Effects of thermal neutron radiation on a hardware-implemented machine learning algorithm

Matheus Garay Trindade, Fabio Benevenuti, M. Letiche, J. Beaucour, Fernanda Lima Kastensmidt, Rodrigo Possamai Bastos

Effects of Thermal Neutron Radiation on a Hardware-Implemented Machine Learning Algorithm

M. Garay Trindade[a], F. Benevenuti[b], M. Letiche[c], J. Beaucour[c], F. Kastensmidt[b], R. Possamai Bastos[a]

[a] *Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, 38000 Grenoble, France * Institute of Engineering Univ. Grenoble Alpes*

[b] *Universidade Federal do Rio Grande do Sul, 90040-060, Porto Alegre, Brazil*

[c] *Institut Laue-Langevin, 38000, Grenoble, France*

Abstract

Hardware-implemented machine learning algorithms are finding their way in various domains, including safety-critical applications. This has demanded these algorithms to perform correctly even in harsh environmental conditions, such as in avionics altitudes. Support Vector Machine (SVM) is an important Machine Learning that has been target of hardware implementation in recent years. This is the first work to asses both Binary and Multiclass SVMs under thermal neutron radiation, a type of particle noticeably present in high altitudes. A fault injection campaign along with a radiation test with the D50 thermal neutron source, at the Intitut Laue-Languevin, has been performed. The results show a high intrinsic fault tolerance for both varieties of the SVM algorithm, especially for the Multiclass SVM.

1. Introduction

Machine learning algorithms have been regarded by both the academic and industrial domains as performant tools for learning how to predict future outcomes from existing data. As these algorithms are capable to provide very accurate classifiers, they are making their way in a myriad of domains, such as medic [1], robotics [2] and even on geoscience/aerospace [3]. In all these fields, an algorithm is needed to identify a pattern in raw data and, based on what has been identified, perform an action.

Support Vector Machines (SVM) [4] is popular a machine learning algorithm used in data mining and pattern recognition. SVM is capable of performing both classification and regression with high generalization capacity. The SVM algorithm works by deriving a linear separator from a set of labeled data during a training phase and then using this classifier to label a new unknown data, which makes SVMs commonly used in classification tasks. With the ever-increasing amount of data, an open issue is finding alternative platforms and implementations for the algorithm to improve its execution time so it is able to cope with rising performance requirements.

Field-Programmable Gate Arrays are common platforms for algorithm acceleration, which has made the architecture a target for SVM implementation [5]. On the other hand, FPGAs are known to be sensitive to radiation effects [6], notoriously Static Random-Access Memory (SRAM)-based FPGAs. As the name suggests, the configuration bitstream is stored in a SRAM, which may be corrupted by particle-induced transients, an effect known as Singe Event Upset (SEUs). Notably, transient faults in the configuration memory of an FPGA may change the circuit routing, the values of the Lookup Table (LUT) and the data inside the Block RAM memory (BRAM). In these cases, the FPGA must be reconfigured as the effects will remain unless they are overwritten. Other effect that may arise is a bit inversion of a Flip-Flop (FF) of a Configuration Logic Block (CLB) of the user's sequential logic. These faults are corrected by the next load to the FF.

The effects of radiation induced transients have been studied in multiple hardware implementations of machine learning algorithms. Artificial Neural Networks (ANNs) and Convolutional Neural Networks have been analyzed in [7, 8, 9], while [10] focused on Bayesian Machines and [11] on Binary SVM. To the best of our knowledge, no work has been conducted on the effects of thermal neutrons on hardware implemented SVMs nor the effects of radiation in Multiclass SVMs. In this work, we first-handedly investigate these effects by conducting a thermal neutron radiation campaign and an extensive fault injecting campaign on a Binary and a Multiclass SVM to better understand how the radiation affects them. Furthermore, we compare the results of a Multiclass SVM to the ones of a Binary SVM. The radiation test campaign has been performed using the D50 thermal neutron source at the Institut Laue-Languevin [12]. The fault injection campaign was based on partial configuration of the FPGA bitstream. Both campaigns made use of a Zynq-7000 System-on-a-Chip (SoC) as test vehicle.

*Corresponding author: matheus.trindade@univ-grenoble-alpes.fr

February 18, 2021

2. SVM Algorithm

This section describes the SVM algorithm and its use cases. Moreover, aspects to be considered for a hardware implementation are presented.

2.1. SVM Fundamentals

SVM is a supervised machine learning algorithm [4] that is used to perform both classification and regression. The technique has been originally designed to solve binary classification problems, i. e. classifying an observation (herein an input sample) as one of the two possible classes. Being a supervised learning algorithm, its execution is in two phases: training and classification.

Mathematically, an input sample is an n-dimensional vector $\vec{x} \in \mathbb{R}^n$, that is to be classified in one of two possible classes, modeled as $\{1, -1\}$. During the training phase, the algorithm finds the multidimensional plane, i. e. hyperplane, that better describes the difference between the classes based on a set of labeled samples, i. e. samples whose class is known beforehand or also referred to as training set. The hyperplane equation is shown in Equation 1.

$$score(\vec{x}) = \sum_{i=1}^{n} \alpha_i y_i (\vec{x_i} \cdot \vec{x}) + b \tag{1}$$

The support vectors $\vec{x_i} \in \mathbb{R}^n$, weights $\alpha_i \in \mathbb{R}$, support vector label $y_i \in \{1, -1\}$, and bias $b \in \mathbb{R}$ are found during the training phase. When evaluating an input sample $\vec{x}$ in Equation 1, the output is a score. If the score is positive, the class assigned is 1, -1 otherwise.

