf.

[32] Justus, A., 2023. 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. URL: https://doi.org/10.5281/zenodo.7962582, doi:10.5281/zenodo.7962582.

[33] Kim, B.K., 1999. Reliability analysis of real-time controllers with dual-modular temporal redundancy, in: Proceedings Sixth International Conference on Real-Time Computing Systems and Applications. RTCSA'99 (Cat. No. PR00306), IEEE. pp. 364–371.

[34] Koren, Su, 1979. Reliability analysis of n-modular redundancy systems with intermittent and permanent faults. IEEE Transactions on Computers C-28, 514–520. doi:10.1109/TC.1979.1675397.

[35] Kulida, E., Lebedev, V., 2020. About the use of artificial intelligence methods in aviation, in: 2020 13th International Conference "Management of large-scale system development" (MLSD), pp. 1–5. doi:10.1109/MLSD49919.2020.9247822.

[36] Lai, L., Suda, N., Chandra, V., 2018. Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. arXiv preprint arXiv:1801.06601 .

[37] Lakrouni, S., Sebgui, M., Bah, S., 2022. Using ai and iot at the edge of the network, in: 2022 8th International Conference on Optimization and Applications (ICOA), pp. 1–6. doi:10.1109/ICOA55659.2022.9934603.

[38] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324. doi:10.1109/5.726791.

[39] Libano, F., Rech, P., Neuman, B., Leavitt, J., Wirthlin, M., Brunhaver, J., 2021. How reduced data precision and degree of parallelism impact the reliability of convolutional neural networks on fpgas. IEEE Transactions on Nuclear Science 68, 865–872. doi:10.1109/TNS.2021.3050707.

[40] Libano, F., Wilson, B., Anderson, J., Wirthlin, M.J., Cazzaniga, C., Frost, C., Rech, P., 2019. Selective hardening for neural networks in fpgas. IEEE Transactions on Nuclear Science 66, 216–222. doi:10.1109/TNS.2018.2884460.

[41] Libano, F., Wilson, B., Wirthlin, M., Rech, P., Brunhaver, J., 2020. Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on fpgas. IEEE Transactions on Nuclear Science 67, 1478–1484. doi:10.1109/TNS.2020.2983662.

[42] Liu, Z., Liu, Y., Chen, Z., Guo, G., Wang, H., 2021. Analyzing and increasing soft error resilience of deep neural networks on arm processors. Microelectronics Reliability 124, 114331. doi:https://doi.org/10.1016/j.microrel.2021.114331.

[43] Lyons, R.E., Vanderkulk, W., 1962. The use of triple-modular redundancy to improve computer reliability. IBM Journal of Research and Development 6, 200–209. doi:10.1147/rd.62.0200.

[44] Ma, D., Zhang, S., Jiao, X., 2023. Robust hyperdimensional computing against cyber attacks and hardware errors: A survey, in: Proceedings of the 28th Asia and South Pacific Design Automation Conference, Association for Computing Machinery, New York, NY, USA. p. 598–605. doi:10.1145/3566097.3568355.

[45] Mahmoud, A., Hari, S.K.S., Fletcher, C.W., Adve, S.V., Sakr, C., Shanbhag, N., Molchanov, P., Sullivan, M.B., Tsai, T., Keckler, S.W., 2020. Hardnn: Feature map vulnerability evaluation in cnns. doi:10.48550/arXiv.2002.09786, arXiv:2002.09786.

[46] Mahmoud, A., Sastry Hari, S.K., Fletcher, C.W., Adve, S.V., Sakr, C., Shanbhag, N., Molchanov, P., Sullivan, M.B., Tsai, T., Keckler, S.W., 2021. Optimizing selective protection for cnn resilience, in: 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), pp. 127–138. doi:10.1109/ISSRE52982.2021.00025.

[47] Masadeh, M., Hasan, O., Tahar, S., 2019. Input-conscious approximate multiply-accumulate (mac) unit for energy-efficiency. IEEE Access 7, 147129–147142. doi:10.1109/ACCESS.2019.2946513.

