Fisica

CORSO DI DOTTORATO DI RICERCA IN

XXXVI

CICLO DEL CORSO DI DOTTORATO

Characterisation of electronic devices and finite-dimensions deep neural network topology optimisation

Sacha Cormenier
Nome e cognome del dottorando: firma

Paolo Branchini
Supervisor firma

Giorgio Matt
Coordinatore firma

Ph.D. in Physics

36th Cycle

Ph.D. Thesis

Characterisation of electronic devices and finite-dimensions deep neural network topology optimisation

Ph.D. candidate: Sacha Cormenier

Supervisor: Prof. Paolo Branchini

Acknowledgements

Throughout this doctoral thesis, I have been supported and helped by numerous persons who truly deserve to be thanked. Words will never suffice to express my deep gratitude to them.

Firstly, I want to sincerely thank my supervisor Paolo Branchini, who gave me the opportunity to pursue this doctoral thesis, guided and advised me, and dedicated his time to teaching me how to grow as a scientist. His knowledge, experience, patience, commitment, and efforts have greatly contributed to my success in completing this thesis.

Another sincere thank you to Andrea Fabbri, that has made me grow not only as a scientist but also as a person, through his knowledge, experience precious advice given throughout the thesis.

My sincere gratitudes to Pierluigi Contucci for his willingness to teach me, his motivational words, the study opportunities he provided, and his recognition of me as a scientist and a student.

I am truly grateful to Pr. Federico Ricci-Tersenghi, for his patience, will to assist me, and invaluable advice for improving the reliability of results presented in this thesis.

A truly special thank you to Gianluca Manzan, for his unwavering support, help, advice, patience, determination to achieve success in our shared work. We started as colleagues midway through this thesis, and after overcoming numerous challenges together, we ended up as friends.

I am sincerely thankful to my family, for supporting me all along this thesis, believing and cheering in me whatever the challenges that occurred during these three years. Words are not enough to express my gratefulness to them.

A special thank you to Julie Raimbault, for her unwavering support throughout this journey, regardless of the challenges faced. I also want to express my gratitude to Guy, Florence and the entire family for accepting me as one of their own, and supporting me in any situation.

I am grateful to the three p'tits fafas Isaure Landèche, Maxime Parra and Mathilde Lemaire for all the discussions, good times, strange ideas and being my friends during this journey where we shared so much.

An obvious and sincere thank you to the Peaches Hugo, Raph, Thomas, Martine, Max, Lucas, Axel, Chloé, Vivi for sharing countless moments with me, accompanying me all throughout this journey, and offering continuous support along the way.

A huge amount of gratitude to Jules Dallant and Camille Graff for leading the way of the Ph.D., advising and supporting me, having unwavering faith in my work and being present in every situation.

I am also grateful to Broana Ramuz, for checking up on me, sharing both happy and sad moments, visiting, and offering support throughout all the journeys.

Thank you to Alix, Brorenzo, Clémence, Paul, Rayan, Estelle, Alice and Yanni for all the discussions, good times spent together, the laughs and complicity and simply for being my friends.

Thank you to Mayo, Alice, Alien, Paule and Alex for being around me for a long time now, you have been a unwavering support throughout the journeys and the years.

Thank you to Koltok, Aude, Tom, Célia, Dana, Doudou, Gaëtan, Théo, Ruben, Lucie and Juliette for always being around, you have been a great support and your cheers helped me to go through many obstacles of the various journeys.

A sincere thank you to all my comrades who shared laughs and obstacles during the journey of this Ph.D. Thank you to the old fishes Alessio Mattia, Chiara, Simone and Jamal; the new fishes Rita, Gabriele, Pietro and Federico; the Ph.D. friends made along the way Giulia, Massimo, Vittoria, Marco, Nina, Alessandro, Roberto, Ivano, Matteo, Daniele, Andrea, Paolo, Leonardo and Simone; the laboratory comrades Agnese, Valerio, Eleonora, Houssam and Tiziana; and Elisabetta and Dario who does not fit in any of these categories, for everything I shared with you. Thank you to Andrea and Piero for bringing joy every morning and afternoon.

I am grateful to Paolo Deiena, Walter Gemassmer and Marco Renzetti for giving me the opportunities to supervise, being patient with me, helping me learn how to teach, assisting me with my work and sharing a sense of camaraderie.

A special thank you to Gianluca Ghingo, who indirectly allowed me to share uncountable good times, creating unforgettable friendships and unlocking boundaries that changed my entire life. We started as student/teacher and teacher/student, and we ended up as close friends, I truly thank you.

I am sincerely grateful to Leonora, Andrea, Alessandra and Alessandro for accepting me as one of their own. I have truly gathered memories I will never forget, no words can

describe my gratefulness to you, from the bottom of my heart, thank all of you for everything. A particular thank you to Silvia Tosi, not only for being part of these marvellous people, but also for giving me the opportunity to meet them.

A special thank you to Marco Ruggieri for having my back since the very first day until the very last, you have been of great support and made my days more brightful.

A deep thank you to Alessio Rettaroli, for all the shared memories, moments spent together and the friendship we built along the way. Thank you for being my friend.

My deeper gratefulness to Romana Mikušincová, that has brought support, kindness, joy, safety, advice and so much help in everything. Thank you for having been here for me at any moment. Thank you for everything.

Last, I want to thank Raul Ciancarella. You have been along my side since the beginning until the very end, I don't know enough words to express how grateful I am to you. Thank you. For everything.

Abstract

This PhD thesis is driven by the aim of utilizing electronic devices in Earth orbit missions, incorporating embedded artificial intelligence on these devices. Given that electronics interact with particles from various sources in Earth orbits, it's crucial to study their behavior under radiation, as significant issues can arise without proper precautions. Artificial intelligence has proven to be one of the most powerful informatic tools, but if it is to be implemented in space embedded systems, the consequences of radiation on networks must be thoroughly investigated alongside the electronics. The disturbances induced in the network inputs, coupled with the constraints of resources due to power requirements and limited computational capabilities, need to be examined. Optimising the network architecture while considering the limited computational resources in these small networks is essential for enhancing performance.

Contents

List of Figures

List of Tables

Men are defined by their unrealistic dreams
Vir Das

1 Introduction

1.1 Overview of electronics and neural networks in physics

Electronics is a field of research that has been expanding rapidly since the last century. Since the experiments performed by physicists become smaller, more complex or delve wider and deeper into the unknown, electronics has become a mandatory tool to accomplish their purposes. With breakthroughs such as transistors, the computer industry, and Field Programmable Gate-Arrays (FPGA), physicists have been able to achieve discoveries and progress that were not attainable before. Electronics mainly assists in the capturing, analysing and interpretation of data.

More than a helpful tool for physicists, it has nowadays become the backbone of modern physics experiments. The precision and efficiency it provides cannot be overstated, especially in fields demanding the highest accuracy rates. In modern experiments of particle physics, like accelerators, advanced electronic systems can handle vast amounts of data generated while performing other tasks, such as classification and tracking. Physics experiments extend beyond terrestrial facilities to outer space for various objectives, such as probing the Universe without being constrained by the terrestrial atmosphere or studying the Earth for diverse aims. Even if the work differs from the ones done on Earth, electronics are still needed in satellites to help exploring distant exoplanets or to help studying cosmic radiations, for instance.

As we stand on the precipice of technological innovation, the integration of neural networks into electronic devices emerges as a transformative frontier in scientific research. Neural networks, inspired by the intricate architecture of the human brain, bring a new dimension to data processing and analysis. These artificial intelligence constructs have demonstrated remarkable capabilities in recognizing patterns, learning from complex datasets, and making predictions, attributes that align seamlessly with the demands of modern physics

experiments. Not only are neural networks used for complex tasks, but researchers have also been able to integrate them directly into the electronics used in the frontend part of the experiments. This step into the use of neural networks and the development of electronics in experimental physics has allowed for faster, stronger and more accurate jobs. However, this new area has also brought problems that must be taken into consideration during the development of the approach. Resource constraints, electronic noise, noise induced by particle interactions are issues that demand careful consideration.

1.2 Surroundings of the Earth

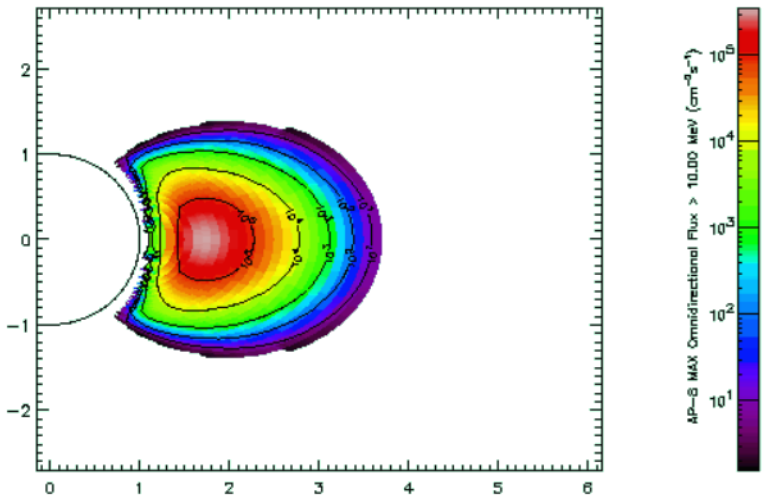
1.2.1 The Van Allen belts

The Earth's magnetic field extends into space and interacts with the solar wind, a stream of charged particles (mainly protons, electrons and small amounts of low energy heavy ions) flowing outward from the Sun. The interaction between the solar wind and the Earth's magnetic field leads to the formation of trapped charged particles regions like the Van Allen Radiation Belts.

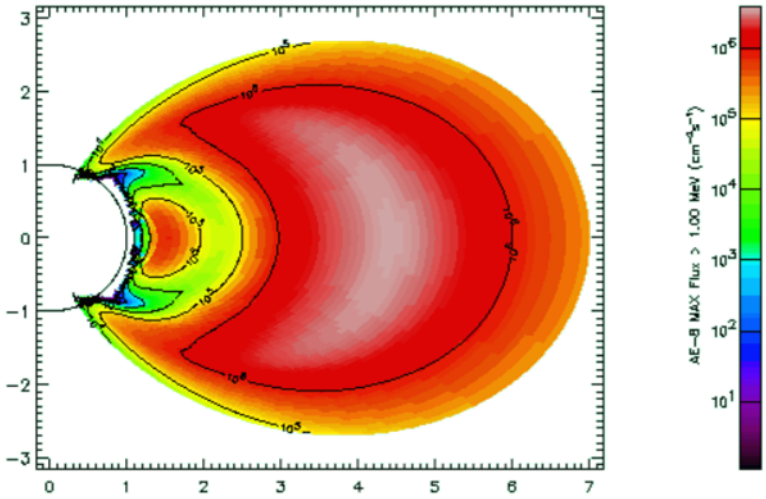
The Van Allen Radiation Belts, discovered by Geiger counters on Explorer 1 [1], are two zones inside the region of Earth's magnetic field, belt-shaped and composed of charged particles coming from the solar wind. The inner belt is mainly composed of high-energy protons (exceeding 100 MeV) and extends from 400 km to 6,000 km of altitude above the Earth. The outer belt is constituted of high-energy electrons (0.1-10 MeV) and is situated from 8,400 km to 36,000 km above the Earth surface [2]. A model of the Van Allen belts is displayed in Fig. 1.1, using the data of two space missions (AP8 and AE8) for the omnidirectional integral flux of protons (AP8) and of electrons (AE8). The particles trapped in the Van Allen Radiation Belts move in spiralling paths along the magnetic field lines and can have various effects and consequences. For example, when major disturbances in the Earth's magnetic field are caused by solar winds, the trajectory of the charged particles can be altered to make them move towards the Earth's poles and provoke ionization of atmospheric components emitting light of diverse colours. These physic phenomenons are also known as auroras.

1.2.2 Integral trapped particles spectra

The orbits between the Earth are divided into several geocentric orbits depending on their altitude regarding the Earth. For example, the Low Earth orbit (LEO) gathers the ranges



(a)



(b)

Figure 1.1: Models for the Van Allen Belts introduced in [3] where axes x and y are measured in Earth radii, being of 6,378 km. **a**, stands for the inner, belt and **b** stands for the outer belt.

between 160 km and 2,000 km, or the Geosynchronous orbit (GEO) stands for an altitude of 35,786 km, corresponding to an orbital period matching with a sidereal day. Since these orbits are placed in the Van Allen belts, it is necessary to study the particles spectra if one wants to characterise them.

The models AP8 and AE8 permitted the prediction for the integral particle spectra for a circular 500 km and 60°inclination orbit for LEO and is presented in Fig.1.2 for the protons and in Fig.1.3 for the electrons for both solar maximum and minimum activity [4]. The solar cycle is approximately 11-years long [5] where during four of them, the solar

activity is the least, being solar minimum, and seven of them stand for the most active conditions of the sun, being the solar maximum.

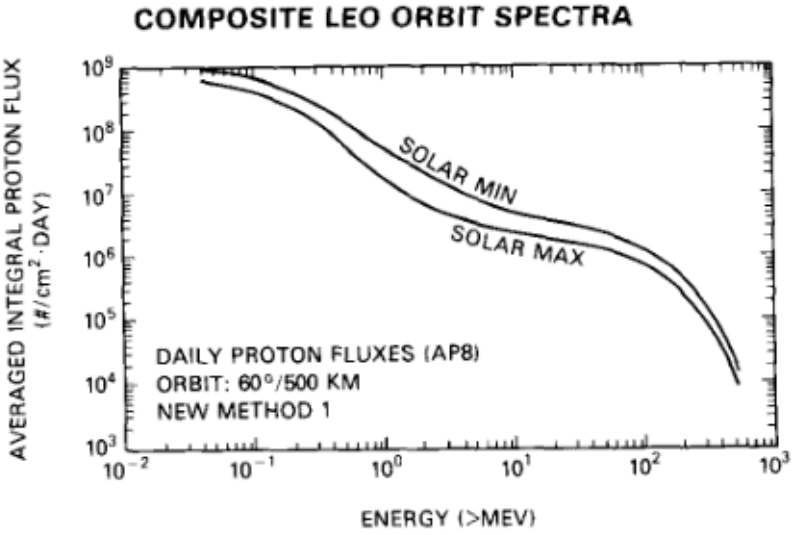


Figure 1.2: Low Earth orbit (LEO) proton fluxes from [4]

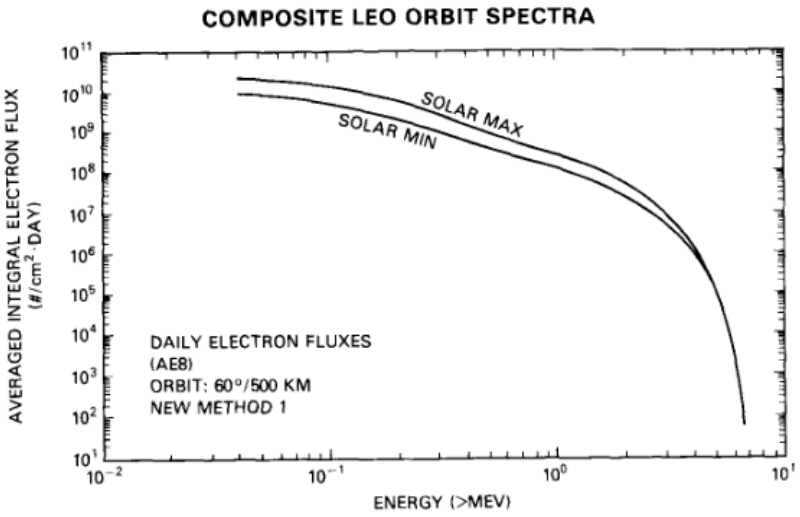


Figure 1.3: Low Earth orbit (LEO) electron fluxes from [4]

The AE8-MAX model permitted to display the integral electron spectrum for GEO and is plotted in Fig.1.4. The maximum and minimum values are not reached because of the solar activities in GEO but regarding the longitudes, corresponding to 160°W for the maximum and 70°W for the minimum. The proton integral flux in the GEO is not displayed because it has a significant proportion of lower-energy protons, while the number of high-energy protons is significantly lowered. The electronics are then not concerned by the trapped protons in the GEO because they can be stopped by thin material [4].

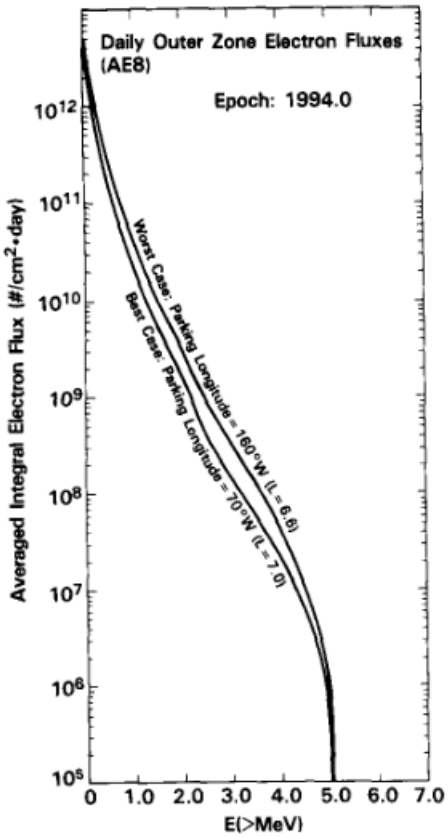


Figure 1.4: Geosynchronous orbit (GEO) electron fluxes from [4]

A major consequence of these trapped charged particle is the radiation hazard for spacecrafts. Indeed, they can be altered by damaging the electronics, with high-energy particles penetrating the shielding of the spacecraft and interacting with the components, which is critical for sensitive instruments and systems. Besides, the miniaturization of the electronics, making the device more powerful, has also made them more vulnerable to radiation, as the order of magnitude between the components and the charged particles are now comparable (in the case of protons and charged heavy-ions).

1.2.3 Galactic cosmic rays and solar particle events in LEO

Trapped charged particles are not the only source of primary ionizing radiation in LEO; galactic cosmic rays (GCR) bring charged particles from outer zones of the Universe into the solar system, composed of protons, helium ions and heavy ions [6]. Coronal mass ejections and strong solar flares can also produce fluxes of charged particles, called solar particle events (SPE), that interacts with the material sent in LEO [7].

In order to gather data about the flux of particles induced by GCR, we use Razaksat satellite orbit data since it has a nominal altitude of 685 km with 9°inclination, which match with our study of LEO. We observe in Fig.1.5 the comparison of particle flux of GCR intensities at solar maximum and minimum levels.