2.2. Multiclass SVM

Originally, SVM was created to perform binary classification. However, it may be extended to fit multiclass problems. This is achieved by dividing the dataset into subsets so it is possible to use binary classification. The two most popular techniques are One-vs-One and One-vs-All. In this work, we made use of One-vs-One, which is explained in the sequence. One-vs-All was left out for the sake of brevity.

In the One-vs-One approach, a binary classifier is trained for each pair of classes. For instance, if there are three possible classes, e.g. $\{A, B, C\}$, a classifier is trained only with samples from classes $\{A, B\}$ in order to classify unknown samples between classes A and B. Another one is trained with samples from classes $\{A, C\}$ to classify an unknown simple into either A or C. A classifier for $\{B, C\}$ is also trained. To infer a class for an unknown sample, it is evaluated on the three trained SVMs, each one inferring one class. The class that is the most inferred is chosen as the final class. For example, if SVM $\{A, B\}$ outputs A, SVM $\{A, C\}$ outputs A and SVM $\{B, C\}$ outputs B, as A has been inferred twice whereas B was only inferred once and C has not been inferred, the final class is A

2.3. Hardware-Implemented SVM Algorithm

In some applications, the SVM has a high throughput requirement, needing to constantly classify input samples, leading to energy and resource consumption. Implementing the SVM in hardware is a powerful alternative to enhance the algorithm performance and potentially saving energy.

Both the training and classification phases of the algorithm have been target of hardware implementation as an Application Specific Integrated Circuit (ASIC) in [13, 14] an in FPGA [15, 5] aiming towards better performance. Other works have focused only on the classification phase [14] [5], conducting the training phase in software platforms as MATLAB and LibSVM [16].

3. Case-Study SVM Architectures

In this Section, we present the SVM architecture implemented for this work. We focused on the classification part.

3.1. State-of-the-Art SVM in Hardware

Various authors have proposed solutions for hardware acceleration of the SVM classification phase [5, 13, 14]. In [14] the dot product between Support Vectors and the input sample are executed in parallel to achieve maximum performance required by their application (voltage-droop prediction). The limitation of this approach is being able to generate only linear and second-order polynomial classifiers. In [5] and [13], the authors implement non-linear classifiers by making use of CORDIC (COordinate Rotation DIgital Computer) for approximating non-linear function through hardware-friendly functions (e.g., shift and sum operations). Following the idea in [14], the operations with the Support Vectors and input samples is also performed in parallel in [5] and [13]. Moreover, implementations based on CORDIC algorithms have higher memory requirements when compared with [14] due to the need of storing lookup tables along with requiring multiple clock cycles to output the result.

3.2. Binary SVM Architecture Design

As both datasets used in this work are linearly separable, to be described in Subsection 3.4, we decided to follow the implementation of [14] due to its performance benefits. Figure 1 shows the case-study SVM architecture. Furthermore, the products $\alpha_i y_i$ have been calculated off-line and only the result has been stored on the FPGA, reducing one multiplication per SV $\vec{x_i}$. The designed circuit is fully combinatorial. The circuit is composed by three main parts: The Multipliers, the Adders and the Output, as illustrated in Figure 1.

In terms of data representation, a 16-bit fixed point with 8 bits for the real part was chosen, as suggested in [14, 5, 13]. This representation was confirmed to fit the

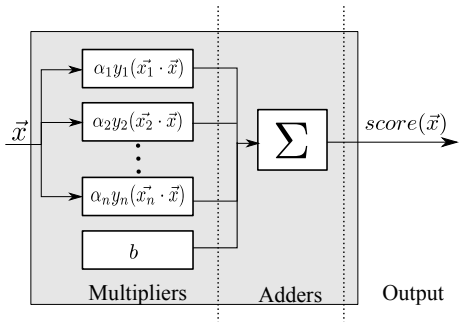


Figure 1: Overview of the hardware-implemented SVM architecture design.

datasets, avoiding overflows and maintaining sufficient precision through simulations. As the input samples in both datasets are 2-dimensional, input vectors are 32-bit wide. The primary output is composed of 49 bits to maintain calculation precision.

3.3. Multiclass SVM Architecture Design

As discussed in Subsection 2.2, a Multiclass SVM is composed by a collection of Binary SVMs aggregated by a voter. In this work, each Binary SVM was implemented following the description in Subsection 3.2, using the same value representation. The voter was not implemented in the FPGA, outputting the score of all the SVMs. It is up to the client application to evaluate the score and perform the voting. The designed circuit is also fully combinational.

3.4. Set of Input Vectors for the Binary SVM

The target set of input vectors (dataset) has been obtained from Monte-Carlo simulations representing current peaks and global delays obtained from golden integrated circuits (ICs) and faulty ICs [17]. The input vector is 2-dimensional, and 150 input vectors have been obtained from golden IC samples and 150 input vectors from faulty IC samples. The dimensions are thus:

- **Dimension 1**: global delay
- **Dimension 2**: current peak

This set of input vectors is sufficient to distinguish faulty asynchronous IC samples from fault-free asynchronous IC samples [17]. The set of input vectors has been partitioned into 2 subsets of the same size, one for training and another one for classification, each one with 75 golden IC samples and 75 faulty IC samples. A SVM model has been generated by using MATLAB. From this model, we have obtained the α's and their respective support vectors $\vec{x_i}$. In total, 50 support vectors $\vec{x_i}$ have been generated at the training phase. To better exercise the circuit, 116 random samples were added to the dataset.