[48] Merenda, M., Porcaro, C., Iero, D., 2020. Edge machine learning for ai-enabled iot devices: A review. Sensors 20. doi:10.3390/s20092533.

[49] Multicomp pro, (accessed on 14-Nov-2024). Bench Top Remote Programmable Linear Mode Power Supply (MP710078). Available online: https://www.farnell.com/datasheets/2830052.pdf.

[50] Nazar, G.L., Kopper, P.H., Leipnitz, M.T., Juurlink, B., 2021. Lightweight dual modular redundancy through approximate computing, in: 2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC), pp. 1–8. doi:10.1109/SBESC53686.2021.9628356.

[51] Parashar, A., Rhu, M., Mukkara, A., Puglielli, A., Venkatesan, R., Khailany, B., Emer, J., Keckler, S.W., Dally, W.J., 2017. Scnn: An accelerator for compressed-sparse convolutional neural networks. SIGARCH Comput. Archit. News 45, 27–40. doi:10.1145/3140659.3080254.

[52] Possamai Bastos, R., Dutertre, J.M., Garay Trindade, M., Viera, R.A.C., Potin, O., Letiche, M., Cheymol, B., Beaucour, J., 2020. Assessment of on-chip current sensor for detection of thermal-neutron-induced transients. IEEE Transactions on Nuclear Science 67, 1404–1411. doi:10.1109/TNS.2020.2975923.

[53] Possamai Bastos, R., et al., 2022. Assessment of tiny machine-learning computing systems under neutron-induced radiation effects. IEEE Transactions on Nuclear Science 69, 1683–1690. doi:10.1109/TNS.2022.3176485.

[54] Rajappa, A.J., Reiter, P., Sartori, T.K.S., Laurini, L.H., Fourati, H., Mercelis, S., Famaey, J., Bastos, R.P., 2023. Smart: Selective mac zero-optimization for neural network reliability under radiation. Microelectronics Reliability , 115092doi:https://doi.org/10.1016/j.microrel.2023.115092.

[55] Ramos, A., Carrasco, A., Fontanet, J., Herranz, L., Ramos, D., Díaz, M., Zazo, J., Cabellos, O., Moraleda, J., 2024. Artificial intelligence and machine learning applications in the spanish nuclear field. Nuclear Engineering and Design 417, 112842. doi:https://doi.org/10.1016/j.nucengdes.2023.112842.

[56] Reviriego, P., Maestro, J.A., Baeg, S., Wen, S., Wong, R., 2010. Protection of memories suffering mcus through the selection of the optimal interleaving distance. IEEE Transactions on Nuclear Science 57, 2124–2128. doi:10.1109/TNS.2010.2042818.

[57] Sabbagh, M., Gongye, C., Fei, Y., Wang, Y., 2019. Evaluating fault resiliency of compressed deep neural networks, in: 2019 IEEE International Conference on Embedded Software and Systems (ICESS), pp. 1–7. doi:10.1109/ICESS.2019.8782505.

[58] Schorn, C., Guntoro, A., Ascheid, G., 2018. Efficient on-line error detection and mitigation for deep neural network accelerators, in: Gallina, B., Skavhaug, A., Bitsch, F. (Eds.), Computer Safety, Reliability, and Security, Springer International Publishing, Cham. pp. 205–219. doi:10.1007/978-3-319-99130-6_14.

[59] Shi, W., Dustdar, S., 2016. The promise of edge computing. Computer 49, 78–81. doi:10.1109/MC.2016.145.

[60] Su, F., Liu, C., Stratigopoulos, H.G., 2023. Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives. IEEE Design & Test 40, 8–58. doi:10.1109/MDAT.2023.3241116.

[61] Su, W., Li, L., Liu, F., He, M., Liang, X., 2022. Ai on the edge: a comprehensive review. Artificial Intelligence Review 55, 6125–6183. doi:10.1007/s10462-022-10141-4.

[62] Vargas, F., Nicolaidis, M., 1994. Seu-tolerant sram design based on current monitoring, in: Proceedings of IEEE 24th International Symposium on Fault- Tolerant Computing, pp. 106–115. doi:10.1109/FTCS.1994.315652.