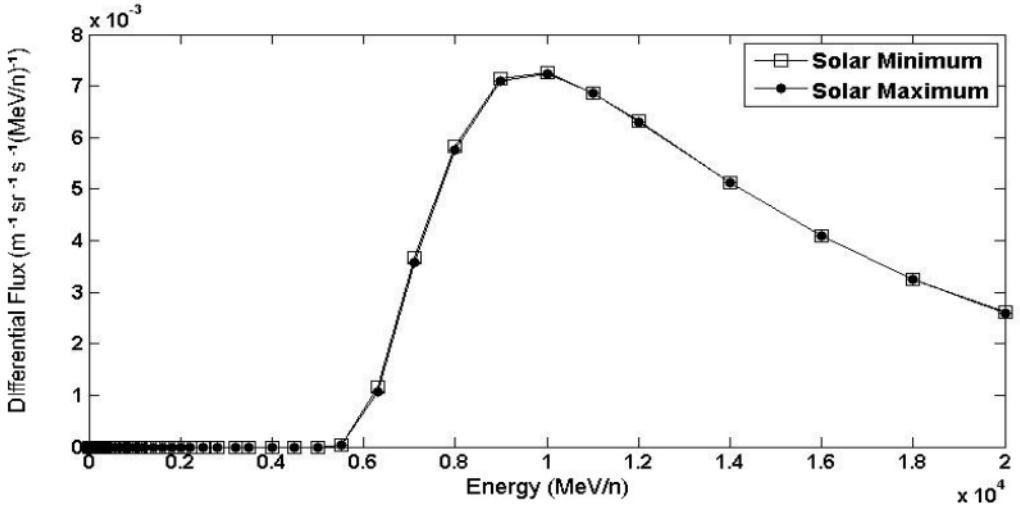


Figure 1.5: GCR fluxes during solar maximum and solar minimum from [8]

Since data has also been gathered for proton and electron fluxes at the same altitude, being 685 km, we are able to plot and compare the GCR flux with those of the trapped particles. In Fig.1.6 the GCR and proton fluxes are plotted while Fig.1.7 is dedicated for the GCR and electron fluxes. Both figures come from [8]. With these graphs, we observe

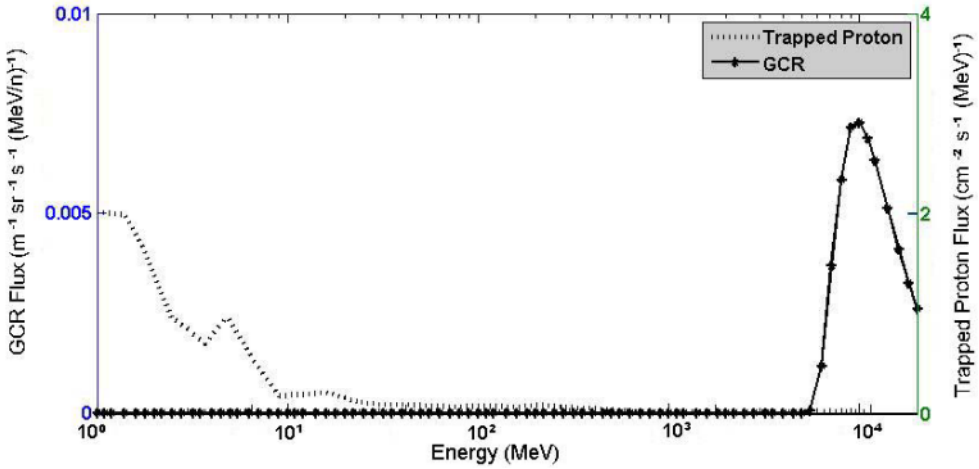


Figure 1.6: Fluxes for GCR and trapped protons at 685 km altitude from [8]

that GCR and trapped particles occupy different ranges of energies: in LEO, GCR are high energy particles above 10 GeV, while trapped particles are less energetic, below 400 MeV

for protons and 10 MeV for electrons. The main conclusion from these plots is that a wide range of particles can be found in LEO, and it is necessary to prevent damages from both of the cases.

Now that we have introduced the actors of our challenges, let's demonstrate how they affect the electronics and what the levels of hazards provoked by them are. To do so, we will exploit the amount of energy transferred by the particles, also called the Linear Energy Transfer, and the dosimetry.

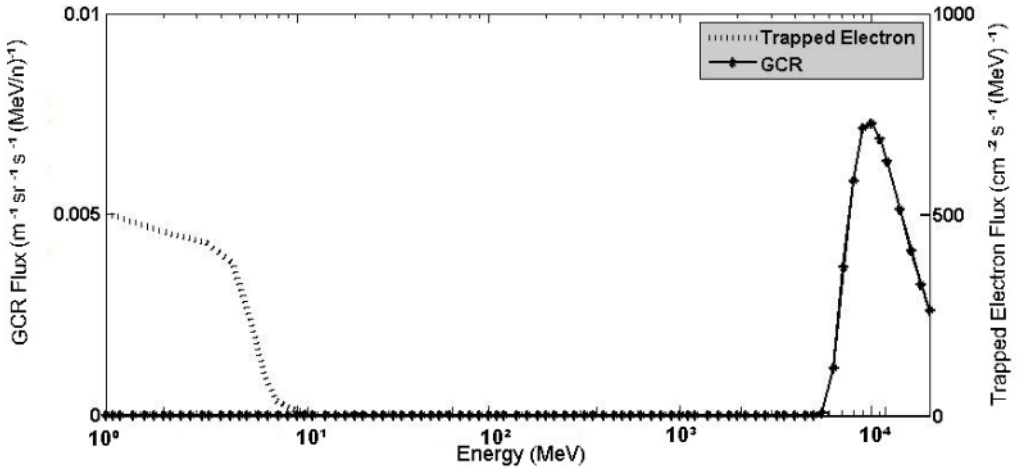


Figure 1.7: Fluxes for GCR and trapped electrons at 685 km altitude from [8]

1.3 Radiations measurements

The Linear Energy Transfer (LET) refers to the amount of energy transferred to a material per unit length by ionizing radiation. It is a measure of the density of energy deposition along the path of a charged particle, such as electrons, protons or charged heavy-ions, as it travels through a material. This feature is directly linked to the absorbed dose, representing the amount of energy deposited per unit mass in a material, which is directly correlated to the hazards threatening the electronics set inside a spacecraft.

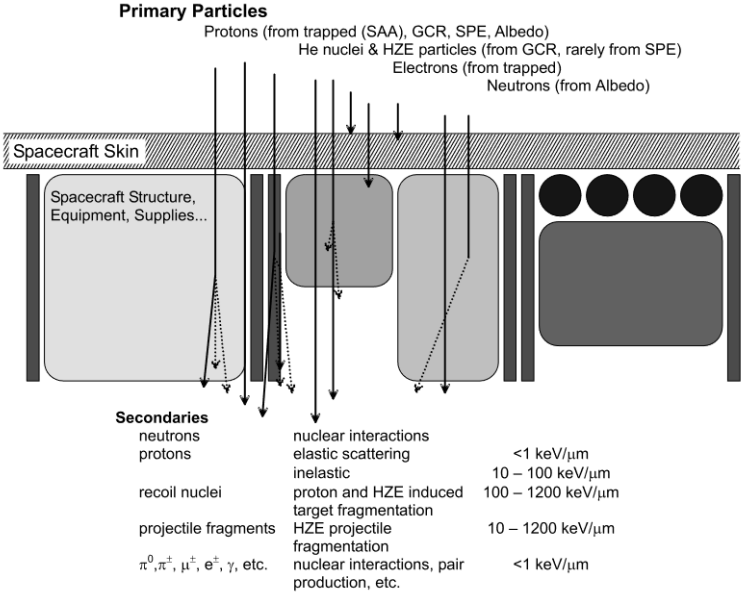


Figure 1.8: Scheme summarising the different primary particles interacting with the spacecraft skin, and the resulting secondary particles, from [7].

When a particle hits a material, such as the skin of a spacecraft in the case of space missions, a nuclear interaction occurs between the two, and the production of secondary particles occurs. Depending on several factors such as the kinetic energy of the incoming particle or its nature, the secondary particles will be of different composition with varying characteristics [7] and can interact with the electronics inside the spacecraft. In Fig.1.8, albedo trapped protons designates GCR protons reflected by the Earth's atmosphere, HZE stands for highly energetic heavy ions component, and SAA is short for the South Atlantic Anomaly, a region where an anomaly in the geomagnetic field makes the protons trapped in it have a high ratio of exposure for spacecraft passing in this zone.

Water, being a dense and hydrogen-rich material, can effectively slow down and absorb high-energy particles, providing a protective barrier for spacecrafts. LET $_{\infty}\mathrm{H_2O}$ of water is relevant because it describes how densely the energy is deposited along the path

of the ionizing particles, and it is the main feature used for these kind of measurements. In Fig.1.9, the integral LET flux spectra at four different altitudes in the SAA are shown: 338, 347, 379 and 391 km. This has been measured by the JSC-TEPC during the STS-63 mission [9]. We witness first a non-negligible LET correlated to the integral fluxes we have observed previously, and an altitude dependency in this region we also expect outside of this zone.

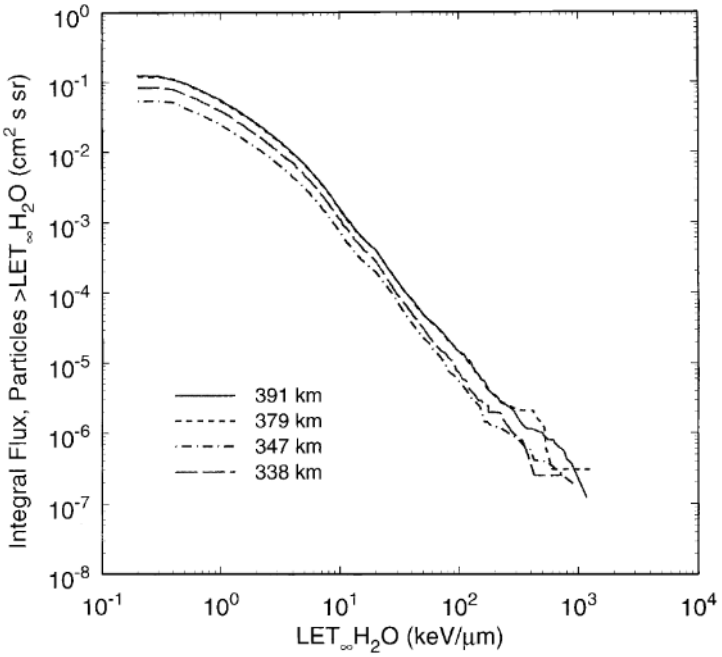


Figure 1.9: Integral LET flux spectra measured at four different altitudes. This data has been collected in the SAA by the JSC-TEPC during the STS-63 mission, from [7].

Regarding the dose rate, Space Shuttle missions have gathered data [10] in three different orbits since the atmospheric density is not homogeneous [10]: 28.5°, 38°-39°and above 57°inclinations; and it is plotted in Fig.1.10 as a function of the altitude. We see the dose not only increases with the altitude but also depends on the angle of inclination. The conditions a spacecraft will face depend on both its altitude and inclination angle, and it is mandatory to take it into account when computing the hazards impacting the electronics in spacecraft.

The previous analysis has been conducted mainly in LEO, as this region is of interest for many space missions. However, different studies have been conducted for various regions in outer space [11], such as the data observed in Fig.1.4, or around the moon as described in [12].

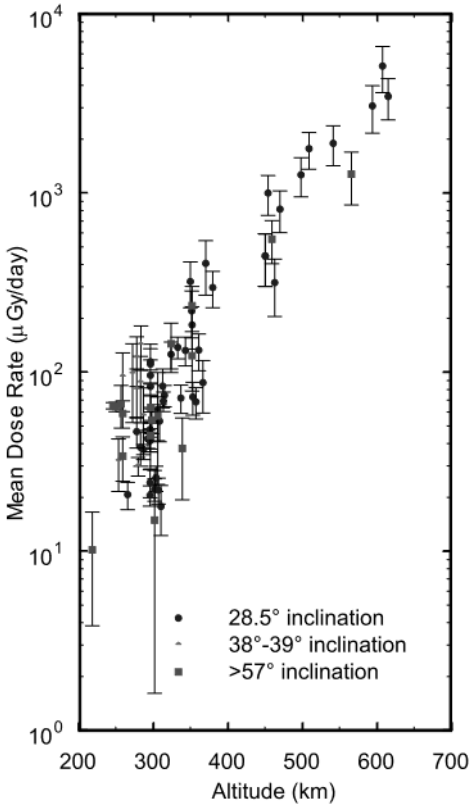


Figure 1.10: Mean dose rate as a function of the altitude measured by the Space Shuttle in three different orbits [10], from [7].

1.4 Electronics in spacecrafts

The use of electronics, particularly embedded systems like FPGAs and Commercial Off-the-Shelf components (COTS), in spacecraft represents a crucial aspect in modern space missions. FPGAs allow high flexibility and adaptability, enabling the implementation of custom digital circuits designed for specific mission requirements. COTS components, readily available and designed for commercial applications, have also found utility in space missions due to their cost-effectiveness and accessibility.

In the challenging environment of LEO, where spacecraft are exposed to intense space radiations, as exposed in Sec.1.3, the choice of electronic components becomes paramount. FPGAs, with their reconfigurable architecture, provide a way to optimize system performance while fixing the effects of radiation-induced modifications. Their ability to dynamically reconfigure circuits in response to changing conditions makes them well-suited for adapting to the dynamic radiation environment encountered in LEO.

COTS components, while not being particularly radiation-hardened, have been increasingly used in LEO missions for non-critical functions. However, the exposure to ionizing radiations in space induces potential risks, such as Single Event Upsets (SEUs) and latch-ups. As a result, spacecraft designers employ various techniques, including redundancy and error-checking mechanisms, to enhance the reliability of COTS components to face radiation challenges. Spacecraft electronics must undergo rigorous testing and qualification processes to ensure their sturdiness in the harsh space environment. In Sec.2.1 we present the results of the irradiation of a particular COTS by atmospheric-like neutrons and heavy-ions.

1.5 AI in electronics

In Sec.3.1, Artificial Intelligence (AI) and its functioning are precisely described. In essence, they are computer systems inspired by the way the human brain works. They consist of interconnected nodes, or "neurons," organized in layers. These networks can learn patterns from data, making them capable of tasks like recognizing images, understanding language, or making predictions. Essentially, AI enables machines to learn and make decisions by mimicking the brain's interconnected structure. They can be integrated into electronic devices, such as embedded platforms, bringing key aspects to be taken into consideration:

- **Resource constraints:** Embedded systems typically have limited processing power, memory, and energy resources. Implementing AI on such systems requires optimization and efficient algorithms to ensure that the AI models can run within these constraints.
- **Power constraints:** In some fields of applications, the total amount of power of a system is finite, and no excess is allowed by any part of this system. AI algorithms integrated into embedded platforms must not exceed allocated power consumption.
- **Optimized algorithms:** AI algorithms need to be optimized for embedded platforms. This may involve using lightweight models, quantization techniques, and model compression to reduce the computational and memory requirements while maintaining acceptable performance.
- **Continuous learning:** Some embedded AI systems support continuous learning or online learning, allowing the model to adapt and improve over time based on new data without requiring a complete retraining process.

1.6 Motivation for this Thesis

Since 2016, the session of the Italian national institute for nuclear physics Istituto Nazionale di Fisica Nucleare (INFN) of the University of Roma Tre has worked with Thales Alenia Space company regarding different studies. Requests such as the characterisation of electronic devices in particle ambient or the development of firmwares for electronic studies purposes has been performed.

A contract has been signed between the Italian Space Agency (ASI) and the company Thales Alenia Space for the design, development, and qualification of a vehicle for the In-Orbit Servicing (IOS) demonstrative mission.

In-Orbit Servicing is a demonstrative mission in LEO, scheduled for launch by 2026. The contract is part of the resources invested through the Italian Government's National Recovery and Resilience Plan (PNRR), through which ASI has been able to finance several significant national programs, such as the Space Factory, highlighted in the ASI 2019-2023 four-year report.

A growing number of satellites orbit the Earth to meet various needs, from geo-localisation to connectivity, weather forecasting to environmental monitoring, and more. To extend their operational life, these satellites would require periodic maintenance or assistance.

The demonstrative mission will test enabling technologies for future in-orbit servicing missions by performing various robotic operations on satellites already in orbit: refuelling, repairing or replacing components, orbital transfer, and atmospheric re-entry.

These operations will be carried out by a robotic arm, developed by Leonardo in collaboration with SAB Aerospace, INFN, and the Italian Institute of Technology (IIT).

In-Orbit Servicing activities represent a paradigm shift, introducing unprecedented system scalability and flexibility, providing opportunities for maintenance and updates in orbit and fundamentally changing the entire approach to satellite design. [13]

This kind of project motivates a study focused on the integration of neural networks in the newest generation of embedded platforms developed by Xilinx company: Versal ACAP. A study over the sturdiness to particle ambient, optimisation of the neural network and the resources must be undertaken.

2 Electronic devices characterisations

We have introduced in Chap.1 the need of studying the effect of particle ambient on electronics before it is used in space missions. In this chapter, we start by presenting the direct consequences of particles interacting with electronics in the case of COTS. This is done through the exploitation of beams of particles from two different natures and the effects of the radiations on electronics. This is presented in Sec.2.1 [14] and Sec.2.2 is dedicated to the presentation of a Versal ACAP board provided by Thales Alenia Space company for being studied in terms of resource consumption and potentiality for the insertion of AI. A second board is presented, a transitional one, for raw and hazardous studies.

2.1 COTS Irradiation

2.1.1 Introduction

In the present time, electronic devices represent a fundamental part of our lives, and they are the main elements for high energy physics experiments, medical applications and space missions, just to name a few. Some of these fields of study take place in environments subjected to various amounts of radiations that can compromise the electronics and the people involved; it is then of paramount importance to study the radiation environment and the effect that such radiations have on electronics, with a particular interest for COTS components, and humans.

COTS components are beginning to be widely used for space missions due to their reduced price and short procurement time. Therefore, a thorough radiation hardness study must be performed on them, since the space environment presents radiation hazards from particle fluxes that can produce effects like:

- Single Event Effect (SEE),
- Latch-up,
- Erosion,
- Total Ionizing Dose effect (TID),
- Displacement,
- Charging,
- Interference.

As reported in [15] there are already some devices, such as the microcontroller ATmegaS128 by ATMEL, that presents a radiation tolerance profile compliant with a LEO orbit. In the same paper, an extensive report on validating SEE caused by protons on COTS like the SPC56EL70L5 by STM, is also reported. The device tested with protons is the same studied in this section.

Other studies on the space environment and its effect on electronic components are reported in [16], where some solutions to mitigate the damages are presented. The most relevant of these are: the implementation of two equivalent systems that work in redundancy, in which a reset is performed if there is a disagreement between them; the application of a watchdog timer, where the system is reset if a pre-settled action is not registered in a specific time interval.

Our study will focus on the validation of a microcontroller by STMicroelectronics, namely SPC56EL70L5. Two tests will be presented: one performed with heavy-ions, the other with atmospheric-like neutrons. Protons, neutrons and heavy ions are present in space, particularly in LEO, where many satellites and cube-seat constellation will be stationed in the near future.

The paper is structured as follows: Sec.2.1.2 describes the tested device; the facilities at which the tests were performed are illustrated in Sec.2.1.3; Sec.2.1.4 describes the experimental methodology; the data analysis is reported in Sec.2.1.5 and finally Sec.2.1.6 summarizes the conclusions.

2.1.2 The SPC56EL70L5 Microcontroller

The Device Under Test (DUT) is the SPC56EL70L5, a 32- bit Power Architecture® microcontroller, manufactured by STMicroelectronics, and developed for automotive chassis and safety applications that require a high Safety Integrity Level (SIL). The device is designed with the possibility to configure it as a dual lock-step and, thanks to this configuration, it can achieve IEC61508 SIL3 and ISO26262 ASILD integrity levels.

A detailed block diagram of the device is shown in Fig.2.1.

The dual lock-step is provided with dual redundancy for the essential components of the device, represented by the cyan blocks in Fig.2.1; this redundancy allows the device to reach the targeted SIL with minimal additional software and hardware.

The critical components receive the same initialization values and execute the same operations; the outputs of these (displayed by the red blocks in Fig.2.1) are compared to check for errors. An error is reported if there is a discrepancy of the outputs.

The chip is a CMOS device contained in a LQFP144 package of dimension 20mm×20mm×1.4mm, and has 144 pins useful for alimentation, I/O peripherals, power management, ADCs (illustrated by the yellow blocks in Fig.2.1).

Each microcontroller is provided with the following memory:

- 2 MB of FLASH memory;
- 192 kB of SRAM memory.

The device operates at a frequency up to 120 MHz, and it is optimized for low power consumption while maintaining high performance processing power.