3.5. Set of Input Vectors for the Multiclass SVM

For the Multiclass SVM, the dataset chosen was the Iris flowers [18]. It is originally a 4-dimensional dataset that contains 150 samples of three different species of Iris flowers (50 of each): setosa, virginica and versicolor. Only two dimensions have been kept for this experiment as they hold enough information for training a performant SVM classifier. The dimensions are:

- **Dimension 1**: Petal length
- **Dimension 2**: Petal width

This dataset has been partitioned in a training set and a classification set, with 75 samples (25 of each species) each. The training set has been used to train a One-vs-One Multiclass SVM, yielding three Binary SVMs, 2 SVMs with 2 Support Vectors $\vec{x_i}$ and one with 16 Support Vectors $\vec{x_i}$. Following the idea used in Subsection 3.4, 116 samples were added to better cover the architecture designed.

4. SVM Reliability Assessment Through Emulated Fault Injection

Emulated fault injection was used in this work to cross-validate results from radiation experiments and further investigate areas of improvement in the DUT.

Compared to emulated fault injection, accelerated irradiation experiments provide a better approximation to the use of the DUT in real environment and can provide a more comprehensive test coverage reaching all relevant structures of the integrated circuit. However, radiation experiments are less powerful in pinpointing the DUT submodules more susceptible to SEUs that could be candidate to mitigation strategies leveraging DUT reliability.

This section describes the fault injection methodology and the results obtained.

4.1. Fault Injection Set-up

The test vehicle used in fault injection is a ZedBoard development board which is equipped with a Xilinx Zynq-7000 MPSoC hosting the DUT. The Zynq-7000 device is divided in two main parts that are a dual core processor system (PS) based on Arm® Cortex®-A9 and a SRAM-based FPGA programmable logic (PL) that, for this device part number, uses technology equivalent to a Xilinx 7 Series Artix-7 FPGA.

The SVM computation core, implemented in VHDL, is wholly hosted in the FPGA (PL) side of the Zynq-7000. A small part of the application, implemented as software in C language, is hosted in the ARM (PS) side of the device and is responsible mostly by coordination and reporting tasks, not playing relevant role in the computation effort. However, between these two parts of software and SVM core, there is a communication infrastructure, based on AMBA AXI interface, with some modules also implemented in the FPGA side through the use of parameterizable IP design

blocks provided by the FPGA manufacturer. The resource utilization and frequency of operation for both the Binary and Multiclass SVMs are shown in Table 1.

	Binary	Multiclass
Frequency	133 MHz	133 MHz
LUT	7%	5%
LUTRAM	1%	1%
FF	2%	2%
BRAM	1%	1%
DSP	40%	28%
IO	1%	1%

Table 1: Resource utilization of Zynq-7000 for the Binary and Multiclass SVMs.

That accessory communication infrastructure using prebuilt IP blocks is relevant to fault injection because, while implemented in FPGA, it is also susceptible to SEUs, and, while implemented by third party vendors, it is not prone to modifications or improvements by mitigation techniques that could be used in the SVM core.

To tackle this difference, a slightly different bitstream was used in fault injection where the communication infrastructure and the SVM core where placed in different regions of the FPGA allowing to study the contribution of these two parts to the overall DUT reliability. With this approach, faults could be injected in either part, separately, to analyze the individual contributions for DUT reliability, or in both parts simultaneously, for comparability with radiation results. It is worth noting that the AXI communication infrastructure was present in radiation experiments, being evidenced here only for the convenience of the analysis of fault injection results.

To further accelerate fault injection campaigns, part of the diagnostic logic was embedded in the ARM processor. The ARM application and FPGA bitstream were loaded from the Flash memory present at the ZedBoard development board. The board reset was implemented by power cycling using an automated power switch controlled by the fault injection campaign script running at the host computer.

Finally, an additional module was implemented in the FPGA to support fault injection. This module is based on Xilinx Internal Configuration Access Port (ICAP) and allows communication with the host computer that also runs the fault injection campaign script. This fault injection module, coded in VHDL, was not present in radiation experiments. This setup used for fault injection is depicted in Figure 2.

4.2. Assessment Metrics

In this paper, bit-flips in the Configuration Random-Access Memory (CRAM) caused by radiation are referred to as fault. FF bit-flips are not possible as our design is fully combinational. Faults may halt the system execution or corrupt its outputs. When it halts the execution, it is

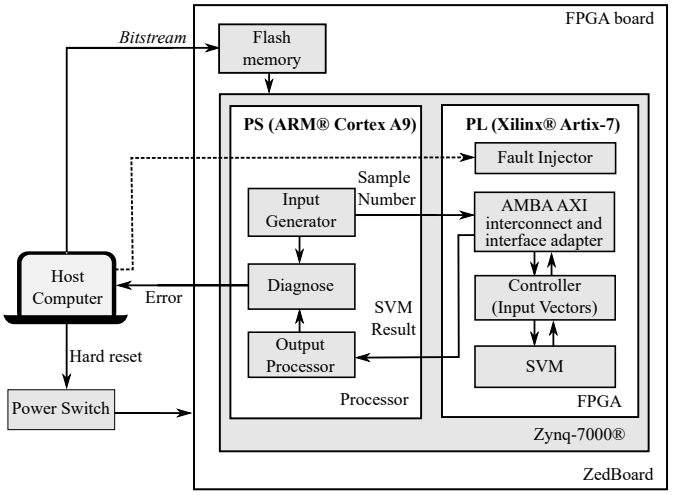


Figure 2: Zynq-7000 set-up under fault injection

defined herein as a **Crash Failure**. When it continues to output values, each output is classified as follows:

- **Masked Fault:** the output does not differ from the golden reference;
- **Tolerable Failure:** there is a mismatch between the expected value and the golden reference but the class assignment is still the correct, i.e. the sign bit is the same;
- **Critical Failure:** the fault resulted in misclassification of a sample.