[63] Vega, A., Bose, P., Buyuktosunoglu, A., 2017. Chapter 1 - introduction and chapter 2 - reliable and power-aware architectures: Fundamentals and modeling, in: Vega, A., Bose, P., Buyuktosunoglu, A. (Eds.), Rugged Embedded Systems. Morgan Kaufmann, Boston, pp. 1–37. doi:https://doi.org/10.1016/B978-0-12-802459-1.00001-4.

[64] Wakerly, J., 1976. Microcomputer reliability improvement using triple-modular redundancy. Proceedings of the IEEE 64, 889–895. doi:10.1109/PROC.1976.10239.

[65] Wang, H.B., Wang, Y.S., Xiao, J.H., Wang, S.L., Liang, T.J., 2021. Impact of single-event upsets on convolutional neural networks in xilinx zynq fpgas. IEEE Transactions on Nuclear Science 68, 394–401. doi:10.1109/TNS.2021.3062014.

[66] Watanabe, M., Watanabe, M., 2019. Full-hardware triple modular and penta-modular redundancies using a high frequency majority voting operation, in: 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 177–181. doi:10.1109/APCCAS47518.2019.8953122.

[67] Weng, O., 2021. Neural network quantization for efficient inference: A survey. arXiv preprint arXiv:2112.06126 doi:10.48550/arXiv.2112.06126.

[68] Yang, G.C., 1995. Reliability of semiconductor rams with soft-error scrubbing techniques. IEE Proceedings - Computers and Digital Techniques 142, 337–344(7). URL: https://digital-library.theiet.org/content/journals/10.1049/ip-cdt_19952162.

Anuj Justus Rajappa is a PhD researcher at imec-IDLab affiliated with the University of Antwerp, Belgium, working in the field of Reliable, Energy-Aware Intelligent Edge Systems. His focus is on reliable machine learning for embedded systems that are sustainable, battery-less, wireless, and autonomous. He has presented his work through various national and international platforms, including ESA-SPAICE, ESREF, RADECS, RISC-V Global Forum, and ESI-Scilab.

Phil Reiter is a senior researcher and project manager with imec and the University of Antwerp, Belgium. He received his M.Sc. in Machine Learning and Deep Learning from the University of Strathclyde in Glasgow, UK. He has over a decade of industry experience as a technical lead in semiconductor design verification and business communications with Advanced Micro Devices, Inc. (AMD) in Toronto, Canada. His primary research interest is on biologically inspired and very low-power AI systems.

Paolo Rech received his master and Ph.D. degrees from Padova University, Padova, Italy, in 2006 and 2009, respectively. He has been an associate professor at Università di Trento in Italy since 2022 and was an associate professor at UFRGS in Brazil since 2012. He is the 2019 Rosen Scholar Fellow at the Los Alamos National Laboratory, he received the 2020 Impact in Society award from the Rutherford Appleton Laboratory, UK. In 2020 Paolo was awarded the Marie Curie Fellowship at Politecnico di Torino, in Italy. His main research interests include the evaluation and mitigation of radiation-induced effects in autonomous vehicles for automotive applications and space exploration, in large-scale HPCs, and quantum computers.

Siegfried Mercelis is an assistant professor at the University of Antwerp. He has master's degrees in music production and engineering (electronics and ICT). From 2012 to 2016 he was employed at Van den Berghe R&D under a Baekeland PhD mandate on optimizing and parallelizing real-time media applications. In December 2016 he obtained his PhD in applied engineering at the University of Antwerp, where he is currently employed as assistant professor and manager of the adaptive intelligence program. His team of 30+ AI researchers is committed to bridging the gap between academic AI research and industry in domains such as chemical process control, autonomous shipping, smart buildings, logistics and mobility.

Jeroen Famaey is an associate research professor at imec and the University of Antwerp, Belgium, and a member of the IDLab research group, where he leads the Connected Systems team. He received his M.Sc. in Computer Science from Ghent University, Belgium, in 2007. Subsequently, he obtained my PhD in Computer Science Engineering at the same university in 2012. His current research interests are in future wireless networks and connected applications.