A full explanation of the chip architecture is available in [17].

2.1.3 Laboratories

Laboratori Nazionali del Sud (INFN-LNS)

The Laboratori Nazionali del Sud (LNS) of INFN in Catania, Italy, hosted the ion measurements, performed with the 0° beam line. The particles are accelerated in the vacuum with the Superconducting Cyclotron (CS) up to 80 MeV/nucleon. Just before exiting in the air, through a 50μm layer of Kapton, the beam is spread using a 15 μm foil of Tantalum.

To ensure that the energy deposition happens inside the device and to carefully evaluate the energy deposition inside the silicon, the Integrated Circuits (ICs) were decapped before the irradiation. Moreover, a 20 mm diameter beam was used to ensure that the whole chip is irradiated homogeneously.

For the purpose of our measurements, two different ions were used: ^{84}Kr accelerated to 1,678.24 MeV and ^{84}Kr accelerated to 780 MeV. The two ions have a LET of 45 MeV.cm^2.mg^{-1} and of 34 MeV.cm^2.mg^{-1} respectively.

ISIS Neutron and Muon Source

Neutron measurements were performed at ISIS Neutron and Muon Source at Rutherford Appleton Laboratories (RAL) in Oxfordshire, UK [18][19].

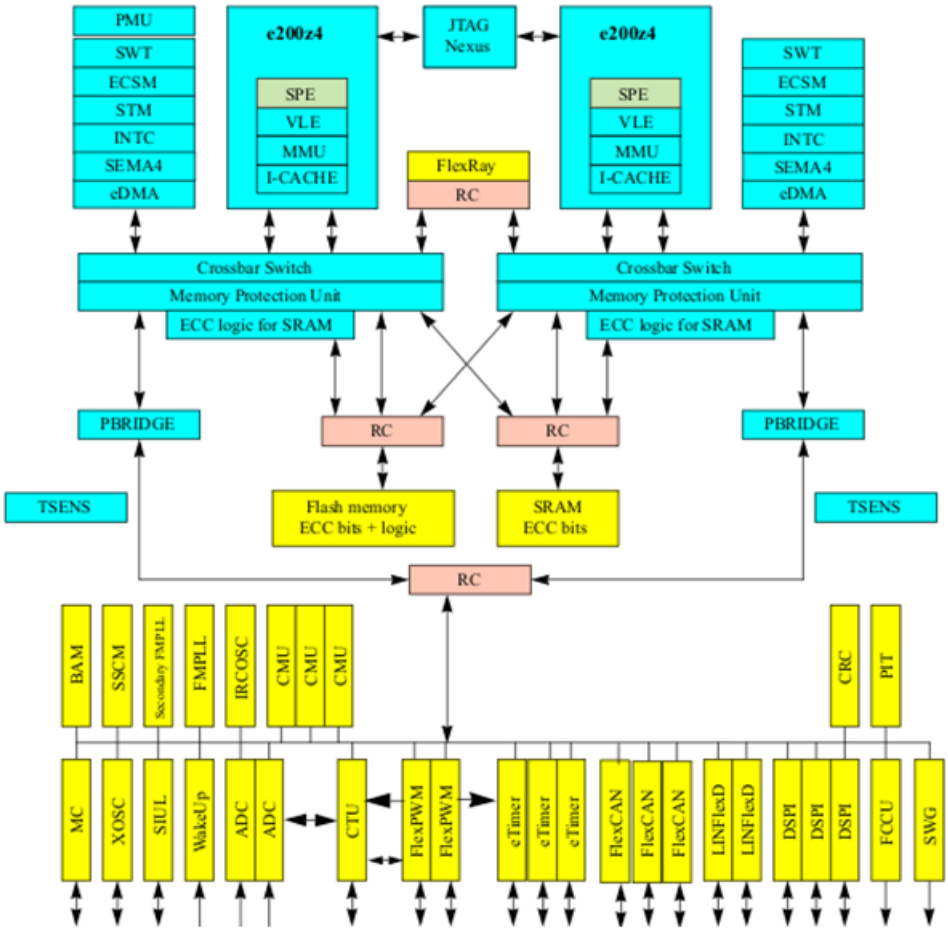


Figure 2.1: Block diagram of the SPC56EL70L5 microcontroller under test. Source: [17].

The ISIS spallation source uses a synchrotron (163 m of circumference) to accelerate protons up to 800 MeV; the proton beam is extracted and directed on a tungsten target to produce neutrons. There are two target stations used for different experiments. For our measurements we used target station 2 where there is a dedicated beam line for irradiating microelectronics with atmospheric-like neutrons, the ChipIr instrument [20], that can produce neutrons with energies $\mathrm{E_n} > 10$ MeV and with a flux of $5 \cdot 10^6$ neutrons.cm^{-2}.s^{-1} [21][22][23]. Through measurements, it has been shown the beam has a 70×70 mm^2 shaped uniform footprint. The energy spectrum of the neutron is shown in Fig.2.2.

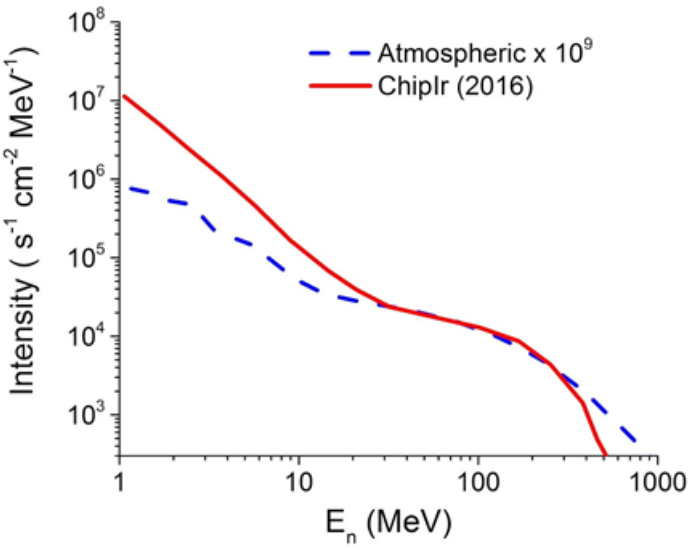


Figure 2.2: Neutron energy spectrum (continuous red line), evaluation of 2019, compared to the atmospheric one (blue line). Source: [24][25].

2.1.4 Methodology and Data Acquisition

To perform the tests, we defined the two types of board used:

- Control board: a remote board, not irradiated, on which the electronics for controlling and powering the microchips based on an FPGA device are mounted; this board is shown in Fig.2.3.
- Test board: a printed circuit on which the DUT is mounted; this board is equipped with the useful connector to bring the microchips pins to the control board and a BJT transistor for internal voltage regulation; this board is shown in Fig.2.4 (front) and in Fig.2.5 (back). After each irradiation, the test board was replaced with a new one.

The two boards communicate with each other via IDC connections. The control board has a USB 2.0 connection to communicate with a pc for controlling the acquisition software and for transferring the acquired data. The main panel of the acquisition software is shown in Fig.2.6.

On the control board there are 14 voltage supply channels, each of these can be monitored individually and, via shunt resistances or Hall-effect sensors, the corresponding currents are acquired. These currents are listed in Tab.2.1.

On the FPGA there is a control firmware that checks the sum of the measured currents; if this sum is greater than a threshold, the power supply is turned off to protect the

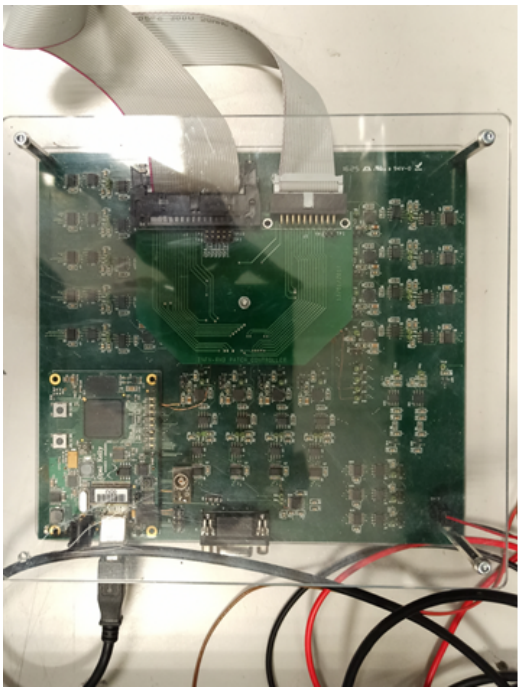


Figure 2.3: Control board. The IDC connectors on top bring the power supply to the test board. The USB connector at the bottom gives access to the acquisition software. This board is put outside the irradiation rooms.

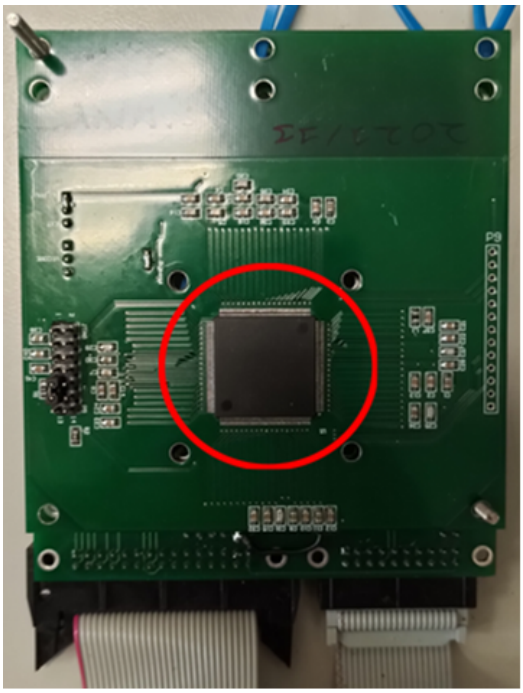


Figure 2.4: Test board, front side. The DUT is highlighted by the red circle and is powered up by the control board via IDC connectors.

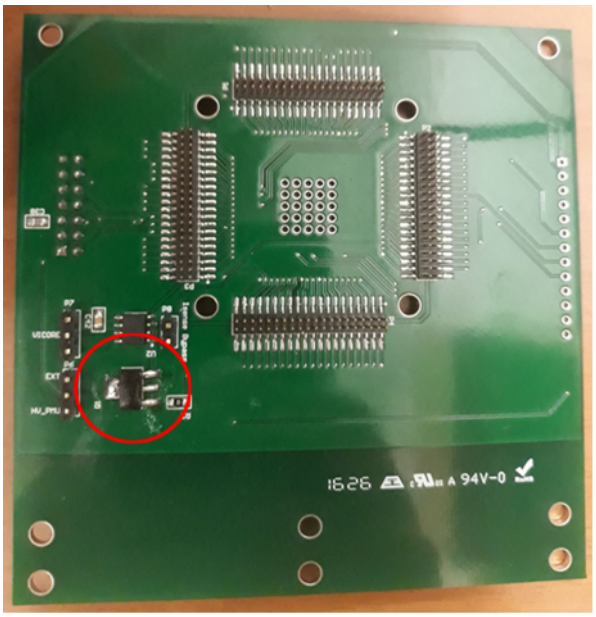


Figure 2.5: Test board, back side. The BJT transistor for internal voltage regulation is highlighted by the red circle.

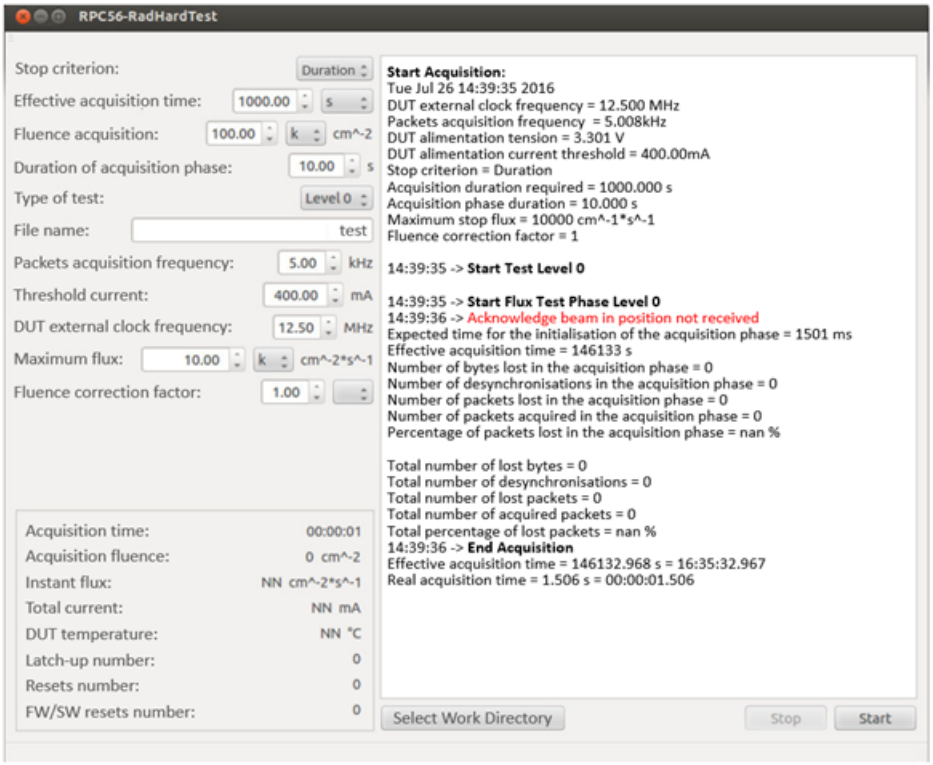


Figure 2.6: Main panel of the control software. It is possible to set the acquisition time, the current threshold for latch-up, the clock frequency and parameters for the flux. The software monitors and shows all the set parameters and gives feedback on the number of resets.

DUT, this defines a Single Event Latch-up (SEL). After a SEL occurs, the power supply is not available for 2 s.

Pin Number	Current	Current Description
6	I_IO	I/O pin
16	I_REG_0	PMU regulator
21	I_IO	I/O pin
27	I_OSC	Internal oscillators
50	I_ADDR0	ADC0 reference
56	I_ADDR1	ADC1 reference
58	I_ADV	Integrated ADC power supply
72	I_PMU	Power Management Unit (PMU)
91	I_IO	I/O pin
95	I_REG_1	PMU regulator
97	I_FLA	2 MB Flash memory
126	I_IO	I/O pin
130	I_REG_2	PMU regulator
EXTERNAL	I_Hall	Whole current monitor

Table 2.1: Monitored currents with pins

The firmware installed inside the DUT generates a square pulse on pin GPIO 79. This pin is monitored by the FPGA to check if the core is working correctly. When the chip is positioned under the irradiating beam, an additional signal is required to ascertain that the CPU is working: an acknowledgement signal in response to a control signal given by the FPGA. This acknowledgment signal is not necessary, but it is rather an additional check.

If the chip does not receive any waveform on the GPIO for more than 6.7 s, a hard reset is performed on the power supply because the execution of the firmware is considered off. This process can be executed as long as desired, usually a 40 s cycle is selected, referred to as a GPIO test.

The purpose of the experiment is to study the behaviour of the chip under accelerated irradiation. During this irradiation, the currents are monitored in two different phases: GPIO phase and Beam phase. The former is the phase when the DUT is irradiated, the latter is used to monitor the beam flux intensity, since the ion beam flux was not monitored by the LNS facility. In addition, the current absorption is measured while the samples are maintained at 80°C.

2.1.5 Data Analysis

Both experiments have been performed following the descriptions made in Sec.2.1.4.

Heavy-ions beam test results

A total of eight samples were irradiated with a heavy-ion beam: five with ^{84}Kr of energy 1,678.24 MeV and LET of 45 MeV.cm^2.mg^{-1}; and three with LET of 34 MeV.cm^2.mg^{-1}. The values for the LET were obtained with a simulation using the software SRIM2013 [26].

To avoid attenuation on the surface of the LQFP144 package, the device was decapped as shown in Fig.2.7. The decapping was performed by STMicroelectronics using a mixture 3:1 of nitric acid (HNO_3) and sulphuric acid (H_2SO_4), at room temperature, dosed with an automatic dispenser.

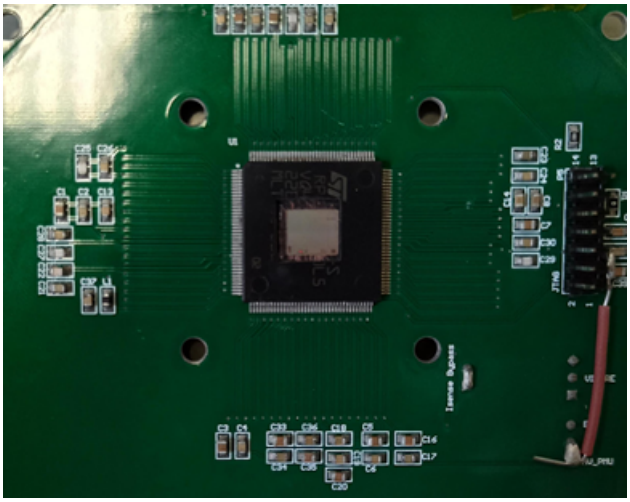


Figure 2.7: Close-up of the deccaped chip. Part of the LQFP package was removed with the acid solution.

For these tests, a mechanical device was implemented to control and measure the ion flux. The DUTs were mounted on a mechanical translator that moves the microcontrollers under or out of the beam. In line with the beam, behind the device, a scintillator crystal coupled to a photomultiplier was positioned. The irradiation of the DUT lasts for about 10 minutes, then the DUT is moved away to let the ions impinge on the scintillator that is irradiated for about 40 s. With this method, an ion counting was performed to verify and validate the flux characteristics. A schematic of the procedure is shown in Fig.2.8. In the figure, the movement of the translator is represented by the black double-headed arrow.

This mechanism represents the two phases of the test described in Sec.2.1.4: GPIO phase and Beam phase. At the time of the experiment, it was not possible to directly

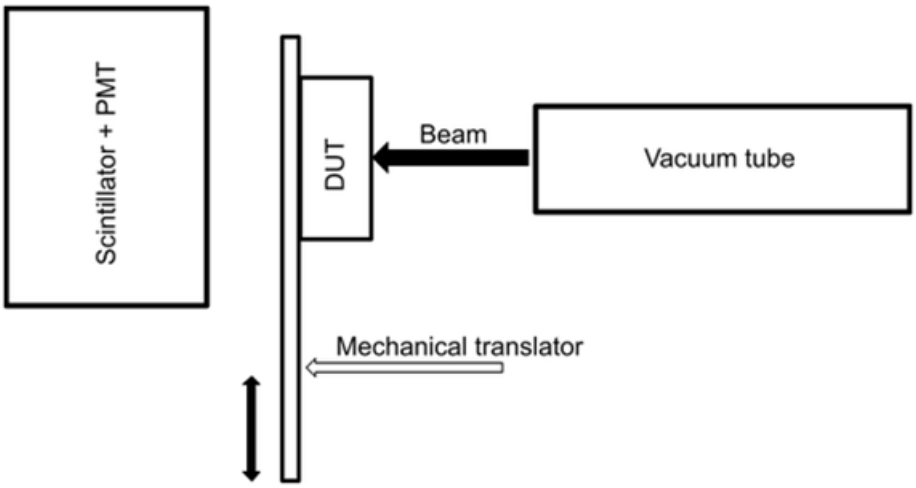


Figure 2.8: Schematic of the irradiation and ion counting. When the translator moves down, the beam hits the scintillator and the ion counting starts.

control the low fluxes of the ion beam, so it was crucial to implement a system that allowed the measurements and permitted to monitor the irradiation received by the DUT. For both irradiations, the latch-up threshold is set to 1 A.