Whenever a fault that has not crashed the system is detected, the entire dataset is run over the classifier. Each of the input samples score is logged for further classification into either *Critical Failure*, a *Tolerable Failure* or *Masked Fault*.

4.3. Fault Injection Methodology

The total volatile memory available at Zynq-7000 device used in these experiments, including cache and RAM memory used by the processor system and the data and configuration memory used by the FPGA, amounts to approximately 32 mebibits.

The Xilinx Vivado design synthesis tool reports a total of 24.5 mebibits in the FPGA side for the device in use, of which approximately 4,9 mebibits (20%) corresponds to memory available to user data in the form of BRAM. Conversely, the remaining 19.6 mebibits (80%) is mostly CRAM that holds configuration for BRAM blocks, DSP blocks, CLB blocks and configuration for all possible signal routing throughout the FPGA switch boxes. The fault injection tool used in this work is targeted specifically to that 80% of FPGA memory relative to the CRAM memory.

In Xilinx 7 Series FPGAs the memory is organized in *frames* of 101 words of 32 bits. A frame is the minimal

unit of CRAM memory that can be read or written to the FPGA using the Xilinx Internal Configuration Access Port (ICAP) hardware block available at Xilinx 7 Series FPGAs. This is the same hardware block shared with the initial configuration of the FPGA or the partial dynamic reconfiguration of the FPGA, but now used to read or write a single frame instead of loading or reading back the whole memory or a block of memory that holds a retargetable module.

The emulated fault injection can be implemented, therefore, by reading a single CRAM memory frame, changing its value and writing the frame back into the CRAM memory.

The injection flow used in this work [19] is presented in the sequence. The approach followed, named *random-accumulated*, consists in injecting faults in randomized positions in the CRAM memory region occupied by the DUT. After each fault injected the DUT is exercised with the completed set of input samples. The outputs are then logged according to 4.2. Those faults are accumulated in memory until a Crash failure is detected and then all faults are cleaned up by reprogramming the FPGA. This process is repeated until a significant number of events is collected. Then, a reliability curve is derived from the data. This curve is generated in order to visualize how the reliability falls when faults start to accumulate, i.e., the percentage of the dataset that was misclassified by the number of accumulated faults. This approach aims in emulating the accumulation of SEUs on the CRAM memory during the DUT operation, where one emulated fault injected would be equivalent to the number of particles given by the static cross section of the underlying device.

All the DUT modules targeted by fault injection were constrained to a rectangular CRAM memory region of 2,070 frames of 3,232 bits amounting to 6,690,240 bits. As an FPGA is a general-purpose device that is being programmed to a particular application, not all these CRAM bits are effectively used by the DUT as many CLB, BRAM, DSP blocks and most of the signal routing paths throughout the FPGA fabric remains unused. Those memory bits effectively required to program the FPGA to a particular application can be called *essential bits* [20]. In the case of the DUT in our experiments, the Xilinx Vivado synthesis tool reported a number of 1,059,559 essential bits in the case of the Binary SVM (Section 3.2) and 728,455 bits in the case of the Multiclass SVM (Section 3.3).

The diagnostic collected from DUT allowed the classification of each CRAM bit according to the criteria already defined in Section 4.2.

4.4. Results

During random-accumulated fault injection campaigns, the two DUT for Binary and Multiclass SVM were tested in three different physical floorplans on the FPGA, allowing faults to be injected separately on the SVM core, on the accessory communication modules and on all modules together.

Several fault injection campaigns were executed to explore the DUT reliability under accumulated faults amounting to more than 10^6 faults injected.

Although the rate of fault injection and the SEUs produced by thermal neutron irradiation shall differ at least by a factor relative to the device static cross section, the relative rank of the reliability curves is consistent among the experiments, as can be observed comparing Figure 3(a).

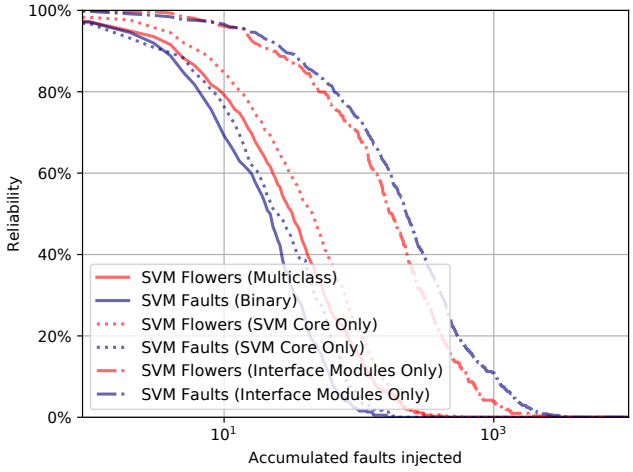
For the fault injection campaigns, whose results are presented on Figure 3(a), any failure, either tolerable or critical, were considered as a functional failure. Additional fault injection campaigns were executed with a relatively lax criteria where only critical failures, that is when the SVM produced an incorrect classification, was considered as a functional failure. These results are presented on Figure 3(b). This is a sound criterion when the SVM is used as a classifier because only the final SVM classification and its semantic meaning are relevant. It is worth noting that, in this implementation of SVM, the classification depends only on the signal bit of the numbers at the SVM primary output and not on the magnitude of those numbers.