In the two cases, the device is stabilized to (79.5±1.0)°C using a Peltier cell equipped with a control loop, composed of a Proportional Integral Derivative (PID) controller, to monitor the hysteresis. At this temperature the absorbed current by the DUT measured, before irradiation, has a mean value of (50.0±0.5) mA.

In Tab.2.2 are reported the results of the tests with the ion ^{84}Kr, and in Tab.2.3 the results with the ion ^{78}Kr. The absorbed current reported is the one after irradiation.

The flux has an uncertainty given by natural radioactivity produced by cement, equal to (20±5) ions.s^{-1}.cm^{-2}.

The effective fluence is obtained by multiplying the mean flux by the irradiation time. The uncertainty on this quantity is given by the integral over time of the natural radioactivity fluence which is equal to (20±5)·10^4 ions.cm^{-2}, for an interval of 10^4 s (ion ^{84}Kr), and (20±5).10^3 ions.cm^{-2}, for an interval of 10^3 s (ion ^{78}Kr).

A test is considered passed, i.e. the chips survived the irradiation, if the current absorbed and the contents of the EEPROM match before and after the irradiation.

Of the eight microcontrollers irradiated, only one, ST02, did not pass the test; the absorbed current for this was (30.0±0.5) mA.

For these samples, we computed the absorbed dose. Since this is proportional to the fluence and the energy loss rate in a material [27], we computed it by using 2.1:

Sample	Fluence (10^7ions.cm^{-2})	Irradiation time (s)	Absorbed current (mA)	Test passed
ST01	1.01	6027	50.0±0.5	Yes
ST02	1.21	6321	30.0±0.5	No
ST03	1.33	8205	50.0±0.5	Yes
ST05	1.27	10790	50.0±0.5	Yes
ST06	0.21	3592	50.0±0.5	Yes

Table 2.2: ^{84}Kr Irradiation results.

Sample	Fluence (10^7ions.cm^{-2})	Irradiation time (s)	Absorbed current (mA)	Test passed
ST04	1.14	13813	50.0±0.5	Yes
ST07	0.23	2706	50.0±0.5	Yes
ST08	0.026	900	50.0±0.5	Yes

Table 2.3: ^{78}Kr Irradiation results.

$$D = 1.602 \cdot 10^{-7}.\phi.LET, \tag{2.1}$$

where D is the dose (expressed in Gy=J.kg^{-1}), ϕ is the fluence (expressed in #particles.cm^{-2}) and *LET* is the Linear Energy Transfer (expressed in MeV.cm^2.mg^{-1}). The numerical factor is a conversion that changes MeV to J and mg to kg. The value obtained is from 15.14 Gy to 95.9 Gy with the ion ^{84}Kr, and from 1.4 Gy to 62 Gy with ^{78}Kr.

For each sample we also evaluated the number of SEL, the number of firmware (FW) blocks (corresponding to the number of times the DUT stuck) and the SEL-FW cross-section defined as 2.2:

$$\sigma = \frac{\#SEL + \#FWblock}{fluence}. \tag{2.2}$$

The results of this computation are reported in Tab.2.4. From this table, it is clear that the only events observed are due to FW blocks, and not to SEL occurred.

From Tab.2.2, 2.3 and 2.4 we can conclude that the DUT has a good resistance to heavy-ions fluences up to 10^7 ions.cm^{-2}, since only one of eight did not pass the test. However, a hot redundancy is needed to ensure the service of the data acquisition because we observed FW interruption during the irradiation.

The ion environment in a real orbit has much lower flux/fluence than our experiment [28]. To compare our result with the real environment, we have computed the FW block

Sample	Fluence (10^7ions.cm^{-2})	#SEL	#FW Block	σ (cm^2)
ST01	1.01	0	816	$8.08 \cdot 10^{-5}$
ST02	1.21	0	802	$6.63 \cdot 10^{-5}$
ST03	1.33	0	56	$4.21 \cdot 10^{-6}$
ST04	1.14	0	479	$4.20 \cdot 10^{-5}$
ST05	1.27	0	1248	$9.83 \cdot 10^{-5}$
ST06	0.21	0	130	$6.19 \cdot 10^{-5}$
ST07	0.23	0	120	$5.22 \cdot 10^{-5}$
ST08	0.026	0	15	$5.77 \cdot 10^{-5}$

Table 2.4: Number of SEL and FW block with cross-section.

rate for our experiment and the GEO and LEO environments [29]. The results of these computations are shown in Tab.2.5, where we have taken as example samples ST01 and ST04.

LET (MeV.cm^2.mg^{-1})	Environment	σ (cm^2)	Flux (cm^{-2}s^{-1})	FW Blocks rate (s^{-1})
	Experiment (ST01)		$1.68 \cdot 10^3$	$1.35 \cdot 10^{-1}$
45	LEO	$8.08 \cdot 10^{-5}$	$2.31 \cdot 10^{-8}$	$1.87 \cdot 10^{-12}$
	GEO		$6.37 \cdot 10^{-9}$	$5.14 \cdot 10^{-13}$
	Experiment (ST04)		$8.52 \cdot 10^2$	$3.47 \cdot 10^{-2}$
34	LEO	$4.20 \cdot 10^{-5}$	$8.10 \cdot 10^{-8}$	$3.40 \cdot 10^{-12}$
	GEO		$2.08 \cdot 10^{-8}$	$8.75 \cdot 10^{-13}$

Table 2.5: Flux and event rates for the experiment, LEO and GEO.

In the case of the experiment performed, there are more FW blocks than the ones expected in the real LEO and GEO environment; this is due to the higher fluxes used to accelerate the experiment. Considering ST01, as an example, this sample registers 816 FW blocks, which corresponds to a period between FW blocks of about 7.41 s which corresponds to a period of $5.35 \cdot 10^{11}$ s and $1.94 \cdot 10^{12}$ s for the LEO and GEO respectively when computing with the corresponding flux. For a three years mission, the FW block rates are $1.76 \cdot 10^{-4}$ for LEO and $4.85 \cdot 10^{-5}$ for GEO.

Neutron beam test results

The neutron beam experiment lasted 65 hours divided into four days, from the 27^{th} of April 2021 to the 30^{th} of April 2021. Seven samples were tested with neutron fluences on a single sample with a fluence up to 10^{11} neutrons.cm^{-2}.

In order to keep the same configurations, the DUT and the acquisition software are the same as those used in the heavy-ions tests but, in this case, there is no necessity to have the GPIO phase and the Beam phase since the fluence of the beam is directly measured by the hosting facility.

To begin with, the samples are tested in the experimental setup, with the neutron beam off, for a couple of minutes to collect some references. In a second time, the samples get irradiated for several hours. They get finally tested with the beam off again, to check the state and the behaviour of the chips during ten minutes after the irradiation.

The total fluence irradiated during this period can be seen in Fig.2.9. The grey areas represent the effective time under beam for each sample, labelled in the bottom left corner of the area. The points indicate the breaking time of the damaged chips.

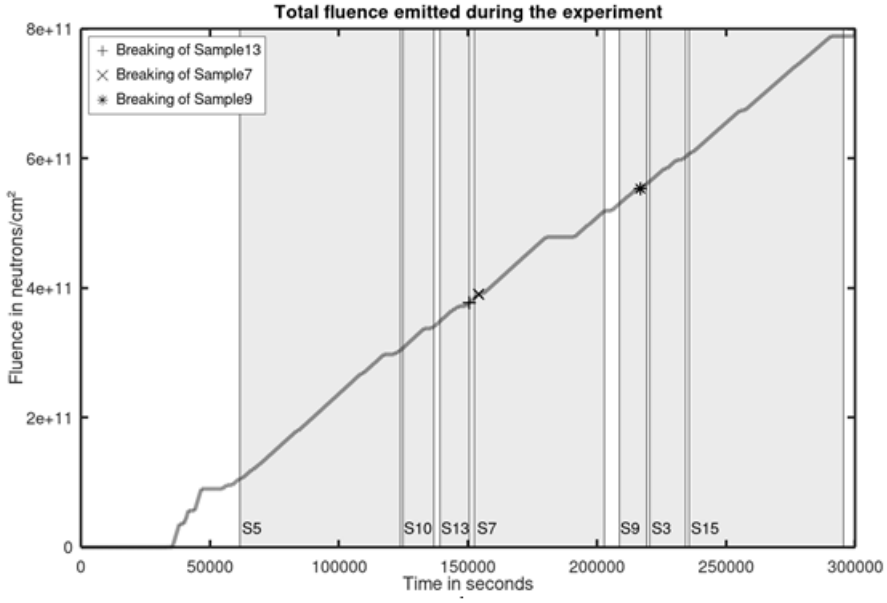


Figure 2.9: Total fluence registered by RAL. Each grey area represents the irradiation time of a chip, whereas the points the time of break for samples: 7 (×), 9 (*) and 13 (+).

The exposure time of each sample is reported in Tab.2.6 alongside the fluence the sample were exposed to, both total and before breaking, and a tag reporting if the chip has broken during the irradiation or not.

Sample	Total fluence (neutrons.cm^{-2})	Fluence before breaking (neutrons.cm^{-2})	Irradiation time (s)	Broken
S3	$3.78 \cdot 10^{10}$	–	13,885	No
S5	$1.98 \cdot 10^{11}$	–	62,220	No
S7	$1.34 \cdot 10^{11}$	$5.89 \cdot 10^{9}$	50,365	Yes
S9	$3.06 \cdot 10^{10}$	$2.31 \cdot 10^{10}$	10,543	Yes
S10	$3.27 \cdot 10^{10}$	–	11,938	No
S13	$2.86 \cdot 10^{10}$	$3.78 \cdot 10^{10}$ [a]	11,361	Yes
S15	$1.82 \cdot 10^{11}$	–	59,886	No

[a] During the whole post-irradiation test, S13 acted broken, whereas it did not during the irradiation test. We assumed this chip broke at the end of the acquisition, so the two fluences reported are the same.

Table 2.6: Neutron irradiation results

For our measurement, we defined a broken chip as one that resets continuously -every 7 s- during the GPIO phase of the test because, as stated in Sec.2.1.4, this defines the hard reset and in this situation the chip cannot recover the firmware.

We monitored the currents registered during the irradiation to check for this pattern of resets, an example of which is shown in Fig.2.10. The vertical axis is the current, registered on the ADC power supply, during irradiation time, expressed in mA; on the horizontal axis the time from the start of acquisition, expressed in seconds. Before the breaking point (represented by the dashed line) the device underwent resets, but these did not occur continuously and every 7 s. Instead, after the break, the current dropped by 1.5 mA and the microcontroller started resetting continuously. Every bunch of resets after break represents the GPIO phase of acquisition, and every bunch is 40 s away from the following, as selected via the acquisition software.

The number and frequency of resets measured in each sample are the main parameters to distinguish a chip has been broken during the irradiation from a chip which did not. Among the seven we tested, three of them (samples 7, 9 and 13) behaved like broken ones after a certain amount of time.

To check the status of all the chips after the irradiation, especially for the broken ones, other tests without the beam were performed two months after the irradiation time. The two-month time gap was necessary to allow the neutron-induced activity of the samples to decay and ensure that they were safe to be handled manually.

The non-beam radiation-less tests lasted two hours and were performed connecting the test board to the control board and monitoring the aforementioned 14 currents. For a functioning chip, we expected no resets, whereas a damaged one should have shown resets

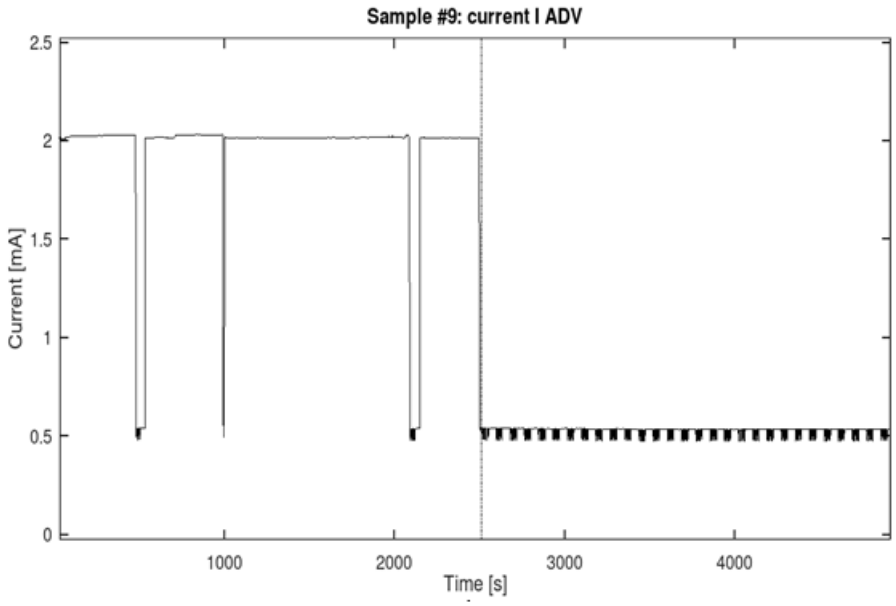


Figure 2.10: Example of the current acquired by a broken chip (sample 9). The vertical dotted line represents the breaking point, on the right side of this we can see the pattern of resets typical of a broken chip. The current shown is the one from the ADC voltage supply during irradiation time.

even in a "no-radiation" environment. Thus, we defined a damaged chip as a sample which provides resets during a test without radiation but does not show continuous resets.

After performing radiation-less tests on microcontrollers, S9 was still manifesting the broken behaviour (a total of 547 resets) during the two-hours test. However, S7 and S13 did not show this pattern despite the presence of resets (respectively 8 and 30), which should not occur for a functioning chip, not previously irradiated. Thus the radiations made these samples at least partially damaged.

To explain this change in the number of resets, we supposed that an effect of self-annealing occurred during the two months between the tests. No specific analysis were carried out, but, from literature [28], we found some evidence of self-annealing in semiconductors at room temperature that occurred over periods of 45, 60 and 100 days. Despite better results regarding the number of resets, the annealing effect did not make the chips to fully recover. Indeed, besides having currents behaving like working samples, the chips did not act like broken ones, displaying a broken pattern. However, unexpected resets were observed, which proves the chips were still partially damaged.

We can establish that the neutron beam irradiation didn't result in SEL behaviour of the DUT, taking into account that the used beam has an intensity several orders higher than the natural LEO environment.

The reset pattern of the broken chips can derive either from the DUT in the front of the test board, Fig.2.3, or from the BJT transistor on the back, Fig.2.4. As a consequence,

it is possible that either only one of these components got broken, or both of them did. To determine which of the two devices is the reason for this behaviour, a solution is to change the transistor with a known-fine one. This modification was made on S7, S9, and S13.

Once again, a two-hours radiation-less test was made for these three samples, with the transistors changed, and respectively 32, 16, and 12 resets were counted. The number of resets for S9 decreased significantly, and the behaviour previously described was no longer observed. The number of resets for S7 and S13 was still in the same range as previously mentioned.

In Tab.2.7, the number of resets for each phase of the analysis are summarized: irradiation, radiation-less and radiation-less after changing the transistor (this last one only for samples 7, 9 and 13).

Sample	Resets during irradiation)	Resets during radiationless tests	Resets after 60 days	Resets after changing the transistor
S3	20	2	0	–
S3	29	20	0	–
S3	3,782	30	8	32
S3	194	18	547	16
S3	147	4	0	–
S3	181	62	30	12
S3	24	0	0	–

Table 2.7: Sample resets summary

These values mean the microcontroller was broken for each of these three samples. Besides, during the irradiation tests, the transistors and the microcontroller broke, but the two-month waiting made an annealing process occur for the transistors of S7 and S13.

The results from Tab.2.6 and Tab.2.7 allow us to say that the DUT has a moderate resistance to neutron fluences up to 10^{11} neutrons.cm^{-2}, since three out of seven appear to be damaged. Changing the BJT helped to lower the number of resets for S9 and S13, whereas S7 increased the number of resets. In the case of S7, a functioning BJT could correctly polarize the DUT and transmit the resets as it should. Thus, the right number of resets occurred and are observed.

2.1.6 Conclusions

The purpose of this work was to study the resistance and the behaviour of the SPC56EL70L5 microcontroller under irradiation of two different particle beams. In order to do it, seven chips were exposed to atmospheric-like neutrons, and eight to heavy-ions.

Such study was performed looking at the number of resets due to the chip, in the case of neutrons. For heavy-ions, a comparison between the currents before and after irradiation was done. We kept the same configuration for both experiments on purpose to keep a matching in the bugs, the functioning and behaviours.

We observed that:

- The microcontroller shows sufficient resistance to neutron beam fluxes up to 10^{11}neutrons.cm^{-2}. Out of the seven samples tested, three (S7, S9, S13) showed a broken pattern immediately after irradiation but, after two months, only one of these (S9) maintained the broken pattern; S7 and S13 showed regular currents and a reduced number of resets. A possible explanation is attributed to an annealing effect that took place during the two months break between the irradiation and the radiation-less tests; but no further analysis was conducted. Even if self-annealing effect could occur in spacecraft, the objective of this study is to present the results without counting on annealing. Since the origin of resets could be the chip as much as the transistor of the board, we changed the latter. After this change, the current did not show the broken pattern any more, but there were still resets, implying the chips were damaged.
- The DUTs show a good resistance to heavy-ion beam fluxes up to 10^{7}ions.cm^{-2}. Only one (ST02) out of eight samples did not pass the test, presenting an absorbed current after irradiation lowered by 40%. All the observed events are due to firmware block, no SEL occurred.

The main issue we can identify in these tests is that the tension regulator has been placed inside the DUT, making it sensitive to interactions, and likely causing disruptions in the patterns. It would be beneficial to utilise an external regulator with a voltage of 1.8 V.

2.2 Embedded systems

In the same manner as COTS have been studied, understanding the response of embedded systems to ambient particles is crucial before deploying them in space missions. In this section, we introduce the two systems that will be utilised for future studies concerning embedded systems.

2.2.1 VCK190es1

Thales Alenia Space has provided INFN Roma Tre with a Versal AI Core Series VCK190es1 (shortened to VCK190), depicted in Figure 2.11, for the purpose of examining power consumption and implementing AI utilising continuous learning for classification tasks for space missions.

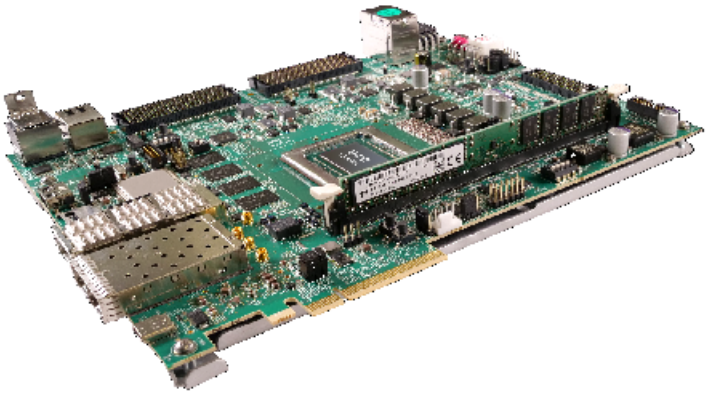


Figure 2.11: VCK190 board, from [30]

The AMD Versal™ XCVC1902 device is integrated into the VCK190 evaluation board, enabling the developments of applications such as machine learning. It offers support for various development tools, including optical transceiver supports, HDMI and USB plugs, connectors and Peripheral Component Interconnect Express (PCIe) [31]. To assist data scientists and developers in programming and optimizing their work, tools such as the Processing System (PS) have been developed, providing numerous components for various development activities [32]. Additionally, a Targeted Reference Design has been created for embedded video processing applications, utilising the PS and AI Engines (AIE) [33].

Among the array of tools accompanying the Versal VCK190, AIE and Digital Signal Processing (DSP) offer computing performance that exceeds current server-class CPUs by a factor of a hundred [30]. While this embedded system is ideal for remote machine learning and signal processing, its suitability for use in space missions, the impact on electronic

components, and the implications for machine learning characteristics must be thoroughly investigated.

Since this electronic board has been provided in only one exemplary, and it comes at considerable cost, the risk of damage during irradiation tests cannot be taken. Therefore, a secondary board, the Zybo-Z7 10, has been selected for initial testing purposes.

2.2.2 Zybo Z7-10

The criteria for selecting the secondary board were as follows: it should have a FPGA with ample resources capable of implementing artificial intelligence, programmability for tailored tasks, features enabling the addition of a camera, processing of video inputs, and an affordable price.

The Zybo Z7 board with a Zynq-7010 System-on-Chip loaded (abbreviated as Zybo Z7-10), displayed in Fig.2.12, is an embedded system that features programmable systems with FPGA logic. It offers various video interfaces such as MIPI CSI-2 and HDMI inputs and outputs, and is priced at less than €400 [34][35]. Featuring a 667 MHz dual-core Cortex-A9 processor tightly integrated with Xilinx FPGA and 270 KB block RAM, it emerges as an ideal candidate for implementing AI through the utilisation of the open-source project PYNQ from AMD company.

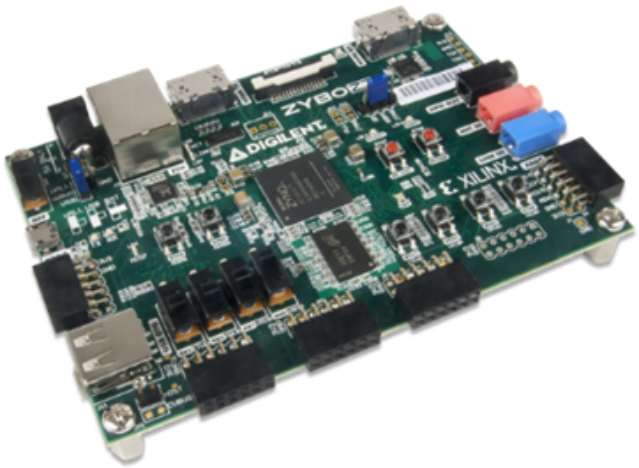


Figure 2.12: The Zybo Z7-10 board, from [34]

The PYNQ project enables programmers and engineers to leverage Adaptive Computing platforms, facilitating updates after deployment and thus optimized for custom applications. PYNQ supplies a Jupyter-based framework including Python APIs for utilising AMD Xilinx Adaptive Computing platforms and is compatible with the Zybo Z7-10 board, allowing Python to be used for programming embedded processors [36]. Therefore, Python

frameworks can directly assign AI components to the electronic device, and machine learning can be correctly implanted using the Brevitas project, explained in Sec.3.2.

3 Neural Networks

3.1 Neural Networks

First created to model the brain neurons [37] and the way they work [38], Artificial Neural Networks (ANN), or shortened neural networks, have evolved to reproduce [39] [40] and surpass [41] [42] tasks that should only be achievable by humans. Mimicking the way neurons transmit information to one another, they can learn and comprehend patterns and connections in data through experience. Neural networks are composed of several units, or neurons, often arranged in layers and all connected to one another [43]. The behaviour of a network is defined by various options and metrics, and the possibilities of use are unlimited [44]. We must introduce how a neural network is constructed for the comprehension of the rest of the study.

3.1.1 Description of a neuron

The first step of understanding the functioning of a neural network is to comprehend what neurons are. They are described as follows [43]:

A neuron receives several inputs x_i, depicted as $x_1, x_2, ..., x_{n-1}, x_n$ in the example shown in Fig.3.1. All of these inputs are summed together while being assigned a value w_i called the *weight* and a constant, the *bias* b. The operation performed by a neuron is

$$y = \sum_{i}^{n} x_i w_i + b. \tag{3.1}$$

The result of this operation y is then processed by a non-linear function called the *activation function* [43], represented as f in Fig.3.1. The non-linearity of this function is preferred for network learning, as it enhances the adaptability of the network to diverse

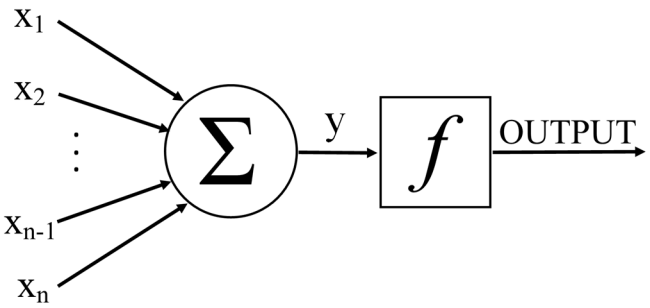


Figure 3.1: Scheme describing how a neuron works. All the inputs $x_1, x_2, ..., x_{n-1}, x_n$ provided are summed by the neuron, and then processed by a function f that must be chosen. The output is then transferred to the next step as an input.

datasets. Activation functions can take several forms with Rectified Linear Unit (ReLU), Sigmoid and tanh [43] being among the most well-known, described as

$$\begin{aligned} \sigma(z) &= \frac{1}{1+e^{-z}}, \\ tanh(z) &= \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} = \frac{1-e^{-2z}}{1+e^{-2z}}, \\ Relu(z) &= max(0,z), \end{aligned} \tag{3.2}$$

and their representation is plotted in Fig.3.2.

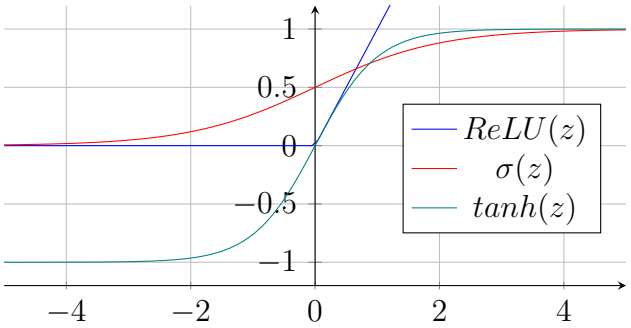


Figure 3.2: The activation functions

The result of such a function provides the output of the neuron, which can serve as the input for an adjacent neuron and/or be studied by an external source. The choice among the different activation functions is made regarding several parameters involved in the computation [45].

The processing of inputs by a neuron aims to assign different importance, with the weights, to the many provided inputs. Subsequently, a threshold, the bias, is set to determine how much of the value the neuron must "fire" as an output. In the case of the Relu function, for example, the neuron sends 0 if the value is smaller than the bias.

3.1.2 Networks

Now we have seen that a neuron can process information by assigning weights relative to a threshold, an interesting approach is to assign different weights and biases and verify the output of these operations. However, to retain the data already obtained, the results could be compared or handled by an external source. Consequently,we need to create several neurons receiving the same inputs, as illustrated in an example in Fig.3.3.

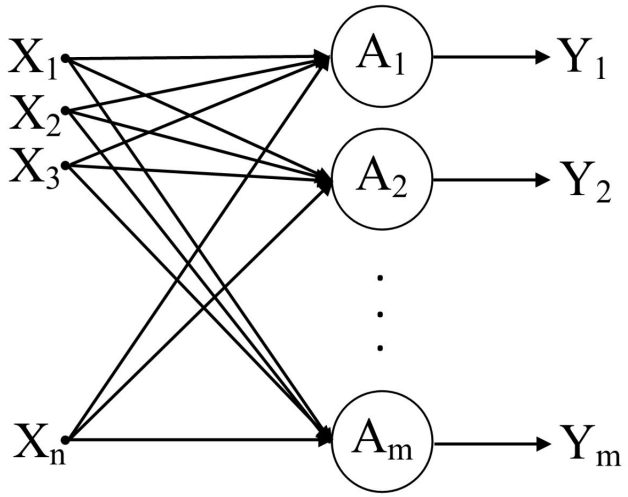


Figure 3.3: Scheme describing how several neurons take the same inputs and process them in different outputs.

As mentioned earlier, each neuron's output can be analysed by an external source. This means that if we assign weights to them, they can be processed as inputs by other neurons. As a result, another layer of neurons can be established, succeeding the existing one, and this process can be repeated. This part of a network is referred to as the hidden layers [43].

In Figs. 3.1 and 3.3, the inputs denoted as $x_1, x_2, ..., x_n$, represent values that can be obtained from various results, such as coordinates, values of a dataset, flattened images or texts for example [46]. It's conventional to represent them in the same manner as neurons and gather them in a layer called the input layer.

Besides, in order to make this ordered aggregate of neurons useful, we require it to yield results from all the calculations. Therefore, the last layer of neurons must provide output values such as positions, binary results, or probabilities, for example. This final layer is termed the output layer [43].

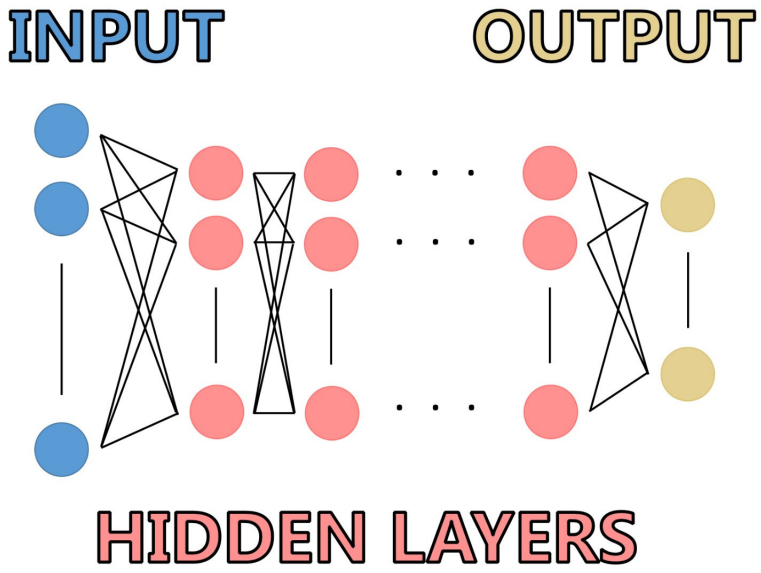


Figure 3.4: Scheme of a neural network.

When all these layers are gathered, we get an object called a neural network, which can be seen in Fig.3.4.

3.1.3 Mechanisms of classification

As said previously, many different tasks can be asked to neural networks, one of them being image classification. To manage such a study, one must give an image among a set of several images ordered into different classes as input. In Fig.3.5 are displayed examples of images coming from the weather set introduced in [47] where images can come from 11 different classes such as hail, rain or snow.

The network is aware of the number and names of the classes. When an image is provided, the layers of colours are separated in case of RGB images, and the pixels are flattened to shape the input layer. The output layer has the same dimension as the number of classes; each neuron within it is associated to a specific class [43]. After processing the input through the hidden layers, the index of the class corresponding to the neuron with the highest value stored in the output layer is chosen as the prediction by the network.

(a) **(b)** **(c)**

Figure 3.5: Three images coming from the weatherdataset [47]. **a**, image from the sandstorm class. **b**, image from the lightning class. **c**, image from the fogsmog class.

If the weights have not been pre-trained to enhance network accuracy, either they are initialised randomly, or they are initialised following a specific method [48], with a value assigned to each connection (represented as a line between two neurons in Fig.3.4).

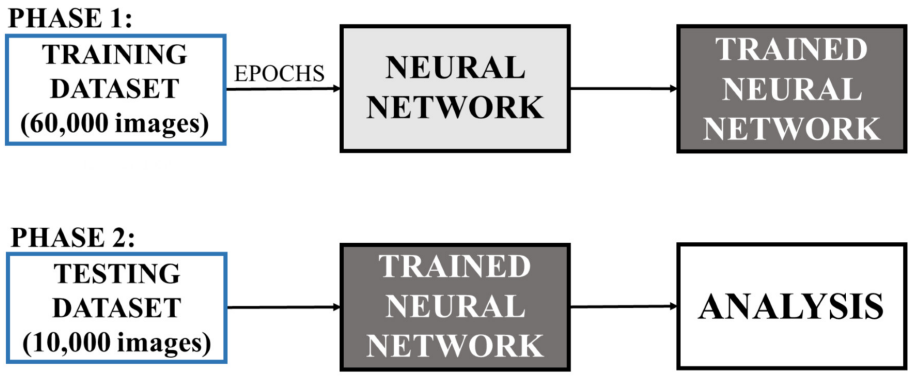


Figure 3.6: Scheme describing how training and testing sessions work on a neural network.

The network cannot make accurate predictions with all of its parameters set pseudo-randomly. This is where the network training comes into effect. Datasets and network usage sessions are separated into two distinct phases in supervised learning, the training and the testing:

- The training session begins with input data passed along the network through the different mathematical operations. The predicted output is obtained in the output layer; this is known as forward propagation [43].

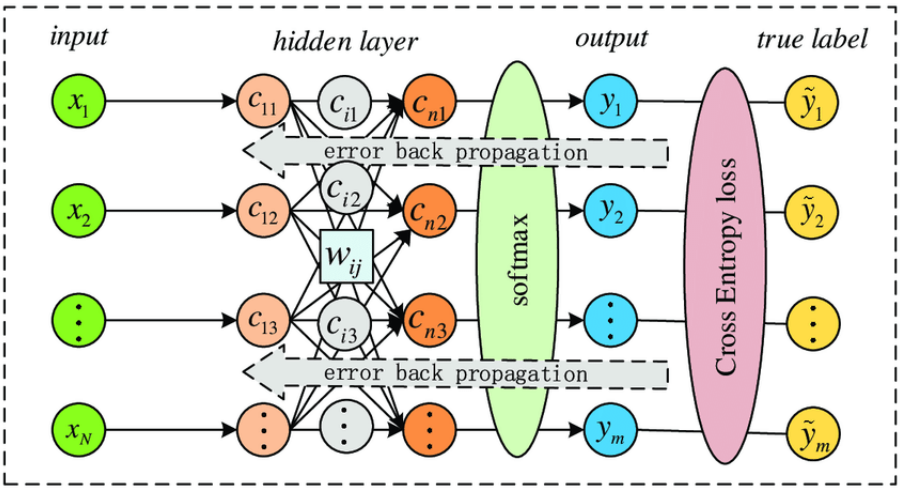


Figure 3.7: The structure of neural network in which softmax is used as activation function and Cross-Entropy is loss function, from [49].

From this point, the output is compared with the correct answer, and the difference is measured using a loss function, such as binary cross-entropy Loss and Hinge loss for example [50]. As they represent the gap between the prediction and the correct answers, the objective is to minimize them. The gradient of the loss is computed and used to update the weights and biases, multiplied by the learning rate, a parameter inserted in advance to moderate the adjustment by the user. This is called the backward propagation [51].

The previous steps are repeated for every image in the training set. This process is reiterated several times, with each iteration called an epoch, aiming to minimise the loss and maximise the accuracy of the network [52].

- The testing session involves presenting different images to the network, not encountered during training, and evaluating its performance through forward propagation [52]. Evaluation tasks include specific analysis such as accuracy or recall.

A scheme summarising what have been explained is shown in Fig.3.6, with a training set of 60,000 images, and a testing set of 10,000 images and one showing an example with softmax as an activation function and Cross-Entropy as a loss function is displayed in Fig.3.7, from [49].

3.1.4 Convolutional neural networks

Convolutional neural networks (CNNs) are types of neural network specifically designed for processing structured grid-like data, such as images. It's a deep learning algorithm that

has proven to be highly effective in various computer vision tasks like image classification, object detection, segmentation, and more. A deep learning algorithm, in contrasts with shallow neural networks, consist of multiple layers to model and understand complex data representations (they usually have more than two hidden layers in their architecture).

Previously introduced neural networks are based on the connection of a flattened image, which has each of his inputs connected to the first hidden layers, they are called fully-connected networks. In contrast with fully-connected networks, requiring light resource needs but losing spatial information because of the flattening of the image, CNNs show the best results in terms of classification for some sets [53].

Before connecting the input and the neurons of the hidden layers, CNNs apply several filters in order to modify the incoming signal into the network:

- A Convolutional layer performing dot products between the grid-liked structure data, considered as a matrix, and a kernel, a smaller matrix being one of the learnable parameters of the network. Performing this product for every spatial possibility, a smaller resulting matrix is obtained. A scheme explaining the functioning of the convolutional task is observed in Fig.3.8
- A Pooling layer down-samples the feature maps generated by the convolutional layers. Most common pooling tasks retain the maximum value within each region of the obtained matrix, or average pooling, computing the average value of the region. Pooling helps reduce the dimensionality of the feature maps while retaining the most important information. Between each operation for both of these tasks, the values of the input matrix are shifted by a number, called the stride. A scheme explaining the functioning of the pooling task is explained in Fig.3.9.
- Activation functions, that are the same as introduced earlier in this Section. Convolution, pooling and activation task are considered a whole task, and can be repeated various times for reducing the dimensionality of the input.
- Fully connected layers connect the resulting matrix with the neurons composing the hidden layers, the same way fully-connected network do it with the flattened input.

While CNNs have better performance with image classification tasks, they also require higher resources, and they are not the best candidates for applications on the Zybo Z7-10 board. However, for widening our study, they will be used with the MobileNet Architecture [56] [57].

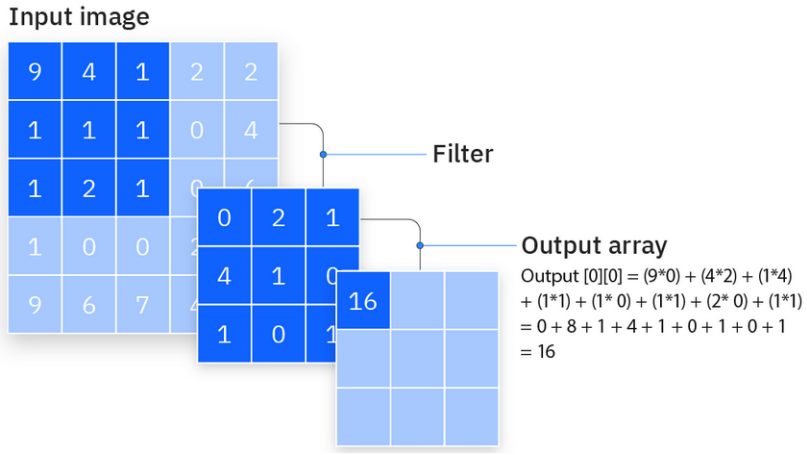


Figure 3.8: Scheme describing the operation of convolution made with an input image of 5x5 input matrix, and a 3x3 kernel, from [54].

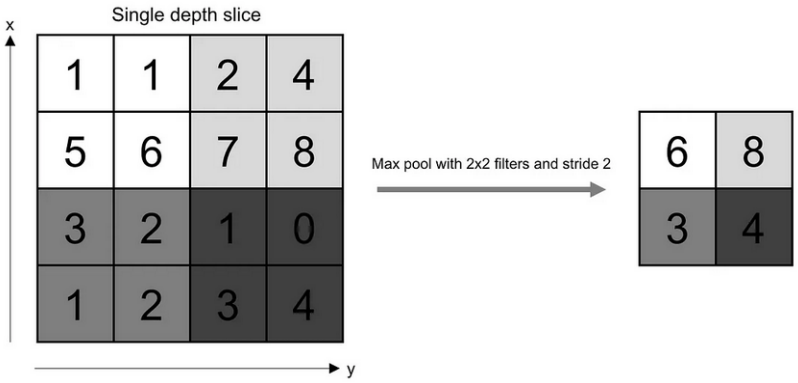


Figure 3.9: Scheme describing the operation of max pooling on a 4x4 matrix with a 2x2 filter and stride 2, from [55]

3.1.5 Overfitting

One of the most prevalent and common challenges in supervised neural networks and machine learning models is overfitting. This phenomenon arises when a model learns to perform well on the training data but fails to generalise to new, unseen data. In other words, the model memorises the training data instead of learning the underlying patterns or relationships, resulting in poor performance on unseen data. This tendency often occurs when the network is trained for too many epochs, causing it to excessively minimise the loss function and closely fit the training set while failing to meet the requirements of the testing set.