In Figure 3(a) and (b), it is noticeable that the Binary SVM is less reliable than the Multiclass version, as the reliability curve falls quicker. In [11], the authors have showed that the Binary SVM has a level of intrinsic fault tolerance. A more in-depth discussion on the reasons in presented later in Subsection 5.5. In our experiment, we have shown that the Multiclass SVM may be even more reliable. It is worth noting that more experiments should be conducted, as other factors may have played a role, such as the different datasets used.

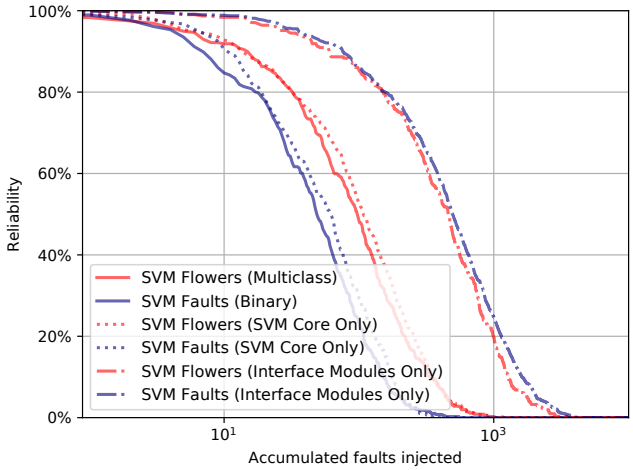
Another aspect explored using fault injection was the impact of the microprocessor interface logic compared to the main SVM computation core. For this, fault injection campaigns were executed injecting faults only over the SVM core and only over the interface modules. The results are also presented on Figure 3(a) and 3(b), where we can observe that the overall DUT reliability is dominated by the SVM core with a significantly higher reliability at the interface logic. This suggests that, despite the use of the third-party interface modules, there is still room for major improvements in the DUT reliability by implementing fault tolerance techniques to mitigate SEUs on the SVM core. It is to be noted that the BRAM was used by the interface modules in order to store the input samples. In a real-world situation, this would probably not be needed, as input samples would be generated by the environment, thus making the reliability numbers of the BRAM in our study not relevant.

5. Radiation Test Experiment and Results

This section describes the radiation experiments of both the Binary and Multiclass SVMs conducted with a thermal neutron source and an analysis of the obtained results.



(a) Fault injection, any failure



(b) Fault injection, critical failures only

Figure 3: SVM Reliability.

5.1. Radiation Test Set-Up

The thermal neutron test has been conducted at the Platform for Advanced Characterisation (PAC-G), hosted by Institut Laue-Languevin (ILL), using the D50 thermal neutron source. Previous papers [12] have demonstrated the relevance of the usage of this equipment to perform reliability testing. It provides a thermal neutron beam with a spectrum ranging from few microelectronvolts to around 100 meV, with a peak around 13 meV. The captured flux (i.e. equivalent flux of 25 meV) is adjustable from 0 to 10^{10} $neutrons\ /\ (s{\cdot}cm^2)$. To keep parts of the board other than circuit, the board has been protected by a polyethylene sheet with a hole above the target chip. Each SVM architecture was tested individually with a constant flux of $6.94 \cdot 10^6$ $neutrons\ /\ (s \cdot cm^2)$. The Binary SVM was tested for 2 hours and 20 minutes, yielding a total fluence of $5.83{\cdot}10^{10}$ $neutrons\ /\ cm^2$, and the Multiclass SVM was tested for 3 hours and 8 minutes, giving a total fluence of $7.833 \cdot 10^{10}$ $neutrons\ /\ cm^2$.

In order to test both of the SVM architectures (Binary and Multiclass) designed, we have made use of the setup is illustrated in Figure 4. The test vehicle used was a ZedBoard, that embeds a Zynq-7000 SoC. This is the same model of board and same part number used in our fault injection experiments, described in 4.1. On the PL part, two components have been instantiated: one of the SVM architectures, marked as SVM in the figure, and an indexed list of its respective input vectors, named the controller. The controller, when given an index, outputs to the SVM the input sample at that index. For example, when given as input the number 8, it will place on the instantiated SVM primary inputs the eighth input sample. The SVM module would contain either the Binary or the Multiclass SVM at a time, i.e. the Binary SVM and Multiclass SVM were tested separately. On the PS part, one module is responsible send to the controller the indexes while a second module reads the output of the SVM through an AXI interconnect and forwards it to a host PC through a serial port. The L2 cache of the ARM processor has been disabled to reduce the probability of faults affecting the PS [21]. No scrubbing mechanism nor the Xilinx Soft Error Mitigation (SEM) core IP were instantiated. We are aware that these IPs would be very useful in a final implementation, as they would not let errors accumulate. However, these tools have a time delay in order to detect/correct the fault. This may still be not sufficient in a short term for our design, as it is fully combinational. Thus, these IPs were left off to observe a worst-case scenario.

5.2. Radiation Test Method

The radiation test methodology is the same for both the Binary and Multiclass SVM and only one architecture was tested at a time. The set of input vectors is continuously evaluated in the SVM currently instantiated in the FPGA. The radiation is able to alter the Configuration SRAM (CRAM) of the FPGA, which contains the bitstream that implements the circuit, creating an error. Mathematically, as the error changes the SVM structure or its weights, it changes the classification function. As the classification function changes, the score of an input sample may deviate from the expected result. Thus, in order to identify an error, the primary outputs of the SVM are constantly compared with a golden error. At the first mismatch, we are sure of the presence of an error. Once we are certain of the presence of the error, the complete set of input samples is evaluated in the architecture, each input sample being the logged according to Subsection 4.2. Once the complete set of input samples has been logged, we rerun the complete set of input samples once more. This was done to identify if transient faults were present in the previous run. If a transient fault had happened during the computation of an input sample, it would have been corrected when the same sample was evaluated for the second time. After the second run, the FPGA is reset and the bitstream is reloaded in the FPGA to correct the errors. It is worth noting that the board is subject to SEUs as well as to Single Event Multiple Upsets (SEMUs), the latter

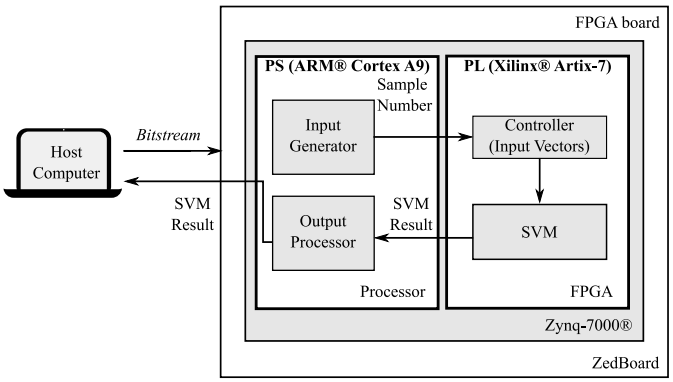


Figure 4: Zynq-7000 set-up under radiation test

being increasingly present in recent technologies given the shrinking size of transistors [22].

5.3. Radiation Test Results for the Binary SVM

During the neutron radiation test campaign, we have identified 24 errors, of which 3 crashed the FPGA and 21 errors that allowed it to continue to produce results. No transient faults have been identified. Even though crashes have been responsible for 12.5% of the total number of errors, they are not related to the case-study SVM architecture design but due to either a fault in the device performing the serial communication with the control computer, i.e. the PS, or on the on the AXI module instantiated on the FPGA fabric, which can cause the PS to hang if it fails to perform a proper handshake. The obtained static cross-section and the Failure In Time (FIT) are respectively $4.11 \cdot 10^{-10}$ cm^2 and 2.67, considering New York's flux at sea level (6.5 *thermal neutrons* / $(h \cdot cm^2)$) [12].

4695 samples were processed by the Binary SVM with radiation-induced errors. Note that the total number of samples is not a direct product between the number of input sample and the number of faults, as it would be expected by the methodology described in Subsection 5.2. This is the case as in some cases, the FPGA would halt after a fault before reevaluating the complete set of input vectors. Of the total number of samples evaluated, 7% resulted in critical failures, while 21.4% have been tolerable

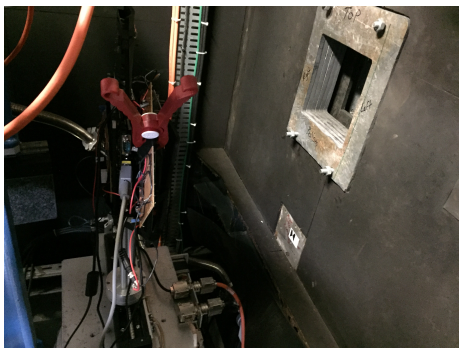


Figure 5: FPGA board installed at the D50 thermal neutron accelerator facility

failures and the majority, 71.6% were masked faults. From these results, it is noticeable that it is more likely for an error not to critically interfere with the application in this study case, as in 93% of the cases, the final classification of a sample would STILL be correct, recreating roughly the results in [11]. It is worth nothing that no fault mitigating or correction has been implemented on the Binary SVM, with the overall error resilience being an intrinsic characteristic of the algorithm.

5.4. Radiation Test Results for the Multiclass SVM

On the radiation campaign of the Multiclass SVM, a total of 16 faults were identified, of which 2 caused crashes. Again, the radiation induced crashes are out of the scope of the paper. The obtained cross-section is $2.042 \cdot 10^{-10} cm^2$ with a FIT of 1.32 using New York thermal neutron flux at sea level.

A total of 2884 samples has been evaluated in the implemented Multiclass SVM, of which 2049 have been classified as masked fault, 799 as tolerable failure, and 36 as critical failure. As observed on the Binary SVM, the Multiclass SVM also has an intrinsic tolerance to faults. The results indicate that only 1.2% of the evaluated samples on faulty Multiclass SVMs have been misclassified. Furthermore, only 3 out of 14 faults that had not crashed the FPGA have led to at least one critical failure.

5.5. Assessment of Results and Comparison of the SVM Architectures

First, we are going to discuss the effect of radiation-induced faults in Binary SVMs. On FPGA implementations of a Binary SVM, errors mathematically change the classification function (Equation 1) as they change either an x_i, an α or the calculation logic. The location of error may have a great importance on how it impacts the architecture. Errors on the least important bits of an x_i or α can potentially cause a small displacement of the classifier, making it less probable that a sample is misclassified. In terms of the architecture, the algorithm is a series of multiplications performed in parallel accumulated in a series of additions. This structure suggests that changes in the least significant bits are less likely to greatly impact the score of a sample, not causing a critical error. For example, an input sample that when evaluated on the SVM should output a score of 2.0, may have its score change to 1.9 if an error happened on the least significant bit, being still classified correctly. However, if an error happens on the most significant bits, the result could become -2.0, which would be a critical error as the signal of the score represents the final class. Note that different samples are affected differently in the event of a fault. Samples that have scores closer to zero are more likely to be misclassified, being sensible even to changes on not so important bits. For instance, a sample that has an original score of 0.2 is more easily made negative than a sample with score of 2. Thus, the distribution of the samples in regard to the

classifier has a great impact on the intrinsic reliability of the algorithm. On the other hand, the training algorithm of the SVM maximizes the distance between samples of different classes, i.e making the score of samples as high as possible, corroborating to an augmented reliability, but still highly dependent of the dataset. Finally, the output of the case-study SVM is composed of 49 bits, of which 48 are irrelevant for the calculation, only the sign bit being used for the final result. This indicates a high level of fault tolerance even to faults close to the output. All these properties translated into a 93% level of tolerance to faults for the case-study Binary SVM.