To illustrate this learning characteristic, an example is depicted in Fig.3.10, where a supervised neural network is tasked with a classification assignment. The training set is composed of 11 red dots and 11 grey stars, and the network is trained to learn a cut to

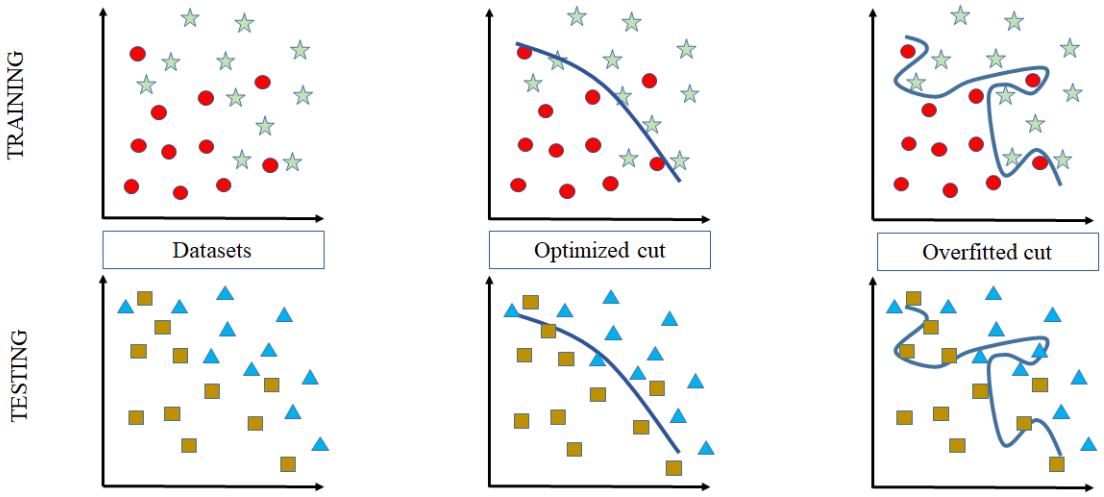


Figure 3.10: Scheme describing how overfitting acts in machine learning.

distinguish between these elements. An optimised boundary would achieve a 77% accuracy on the training set, whereas overfitting would result in 100% accuracy. When these cuts are applied to a testing set consisting of 11 brown squares and 11 blue triangles, the optimised cut yields an 82% accuracy, whereas the overfitted cut incorrectly applies the overfitting phenomenon to unseen data, resulting in a 59% accuracy. Overfitting is a crucial consideration in optimising network performance.

3.1.6 Neural Networks in physics

Neural networks and their derivatives, such as Modular Neural Network designed to handle specific and distinct tasks to different modules of the network, or Recurrent Neural Network optimised for the understanding of sequential or temporal data, have become so powerful and handy they have been integrated in many physics experiments revolutionising the interpretation of data [43] [58] [59]. One main field where neural networks are used in experiments is particle physics [60]. Indeed, high-energy particle collisions emit a massive amount of experimental data [61] and traditional methods prove not to be as robust as neural network for analysis and event classification, first in [60] and enhanced in [62]. They have shown exceptional results working in domains like tracking and identification of particles in experiments occurring in the Large Hadron Collider, for example. Not only in particle physics but also in domains like astronomy and cosmology, neural networks have been proven valuable for works like the processing and the interpretation of astronomical observations [63].

However, when neural networks are set on frontend facilities in particle experiments or subject to cosmic rays in outer space, as seen in Chapter .1, the damages in the electronic,

resulting not only from charged particles, have repercussion on the functioning of the networks. If one wants to use neural networks in particle ambient, it is mandatory to study the consequences on the performances of such an important analytical tool.

3.2 FINN and Brevitas projects

As discussed in Sec.2.2, our objective is to integrate a neural network into the Zybo Z7-10 board. To achieve this goal, the initial step involves using the FINN framework [64], an experimental project developed by Xilinx Research Labs aimed at optimising quantised neural networks (QNNs) in embedded systems.

3.2.1 QNNs and BNNs

QNNs represent a type of neural network where weights and activations are encoded with a reduced number of bits compared to traditional neural networks. In traditional neural networks, weights and activations are typically represented as floating-point numbers, necessitating higher precision and consuming more memory and computational resources.

QNNs offer a more efficient alternative, with Binary Neural Networks (BNNs) representing a stronger form of QNNs proposed in [65]. In BNNs, weights and activations are reduced to only binary values (+1 or -1), resulting in a significant reduction in memory footprint and computational complexity of the network. This makes BNNs highly suitable for deployment on resource-constrained devices such as microcontrollers, embedded systems, and low-power edge devices, positioning them as ideal candidates for implementing neural networks on the Zybo Z7-10 board.

However, while QNNs already achieve reduced costs while maintaining performance, BNNs take this reduction to an even greater extreme. This efficiency comes with a trade-off, as maintaining accuracy and performance becomes a challenge for BNNs compared to QNNs.

3.2.2 Brevitas library and functioning of the network

Prior to implementing the BNN into the Zybo Z7-10 using FINN, the BNN must be created and trained in a manner that allows FINN to convert all characteristics of the trained network into electronic format. To achieve this, the Brevitas library [66], a research project developed by Xilinx, has been designed with the objective of ensuring that the trained model files are compatible with FINN, and can be seamlessly integrated into the electronics.

The Brevitas library is written in Python and utilises common PyTorch functions for creating different layers and computing methods, while also implementing the quantisation

process. Subsequently, we will describe the network's functionality and the functions utilised to obtain the results presented in the following sections.

For Fully-Connected (FC) layers (in contrast to convolutional networks, explained in Sec.3.1.4), which is the approach previously outlined for the functioning of neural networks in image processing, the image is flattened and provided to the network as an input array. Each element of the array corresponds to the value of a pixel. The various characteristics of the network are summarised as follows:

- The dropout, which represents the probability of randomly setting elements of the input tensor to zero, is set to 0.2. The input dimension is determined by the number of pixels composing each image,
- The number of layers and neurons within each layer is selected. In the subsequent cases, if not specified, the network consists of three layers, each with sixty-four neurons,
- The dimension of the output layer depends on the number of classes present in the dataset,
- The number of bits allocated to the weights and activations is set to 1 for both,
- The number of epochs is set to 500 for having performance good enough, no risk of overfitting and to reduce the time of computation, a plot showing the mean of recognition in function of the number of epochs is observed in Fig.3.11,
- The learning rate is set to 0.02 and evolves according to the ADAM optimiser [67],
- Weights are initialised with a uniform distribution between -1 and 1,
- Biases are initialised to 0,
- The loss function employed is the Square Hinge loss. It computes the square of the Hinge loss value.

The network is trained using the previously described characteristics, and the final model of the trained network is saved into a file formatted to be read by the FINN framework.

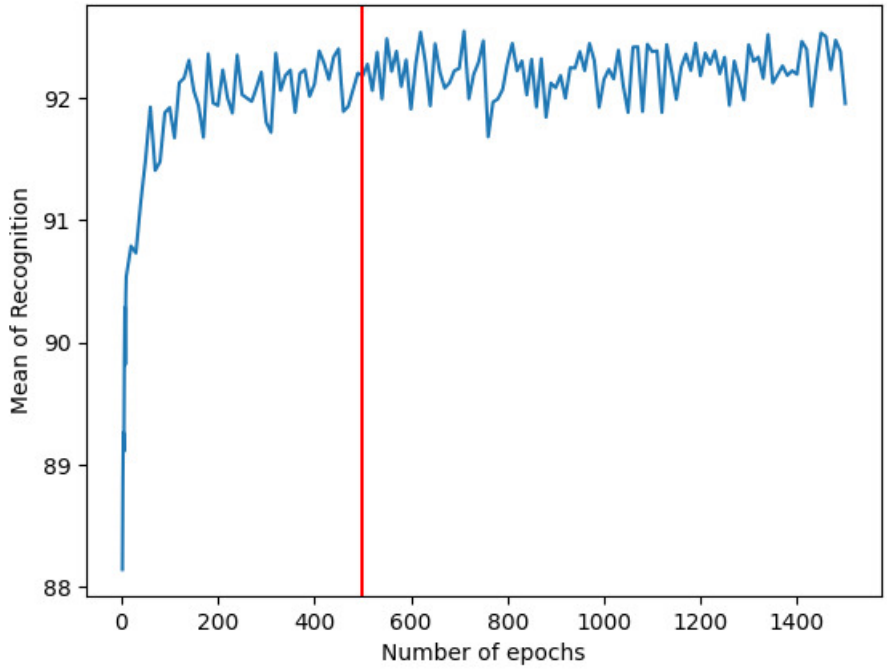


Figure 3.11: Plot displaying the mean of recognition of the network in function of the number of epochs. The red line stands for 500 epochs.

3.2.3 FINN framework

Upon completion of the network training, a file is generated containing all the necessary information for testing purposes. The FINN framework utilises this file and assigns various neural components to different electronic modules. An illustration of the allocation of different layers in the electronics is presented in Fig.3.12, and another illustration showing the various connections between the neural components by the electronics is displayed in Fig.3.13

By utilising specific software such as Vivado, a file is generated and implemented into the Zybo Z7-10, which is readable by PYNQ. This file allocates the modules as described by FINN. Once the model has been implemented into the FPGA, the board becomes accessible via Jupyter Notebook for communication. Using Python commands, the model can then be tested. A scheme summarising the functionning of Brevitas, FINN and PYNQ is displayed in Fig.3.14.

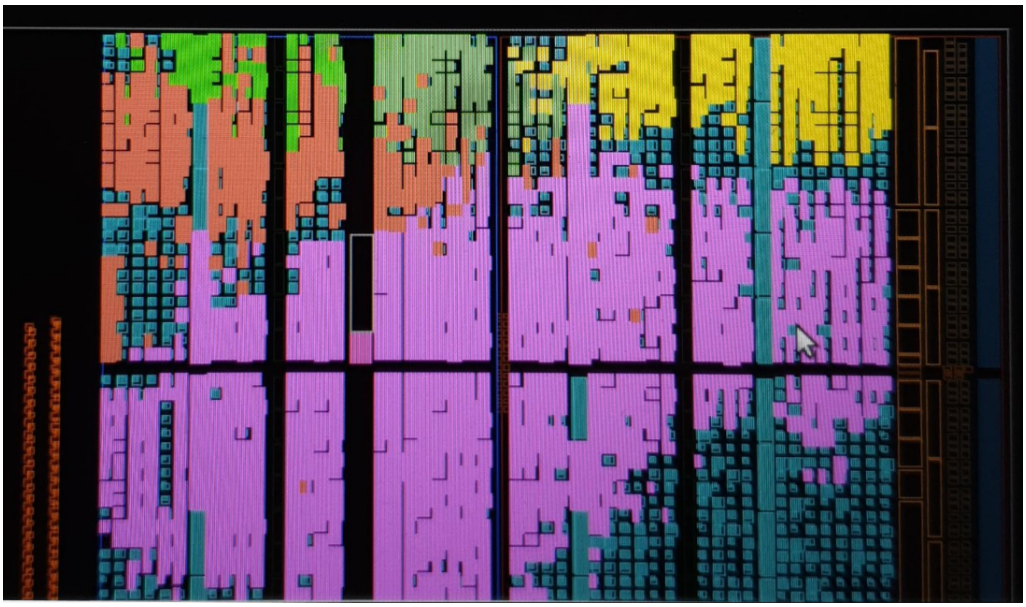


Figure 3.12: Display of the electronics composing the Zybo Z7-10 board. The network elements allocated to electronic ones are coloured. Purple for the input layer, yellow, orange and khaki for the hidden layers and bright green for the output layer.

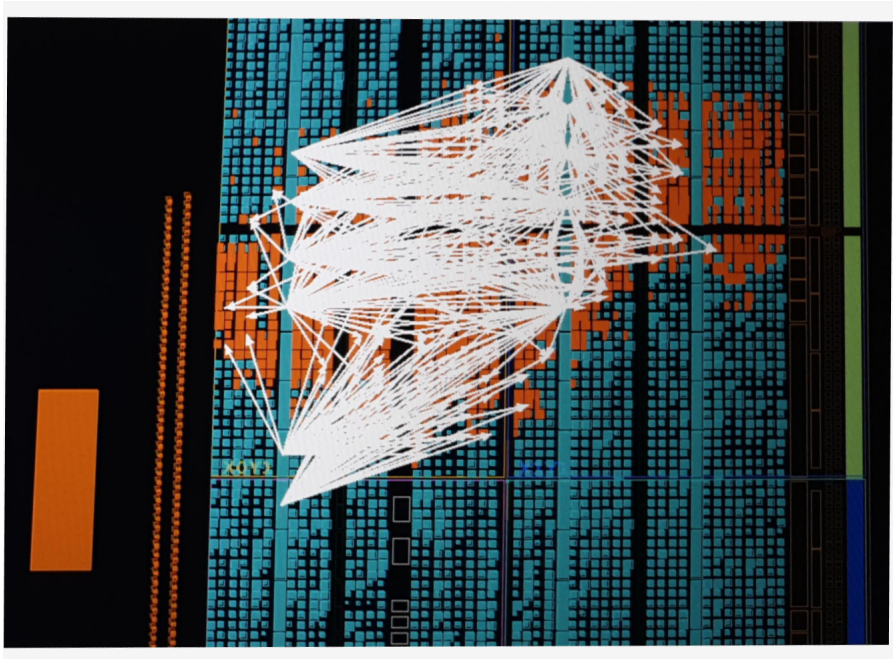


Figure 3.13: Display of the electronics composing the Zybo Z7-10 board and the connections between elements. Each of the 784 white arrows represents the communication of a single pixel to a neuron of the input layer, in orange in this figure.

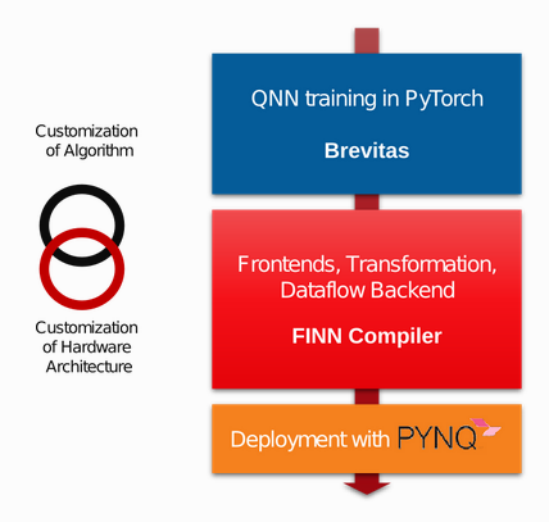


Figure 3.14: Scheme describing the organisation between the Brevitas library, the FINN framework and PYNQ, from [68].

3.3 MNIST dataset and performances

3.3.1 MNIST dataset

The Modified National Institute of Standards and Technology (MNIST) databset [69] provides a 10-class database for image recognition with low-dimensional pictures, making it suitable for initial tests with the Zybo and the Brevitas project, which supplies examples using this database. We prefer using the optimised network using LFC, and allocating 1 bit to input quantisation, 1 bit to weight quantisation and 1 bit to activation quantisation. This preference is based on its support by the Zybo, and it has shown the highest accuracy results in this configuration.

The MNIST dataset is a 10-class database of images displaying handwritten digits from '0' to '9'. The sample dimensions are 28x28 pixels, introduced in greyscale levels. The total set comprises 60,000 training images and 10,000 testing images. Examples of samples are shown in Fig.3.15

3.3.2 Accuracy, F_β-Score and Mean of Recognition

The initial architecture used with the Brevitas project comprises an input layer with 784 neurons (equivalent to the number of pixels in a single image from the MNIST database), three hidden layers with 64 neurons each, and an output layer of ten neurons. The selection of the input and output layers is based on MNIST parameters, while the configuration of

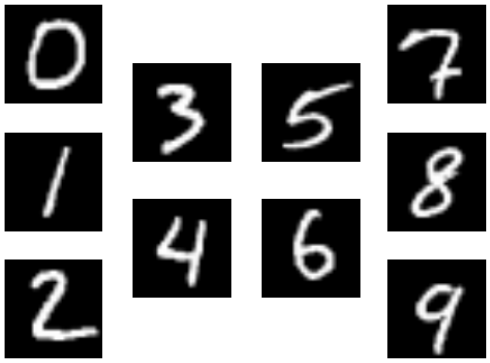


Figure 3.15: Samples of digits in the MNIST database.

the hidden layers is chosen for its effectiveness in the Brevitas project and adherence to constraints arising from the limited resources of the Zybo board. A visual representation of the architecture is shown in Fig.3.16.

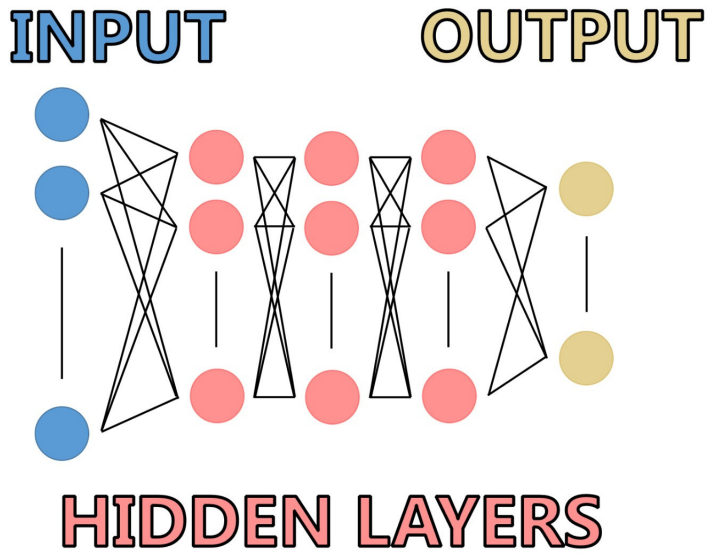


Figure 3.16: Scheme displaying the architecture of the network used for the job with the MNIST database. The input layer is composed of 784 neurons, three hidden layers of 64 neurons each are set, and the output layer is composed of ten neurons.

When an image is input into the network and undergoes forward propagation, the network generates a prediction. During a training session, this prediction is utilised to calculate the loss function. In the case of a testing session, the result is stored and can be visualised. By aggregating predictions for each image corresponding to a designated digit, a vector is formed. An illustration of this result after the network has been trained is presented in 3.3 for the digit '0,' where each element corresponds to the number of predictions made by the network for the respective digit. In this example, 980 different images of '0' have been

provided; 956 of them have been correctly predicted, one has been predicted as a '2', 6 as a '5' and so on.

$$\begin{array}{rc} \textbf{digits} & \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \mathbf{0} = & \begin{bmatrix} 956 & 0 & 1 & 0 & 0 & 6 & 9 & 3 & 5 & 0 \end{bmatrix}. \end{array} \tag{3.3}$$

For each digit, a similar vector is obtained, and all these vectors can be consolidated into a single table, Tab.3.1 is an example of it. The diagonal of this table summarizes the correct predictions made by the network.