As mentioned in Subsection 2.2, Multiclass SVMs are made of a composition of Binary SVM, thus inheriting the intrinsic reliability properties previously discussed. However, our Multiclass SVM presented even higher levels of fault-tolerance in comparison to purely Binary SVM, only having a 1% rate of critical errors, suggesting that it is more reliable than the Binary counterpart. This behavior has also been present in the fault-injection campaign, as shown in Figure 3(b), in which the reliability of the Binary falls more rapidly than the one of the Multiclass. One possible explanation is that parts of the Multiclass SVM are irrelevant when evaluating a sample. For example, in our case, there were three possible classes for an unknown sample: *virginica*, *versicolor* and *setosa*. As the Multiclass SVM for this study follows the One-vs-One approach, described in Subsection 2.2, three Binary SVMs have been trained, one for each pair of classes. Assuming that an unknown sample should be classified as *virginica*, the result of the Binary SVM to classify between *versicolor* and *setosa* is irrelevant as long as the output of the other two remain correct. Therefore, Multiclass SVMs may build an extra level of reliability when compared Binary SVMs, which is indicated by the radiation results. It is worth pointing out that the datasets used have been different, which may have an impact in the levels of reliability. Further evaluation using both fault injections and radiation tests would be needed to better compare the difference in terms of reliability between the two architectures.

5.6. Comparison with State-of-the-Art Works

Few authors have explored the radiation effects on machine learning algorithms as it is still a new field. The intrinsic fault tolerance of an FPGA implementation of Artificial Neural Networks (ANN) is evaluated in [7], in which the authors perform a fault injection campaign along with a heavy ion campaign. The work is complemented in [8], where the same architecture along with an FPGA implementation of a Convolutional Neural Network (CNN), a very popular variant of ANN for image processing applications, have been evaluated under the effect of neutrons. Graphics Processing Unit (GPU) implementations of CNNs have also been evaluated under neutron radiation in [9].

In [7, 8], the authors have made used of the same dataset that we have used for our Multiclass SVM to train an ANN. Also, they have used the same FPGA platform. When comparing with their results, the authors have found that 65% of the faults led only to tolerable failures. In our case, we achieved 79% in respect to that, suggesting that SVMs may have a higher reliability in comparison to ANNs. We have also had reliability figures comparable to those of [9], even though the datasets used are different, but still with multiple output classes. Using GPU and CNNs, the authors have found that around 82% of the faults in one configuration bit of their GPUs would lead to no critical error. When comparing to GPU implementations of CNNs, while GPUs present better fault tolerance, FPGA implementations cannot be ruled out as they may be faster for some applications [23].

6. Conclusions

This work presents the first evaluation of SVMs under thermal neutron radiation along with the first assessment of radiation effects on Multiclass SVMs. Furthermore, both architectures were also thoroughly evaluated with an extensive fault injection campaign to correlate with the radiation results. On both the radiation and fault injection campaigns, the Multiclass SVM presented an overall higher reliability when compared to a Binary SVM. It is worth noting that neither designs had any error detection nor error correction mechanisms implemented, suggesting that the Multiclass SVM has a higher intrinsic reliability. Also, in our experiment, the Multiclass SVM performed better in terms of reliability than an ANN trained for the same dataset, suggesting that it may be more reliable.

As a future work, we intend to evaluate different datasets for both the Binary and Multiclass SVMs to further inspect our results. We also plan in conducting fault injection campaigns in a more fine-grained manner, e.g., injecting faults in each SV at a time or specific regions of the circuit, with multiple datasets in order to try to generate a model for the reliability that we could potentially extrapolate to other datasets.

Furthermore, we plan to test different architectures for the algorithm. For instance, the chain of adders at the end of the circuit could benefit from being pipelined. In terms of reliability, we could expect some differences. The circuit would be bigger, as several registers would have to be added, which could make the circuit somewhat more fragile. Apart from that, another type of error could become more present, which are errors that affect only one sample of the dataset and that are erased once another input sample is evaluated. They could happen as intermediate results would have to be stored in memory elements.

Acknowledgments

This work has been partially supported by: IRT Nanoelec (ANR-10-AIRT-05) and LabEx PERSYVAL-Lab

(ANR-11-LABX-0025-01), both funded by the French government program "Investissement d'avenir"; and Multi-Rad project funded by Région Auvergne-Rhône-Alpes's international ambition pack. In addition, we would like to thank R. A. Guazzelli (TIMA), Alexandre Coelho (UFC), and R. V. Della Giustina (ILL) for the help with the radiation test logistics.

References

[1] I. Garali, M. Adel, S. Bourennane, E. Guedj, Histogram-Based Features Selection and Volume of Interest Ranking for Brain PET Image Classification, IEEE Journal of Translational Engineering in Health and Medicine 6, Art. No. 2100212. doi:10.1109/JTEHM.2018.2796600.

[2] C. Weinrich, C. Vollmer, H. M. Gross, Estimation of human upper body orientation for mobile robotics using an SVM decision tree on monocular images, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 2147–2152. doi:10.1109/IROS.2012.6386122.

[3] I. Saha, U. Maulik, S. Bandyopadhyay, D. Plewczynski, SVMeFC: SVM Ensemble Fuzzy Clustering for Satellite Image Segmentation, IEEE Geoscience and Remote Sensing Letters 9 (1) (2012) 52–55. doi:10.1109/LGRS.2011.2160150.

[4] V. Vapnik, The Nature of Statistical Learning Theory, Springer Science & Business Media, 2013, google-Books-ID: EqgACAAAQBAJ.

[5] M. Ruiz-Llata, G. Guarnizo, M. Yébenes-Calvino, FPGA implementation of a support vector machine for classification and regression, in: International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–5. doi:10.1109/IJCNN.2010.5596820.

[6] P. Hazucha, T. Karnik, J. Maiz, S. Walstra, B. Bloechel, J. Tschanz, G. Dermer, S. Hareland, P. Armstrong, S. Borkar, Neutron soft error rate measurements in a 90-nm CMOS process and scaling trends in SRAM from 0.25-/spl mu/m to 90-nm generation, in: IEEE International Electron Devices Meeting, 2003, pp. 21.5.1–21.5.4. doi:10.1109/IEDM.2003.1269336.

[7] F. Libano, P. Rech, L. Tambara, J. Tonfat, F. Kastensmidt, On the Reliability of Linear Regression and Pattern Recognition Feedforward Artificial Neural Networks in FPGAs, IEEE Transactions on Nuclear Science 65 (1) (2018) 288–295. doi:10.1109/TNS.2017.2784367.

[8] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, P. Rech, Selective Hardening for Neural Networks in FPGAs, IEEE Transactions on Nuclear Science 66 (1) (2019) 216–222. doi:10.1109/TNS.2018.2884460.

[9] F. F. dos Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, P. Rech, Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs, IEEE Transactions on Reliability (2018) 1–15doi:10.1109/TR.2018.2878387.

[10] A. Coelho, R. Laurent, M. Solinas, J. Fraire, E. Mazer, N. E. Zergainoh, S. Karaoui, R. Velazco, On the Robustness of Stochastic Bayesian Machines, IEEE Transactions on Nuclear Science 64 (8) (2017) 2276–2283. doi:10.1109/TNS.2017.2678204.

[11] M. G. Trindade, A. Coelho, C. Valadares, R. A. C. Viera, S. Rey, B. Cheymol, M. Baylac, R. Velazco, R. P. Bastos, Assessment of a hardware-implemented machine learning technique under neutron irradiation, IEEE Transactions on Nuclear Science 66 (7) (2019) 1441–1448. doi:10.1109/TNS.2019.2920747.

[12] C. Weulersse, S. Houssany, N. Guibbaud, J. Segura-Ruiz, J. Beaucour, F. Miller, M. Mazurek, Contribution of thermal neutrons to soft error rate, IEEE Transactions on Nuclear Science 65 (8) (2018) 1851–1857. doi:10.1109/TNS.2018.2813367.

[13] J. C. Wang, L. X. Lian, Y. Y. Lin, J. H. Zhao, VLSI Design for SVM-Based Speaker Verification System, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 23 (7) (2015) 1355–1359. doi:10.1109/TVLSI.2014.2335112.

[14] F. Ye, F. Firouzi, Y. Yang, K. Chakrabarty, M. B. Tahoori, On-Chip Droop-Induced Circuit Delay Prediction Based on Support-Vector Machines, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35 (4) (2016) 665–678. doi:10.1109/TCAD.2015.2474392.

[15] M. Papadonikolakis, C. S. Bouganis, A Heterogeneous FPGA Architecture for Support Vector Machine Training, in: 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, 2010, pp. 211–214. doi:10.1109/FCCM.2010.39.

[16] C.-C. Chang, C.-J. Lin, LIBSVM: A Library for Support Vector Machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27. doi:10.1145/1961189.1961199.

[17] L. Acunha Guimarães, T. Ferreira de Paiva Leite, R. Possamai Bastos, L. Fesquet, Non-Intrusive Testing Technique for Detection of Trojans in Asynchronous Circuits, in: DATE, 2018.

[18] R. A. Fisher, UCI machine learning repository (1936). URL `http://archive.ics.uci.edu/ml`

[19] F. Benevenuti, F. L. Kastensmidt, Comparing exhaustive and random fault injection methods for configuration memory on SRAM-based FPGAs, in: 2019 IEEE Latin American Test Symposium (LATS), 2019, pp. 87–92. doi:10.1109/LATW.2019.8704647.

[20] R. Le, Soft error mitigation using prioritized essential bits, Application Note, XAPP538. (April 2012). URL `https://www.xilinx.com/support/documentation/application_notes/xapp538-soft-error-mitigation-essential-bits.pdf`

[21] L. A. Tambara, P. Rech, E. Chielle, J. Tonfat, F. L. Kastensmidt, Analyzing the Impact of Radiation-Induced Failures in Programmable SoCs, IEEE Transactions on Nuclear Science 63 (4) (2016) 2217–2224. doi:10.1109/TNS.2016.2522508.

[22] D. Rossi, M. Omana, F. Toma, C. Metra, Multiple transient faults in logic: an issue for next generation ICs?, in: 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2005, pp. 352–360. doi:10.1109/DFTVS.2005.47.

[23] E. Nurvitadhi, a. D. Sheffield, A. Mishra, S. Krishnan, D. Marr, Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC, in: 26th International Conference on Field Programmable Logic and Applications (FPL), 2016, pp. 1–4. doi:10.1109/FPL.2016.7577314.