Input \ Prevision	0	1	2	3	4	5	6	7	8	9
0	956	0	1	0	0	6	9	3	5	0
1	0	1115	4	2	0	1	3	0	10	0
2	11	5	959	10	11	4	8	8	15	1
3	4	4	22	882	1	35	1	13	39	9
4	2	5	4	0	895	0	18	3	3	52
5	10	2	6	18	7	819	8	3	12	7
6	16	3	7	1	5	15	907	0	4	0
7	2	13	21	4	6	1	0	947	5	29
8	6	6	14	14	12	22	8	11	872	9
9	8	9	4	8	20	8	3	14	13	922

Table 3.1: Table gathering all the predictions made by the network for a specific seed with 500 epochs. In green are the True Positives, in blue are the False Positives and in red are the False Negatives for the digit '0'.

A feature indicating the precision of the network for a given testing session is the Accuracy, defined as

$$Acc = \frac{\sum_{n=0}^{9} Corr_n}{\sum_{i=0}^{9} Len_n}, \tag{3.4}$$

where $Corr_n$ is the number of correct predictions by the network for the digit 'n' and Len_n represents the total number of images in the testing dataset for the same digit. The Accuracy for a given digit 'n' is given as

$$Acc_n = \frac{Corr_n}{Len_n}. \tag{3.5}$$

In the case of the results inserted in Tab.3.1, we observe an accuracy $Acc = 0.93$. While Accuracy is relevant for assessing the precision of the network, the unequal distribution

of images in the testing dataset across different digits prompts the introduction of two features crucial for characterising the network.

To define the $F_\beta - score$, we first need to introduce the precision and recall as:

$$\begin{aligned} precision &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ recall &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\,, \end{aligned} \tag{3.6}$$

where, for a digit *n*, *TP* stands for True Positive corresponding to values correctly predicted by the network. *FP* stands for False Positive, indicating predictions of *n* when the input is not, and *FN* stands for False Negative, representing the incorrect predictions when *n* is the input. An example of these parameters with digit '0' are shown in Tab.3.1. The precision represents the ratio of relevant items in the retrieved ones, while recall stands for the retrieved items among the relevant ones.

The $F_\beta - score$ is then defined as:

$$F_\beta = (1+\beta^2)\cdot\frac{precision\cdot recall}{(\beta^2\cdot precision)+recall}. \tag{3.7}$$

The preference for F_β over Accuracy arises from the flexibility of choosing β to assign more or less importance to False Positives and True Negatives. Accuracy, on the other hand, considers only False Negatives with a fixed weight. While this may not be critical for the MNIST dataset, it becomes significant in cases where predicting a certain class has more impact than others. Additionally, since the number of images in the testing dataset is not equally distributed among all digits, F_β takes this into account in its computation. Both parameters are computed in Tab.3.2, and it can be noticed the F_1 score, which adds False Positives and assign them a weight along with True Negatives, modifies some values. In these examples, the F_1 score for digits '3' and '5' have a difference of 0.003 whereas their accuracies have a difference of 0.045. This difference is explained by the fact the digit '3' is less predicted by the network when another class is provided as input. The last line of Tab.3.2 are the mean of the upper lines. The Mean for Acc_n corresponds to the Mean of Recognition.

The second feature we want to use instead of the Accuracy is called Mean of Recognition (MR) and is defined as

$$MR = \frac{1}{10}\sum_{i=0}^{9} Acc_i\,. \tag{3.8}$$

Digits	Acc_n	F_1
0	0.976	0.958
1	0.982	0.971
2	0.930	0.925
3	0.873	0.905
4	0.911	0.923
5	0.918	0.908
6	0.947	0.943
7	0.921	0.933
8	0.895	0.893
9	0.914	0.904
Mean	0.935	0.926

Table 3.2: Table summarising the Accuracy and F_1 score for the values provided in Table 3.1.

Unlike the Accuracy, the MR doesn't take into account the difference in the number of images for each digit, they are all considered equally in the final result.

Since we consider the MR more relevant than the Accuracy and the F_β-score for studying the effectiveness of the network with the MNIST dataset because it provides equal proportion to each digit, we will prioritize the use of this metric.

3.4 Noise implementation

As demonstrated in Sec.2.1, cosmic waves do interact with electronics. Given that the project's objective is to connect the Versal VCK190 to a camera on a spacecraft, it is essential to study the consequences of these interactions on both components. This section will specifically delve into the effects of particles interacting with the camera and their implications for the network's operation.

3.4.1 Modification of the database

Technique of implementation

The camera used is a Pcam 5C designed around the Omnivision OV5640 5-megapixels sensor, a Complementary Metal-Oxide-Semiconductor (CMOS) images sensor. Comprising an array of light-sensitive diodes, CMOS sensors convert light into electrical signals, storing them for computer-readable data. These signals serve the dual purpose of generating images or videos and quantifying the light levels within a given scene, a deeper explanation of

CMOS can be retrieved in [70]. CMOS sensors are sensible to cosmic rays as the radiation, primarily composed of charged particles, interact not only with the diodes but also with the electronic devices comprising the apparatus as showed in [71]. Given our goal to train a network with images acquired by the camera set on satellites, it is essential to replicate this interference [7] [8] [11] [12].

We base our approach on the methodology presented in [72], where we replicate noise in the MNIST database by randomly switching the value of pixels in both the training and testing sets. Although they use black and white images, their method seems coherent with the results shown in [71].

To do so, we propose not only to add white pixels to the black background of the images, but also to modify the grey and white pixels by switching their colour relatively to the grey bar. In black and white images, each pixel corresponds to a value between 0 and 255, with 0 being black and 255 being white. The objective is to perform a symmetrical switch on random pixels regarding the centre of the grey bar, which falls between 127 and 128, as illustrated in Fig.3.17. For instance, a pixel with a value of 0 becomes 255, one with a value of 255 becomes 0, one with a value of 126 becomes 129 and so on.

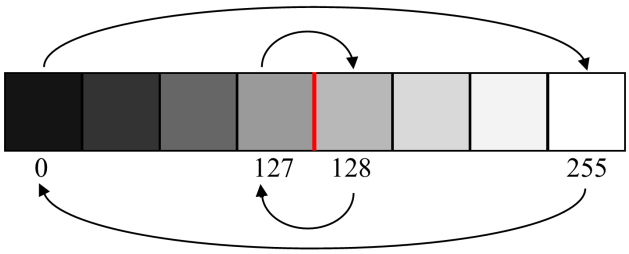


Figure 3.17: Greyscale used for the switch of pixels.

The ratio of pixels switched in each image is adaptable and some examples are shown in Fig.3.18 where implementations of noise with 0%, 5%, 10% and 100% are displayed.

Implementation of noise

During the experiment, the satellite will acquire images for training the network. While some objects may appear in multiple images, the noise induced in the images will vary, and the perturbation of the object will differ each time. Thus, to ensure exhaustive results, considering that the training set is provided multiple times through the epochs, implementing the noise directly into the dataset to obtain a noisy dataset is not suitable. This would result in providing identical noisy images every time. To prevent this, we choose to inject the noise

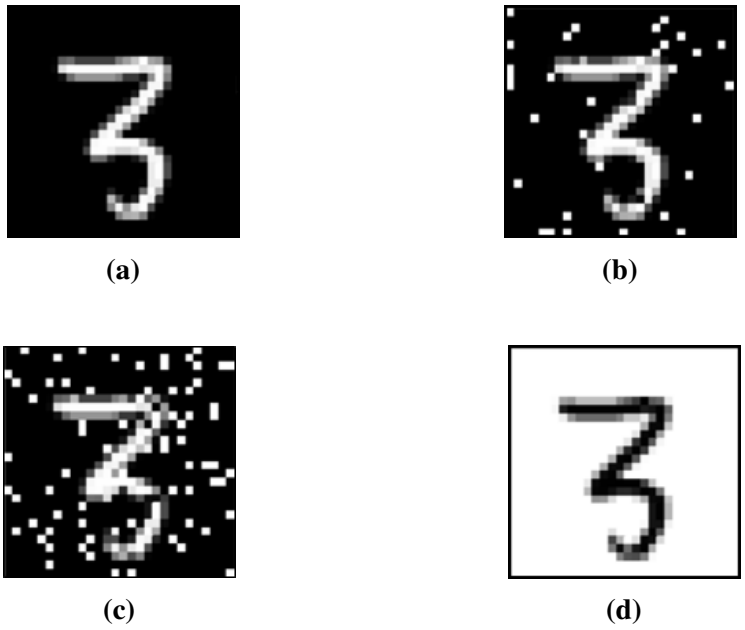


Figure 3.18: Examples of the random pixels implemented into the MNIST images to replicate noise induced by cosmic-rays. The noise implementation is illustrated for **a** 0%, **b** 5%, **c** 10% and **d** 100% noise injected.

into the image right before it is provided to the network for forward propagation, ensuring that the switched pixels are always different.

Oppositely, the testing set is provided only once to the network. While injecting noise right before forward propagation is not as crucial as it is in training, we prefer having a noisy testing dataset. This allows us to use it again with other trained networks, comparing the reliability of each regarding the same sample. Fig.3.19 illustrates how the noise has been implemented to closely mimic real-life situations.

3.4.2 Repercussions on the network predictions

Now that the database has been modified, a thorough investigation into the ensuing effects on the network and its predictions can be conducted. Even if the percentage of noise implemented in the training and testing images in real situations should be close because the flux of cosmic rays during both training and testing sessions is expected not to vary significantly, variations can still occur as introduced in [73] with slow evolution with regard to testing times. Thus, each variation situations need to be studied, at fixed amount of noise since. In physics, the flux is defined as the number of particles crossing a unit area per unit of time.

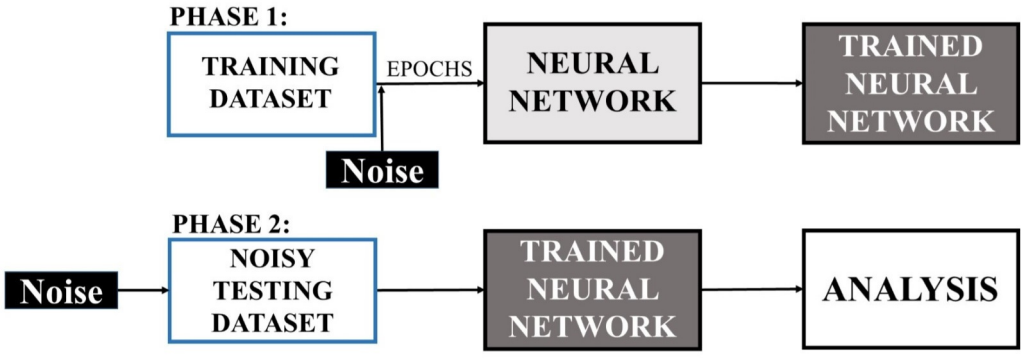


Figure 3.19: Scheme describing how noise is injected into the MNIST dataset for training and testing sessions.

In order to investigate this, we train the network with a noise injection ratio (*R*) into the images, following the method described in Sec.3.4.1. Subsequently, we provide the trained network with the noisy testing dataset injected with different ratios of switched pixels *r*. We examine Acc_n for each digit and plot them together for a given *R* along with the MR. This study is conducted with *r* = 0%, 1%, 2%, 5% and 10%. Due to the fact we have a large number of samples, the probability of success is defined as Acc_n and each prediction is values as 'correct' or 'incorrect' with independence between the results, the conditions to compute the uncertainty using a binomial confidence interval are satisfied

$$C = \pm\sqrt{\frac{Acc_n(1-Acc_n)}{Len_n}}. \tag{3.9}$$

Unmodified dataset

As illustrated in Fig.3.20, we initiate the network by running it with a noise injection ratio *R*=0% in the training dataset, meaning it remains unmodified. For all digits, we observe that as *r* increases, both Acc_n and the MR decrease, with different proportionalities. Additionally, some digits *n* exhibit substantial discrepancies, while others show narrower differences between the two extreme stages of noise implementation. For example, the difference for Acc_3 between *r* = 0% and *r* = 10% is less than 5% of accuracy, whereas the same difference for Acc_1 exceeds 40% of accuracy. As a result of these outcomes, higher noise proportions injected in the testing dataset *r* leads to higher values of standard deviation, which we aim to minimize for homogeneity. The standard deviation being computed as the square root of sample variance, it is independent of the binomial confidence interval.

Another noteworthy observation in Fig.3.20 is that the network trained without noise performs better in recognizing some digits with noisy images than non-modified ones. This is evident in the case of Acc_4, which demonstrates a higher level of precision for $r = 1\%$ and $r = 2\%$ of noise compared to clean images. Moreover, it achieves the lowest value among all noise injection levels for Acc_3. In certain cases, the network appears to be more robust when provided with altered images rather than untouched ones.

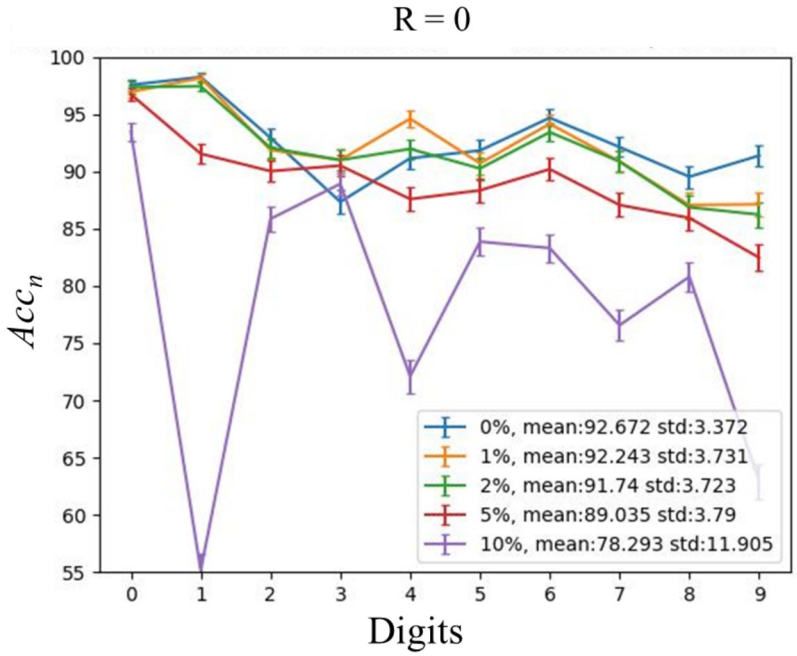


Figure 3.20: Acc_n for all digits, using a network trained with a training dataset with a ratio R=0% of switched pixels. The network is then tested with the testing dataset, which has $r = 0\%$, 1%, 2%, 5% and 10% of noise injected.

R=1% and comparisons

Now we have observed the behaviour of the network trained with a non-modified set, we can examine its response to noise injection. We initiate with a very low percentage of perturbation, injecting only *R* = 1% of switched pixels into the training dataset. The results are presented in Fig.3.21 again illustrating with the Acc_n for each digit at noise levels of *r* = 0%, 1%, 2%, 5% and 10% in the testing set.

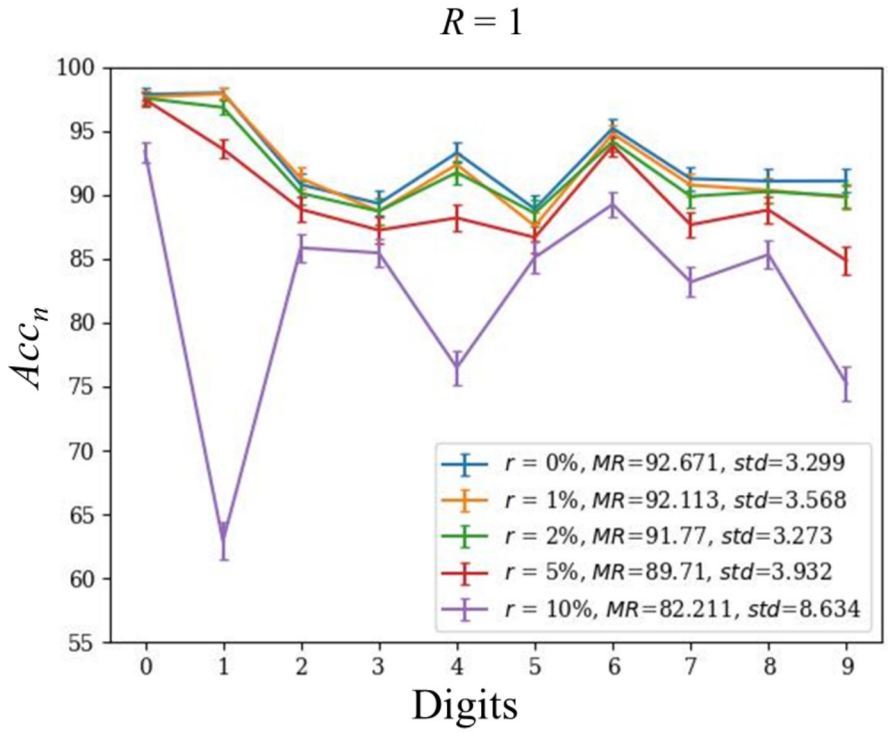


Figure 3.21: Acc_n for all the digits, with a network trained with a training dataset with a ratio R=1% of switched pixels. The network is tested with the testing dataset with a rate of 0%, 1%, 2%, 5% and 10% of noise injected.

While the difference in MR for the clean and the two lowest implementations of noise in the testing set is negligible, the standard deviations show a slight improvement with respect to the plots in Fig.3.20 . At *r* = 5% injection, the MR increases to a small extent, but the standard deviation decreases slightly. The most significant improvement in the values provided by the network occurs at *r* = 10% injection noise. Here, not only does the MR increase by almost 4%, but the standard deviation decreases by more than 3. Digits with initially low Acc_n ('1', '4' and '9') noticeably increase despite the fact they remain lower than the other digits. Despite adding only a small number of perturbations in the noisy

training set, the impact on the network's robustness when provided with highly perturbed inputs is significant.

Another aspect of the network's response to the injection of noise, not evident in the values, is the behaviour of the curves relative to each other. The curves for $r = 0\%$, 1% and 2% almost overlap, and even the curve for $r = 5\%$ noise tends to follow this pattern. In order to quantify this effect, we plot the difference between the curves relative to the one with the highest MR, using r_0, the curve corresponding to r=0%, as a reference. This plot is shown in Fig.3.22 where the curve for r = 1% has six points within the confidence interval and three outside with matching uncertainties. The curve for r = 2% has four points inside the interval, four outside with matching uncertainties, and only one point fully outside. The curve for r = 5% has only one point within the interval and another one outside with matching uncertainties, giving clues about how similar are the curves.

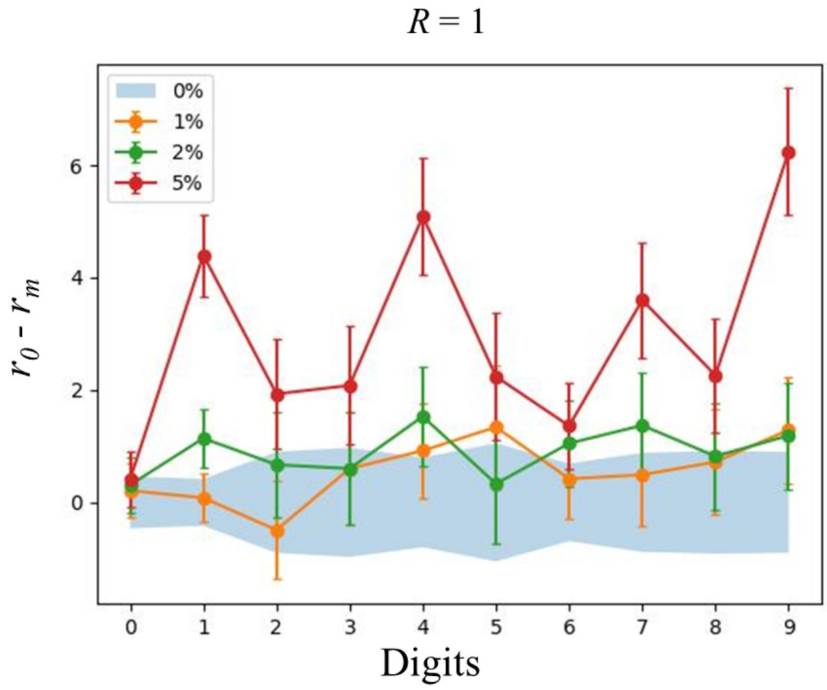


Figure 3.22: Difference of Acc_n with a training dataset with R = 1% between values for r = 0% and the values for r = 1%, 2% and 5%.

To illustrate the concepts introduced in Sec.3.4.1 regarding the methods of noise injection, we can plot the Acc_n for different level of noise rate in the testing dataset while the database supplied for training is modified by injecting noise and then provided with

the same sample for every epoch. The results for this study are presented in Fig.3.23 and Fig.3.24 shows the difference in percentages between the real-life situation method and the percentages when the database is modified and provided several times with the same object.

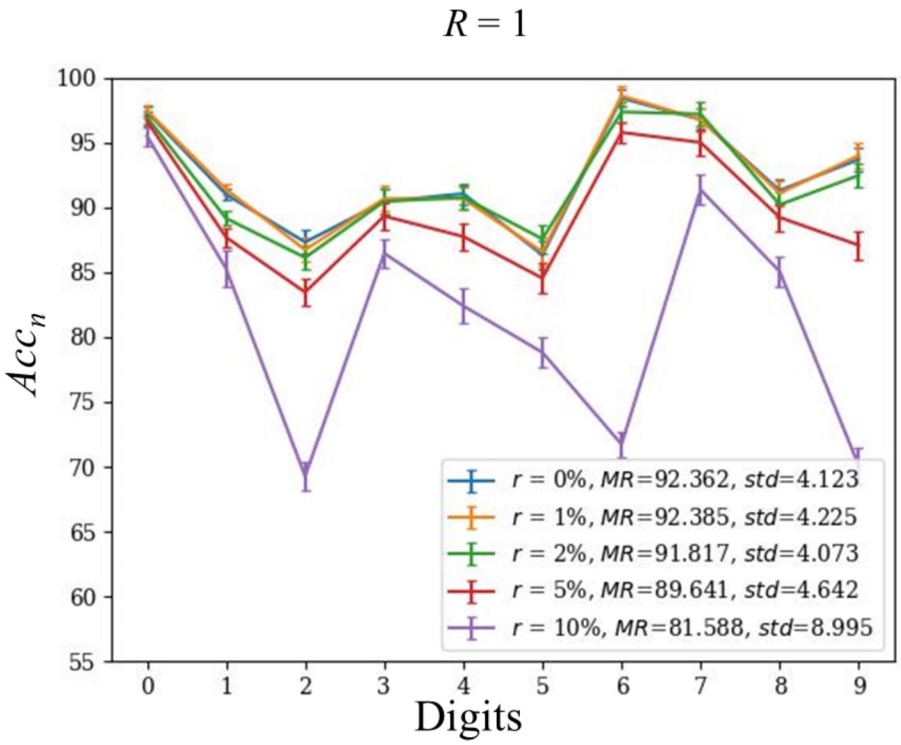


Figure 3.23: Acc_n for all the digits, with a network trained with a training noisy dataset with a ratio R=1% of switched pixels. The network is tested with the testing dataset with a rate of 0%, 1%, 2%, 5% and 10% of noise injected.

Regarding the MR, there is no significant modification for any ratio of noise injected into the testing set. The highest variation is observed for the maximum noise level, and it does not exceed a percent, which is negligible regarding the uncertainty. However, the standard deviation has higher values for all proportions of noise implementation. This suggests that the network has more heterogeneous correct predictions when noise is injected, and the real-life injection brings more consistent predictions. Notably the negative peaks for r = 10% injection noise are not consistent for the same digits. While they were observed for digits '1', '4' and '9' in Fig.3.21, they are observed for digits '2', '6' and '9' in Fig.3.23, indicating that noise helps the network recognize some digits but hinders others.

In accordance with the observation brought with Fig.3.22 the curves with the smallest proportion of noise tend to overlap. This is evident for r = 0%, 1% and 2% and even for r = 5% with a noticeably lower accuracy. The differences between the curves, using r = 1% as a

reference for having the highest MR, are plotted in Fig.3.25. The points for the $r = 0\%$ curve are all within the confidence interval of $r = 1\%$ while only three out of the nine points for $r = 2\%$ are outside the range, and the uncertainty of only one of them does not match the region of interest. Regarding the $r = 5\%$ curve, no points are placed in the interval, but five of them have matching confidence intervals.

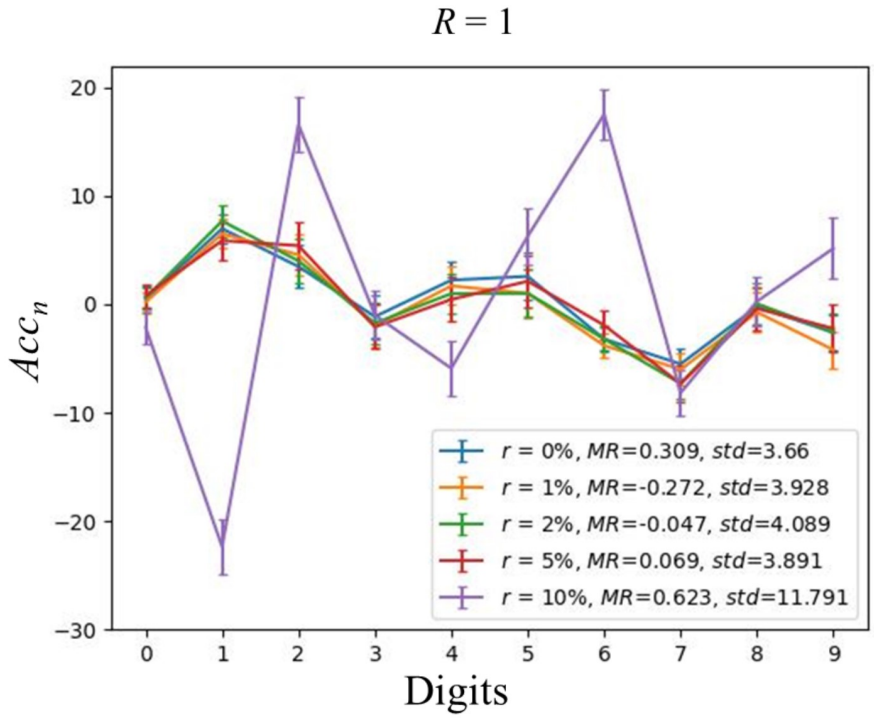


Figure 3.24: Difference of Acc_n between the results obtained with a set injected with noise right before the training, and a noisy dataset provided several times. This plot has been obtained, computing the difference between the curves of Fig.3.21 and Fig.3.23 for a given ratio r.

This observation is even more perceivable in Fig3.24 where the curves for $r = 0\%$, 1%, 2% and 5% are truly overlapping. In contrast, the curve for $r = 10\%$ noise has a very important standard deviation induced by the difference in the negative peaks caused by the modification of correct predictions introduced in the previous paragraph. We plot in Fig.3.26 the difference between the curves plotted in Fig.3.24 but excluding the curve for $r = 10\%$ for better clarity. The difference has been computed by choosing the curve closest to zero as a reference, which is for $r = 2\%$. Among the twenty-seven points displayed in this graphic, eleven are not within the confidence interval, and three of them have no matching uncertainty with $r = 2\%$.

Regarding the plot in Fig.3.23, aside from Acc_6 for $r = 10\%$ noise, this behaviour tends to be the same even with the highest proportion of noise injected: the negative peaks,

as well as the positive ones, occur for the same digits. The curves seem to tighten together, translated by a factor induced by the noise.

Now that the study has been conducted with $R = 1\%$ noise implemented in the training dataset, we aim to reproduce this study by injecting higher proportions of noise. This work has been conducted by implementing noise with ratios $R = 2\%$, 5% and 10% with real-life noise implementation selected, for relevancy with real-life situations.

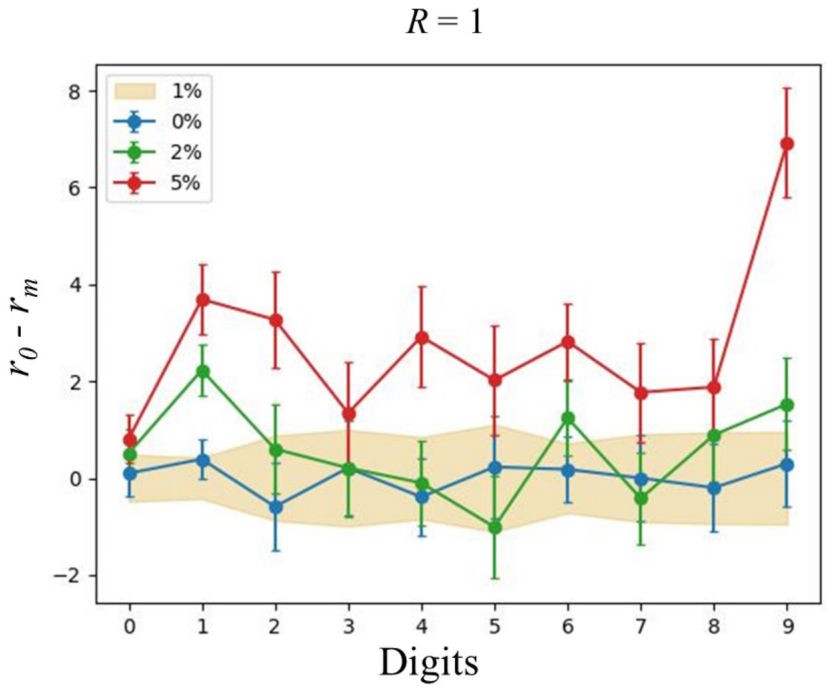


Figure 3.25: Difference of Acc_n with a training noisy dataset with $R = 1\%$ between values for $r = 1\%$ and the values for $r = 0\%$, 2% and 5%.

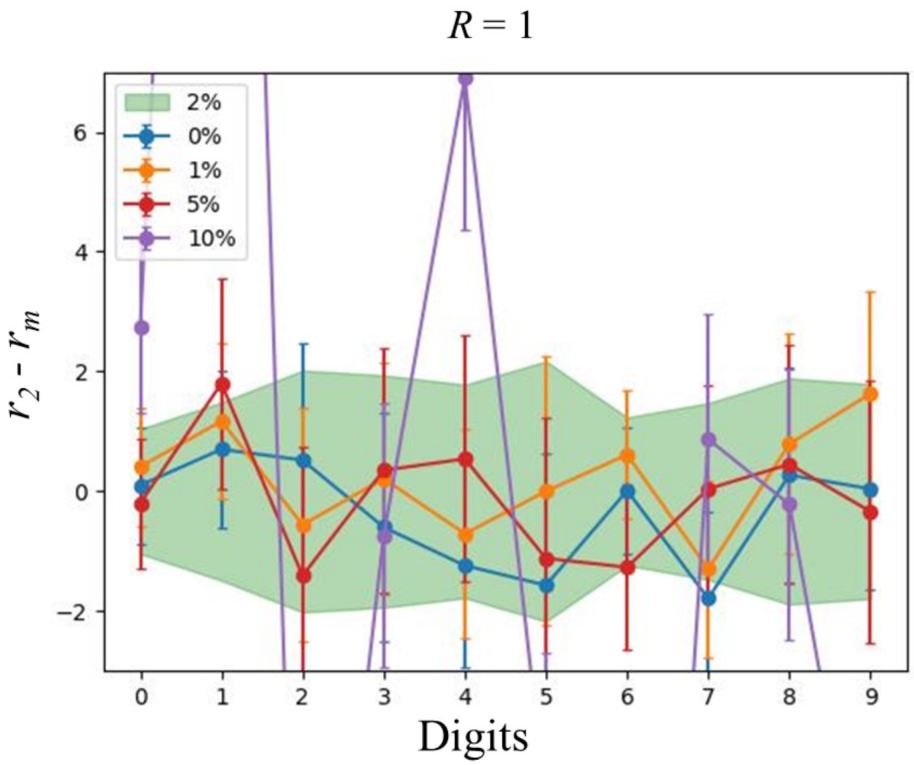


Figure 3.26: Difference of Acc_n with $R = 1\%$ for the difference between the two methods of noise injection. This difference is computed between values for $r = 2\%$ and the values for $r = 0\%$, 1% and 5%.

***R* = 2%**

In Fig.3.27, curves representing Acc_n for a ratio of noise injected in the training set *R* = 2% are plotted, while the noise implemented in the testing set varies with *r* = 0%, 1%, 2%, 5% and 10%. Comparing it to the results obtained with *R* = 1%, the MR has slightly decreased by less than 1% for the lowest amount of noise in the testing set (*r* = 0% and 1%), while the standard deviation has increased. For *r* = 2%, the MR remains almost the same, but the standard deviation has decreased, bringing homogeneity to the results. Meanwhile, we observe that the results for higher ratios of noise *r* have improved; the MR has increased for both *r* = 5% and 10%, while the standard deviation has decreased for both of them.

From these results, and those displayed in Fig.3.21, we notice that the higher the noise *R* implemented in the training set, the better the results for higher implementations of noise *r*. However, it stands out that with a fixed *r*, the results start to worsen as soon as $r < R$, regarding both the MR and the standard deviation. Finally, the peak of good results has been reach until now for $r = R$.

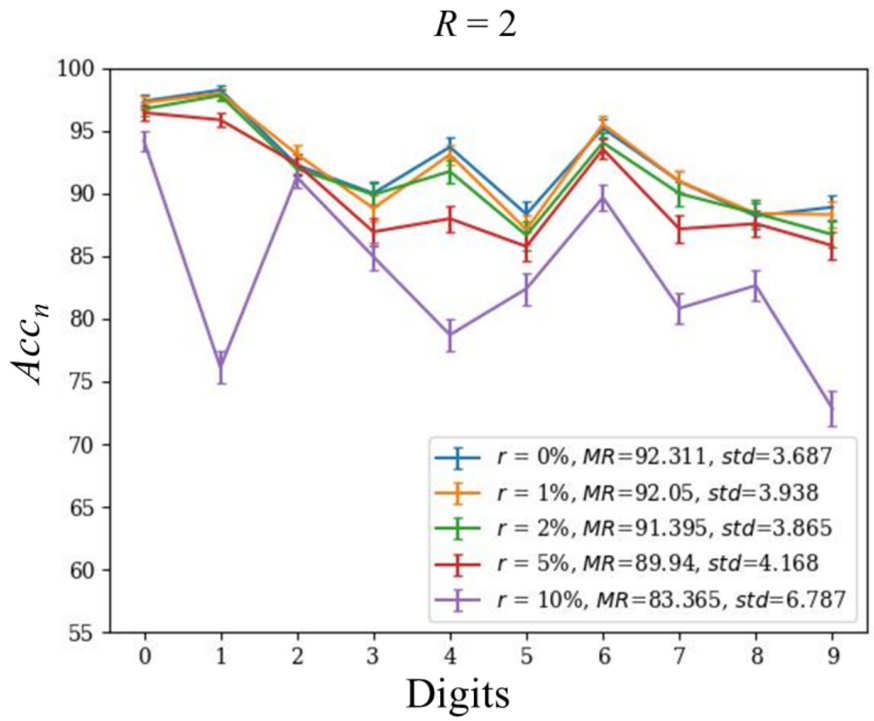


Figure 3.27: Acc_n for all the digits, with a network trained with a training dataset with a ratio R=2% of switched pixels. The network is tested with the testing dataset with a rate of 0%, 1%, 2%, 5% and 10% of noise injected.

In Fig.3.28 the difference in Acc_n between the curve with the highest MR in Fig.3.27 (representing $r = 0\%$) and the various curves with less than $r = 10\%$ is displayed. Eighteen out of the twenty Acc_n from $r = 1\%$ and 2% have uncertainties matching with $r = 0\%$, which is one less than the number in Fig.3.22. Besides, only three Acc_n values from the curve for $r = 5\%$ match the uncertainty with $r = 0\%$, while five of them matched in Fig.3.22.

For better visibility, the plots showing the Acc_n for a ratio of noise $R = 2\%$ with a noisy dataset, the behaviour of the curves shown by the difference between the curves of such a plot, the difference between the two methods of noise injection for a given ratio $R = 2\%$, and the behaviour given by the difference between the curves are all provided in Appendix A. Accompanying the plots, detailed analyses are also presented.

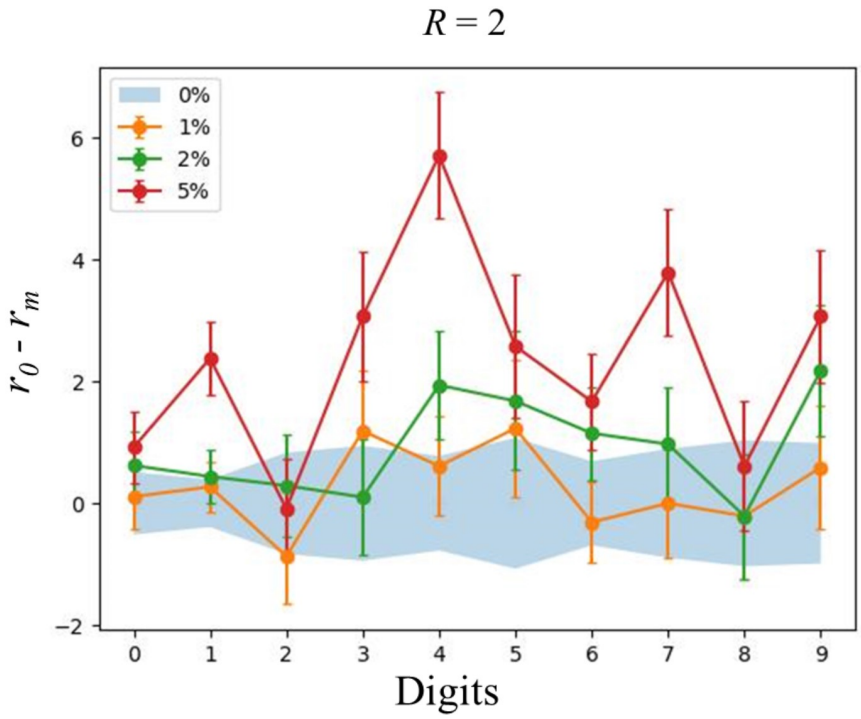


Figure 3.28: Difference of Acc_n with a training dataset with $R = 2\%$ between values for $r = 0\%$ and the values for $r = 1\%$, 2% and 5%.

***R* = 5%**

In Fig.3.29, curves representing Acc_n for a ratio of noise injected in the training set *R* = 5% are plotted, while the noise implemented into the testing set varies with *r* = 0%, 1%, 2%, 5% and 10%. Comparing it to the results obtained with *R* = 2%, the MR has decreased by more than 1% for the lowest amount of noise in the testing set (*r* = 0% and 1%), while the standard deviation has once again increased. For *r* = 2% and 5%, the MR decreased slightly, by less than 1%, and the standard deviation also increased. Meanwhile, we observe that the results for *r* = 10% have improved; the MR has increased by almost 2% and the standard deviation keeps decreasing.

Until now, increasing the ratio of noise injected in the training set by a ratio *R* improved the results for both the MR and the standard deviation for $r \geq R$. We notice this is not verified in this case for $r = R$ for both the MR and the standard deviation, while we keep having this phenomenon for *r* = 10%.

Furthermore, the manifestation of deteriorating results with the condition $r < R$ is verified. Not only is the network less accurate regarding the MR, but also the decreasing standard deviation makes the results more heterogeneous.

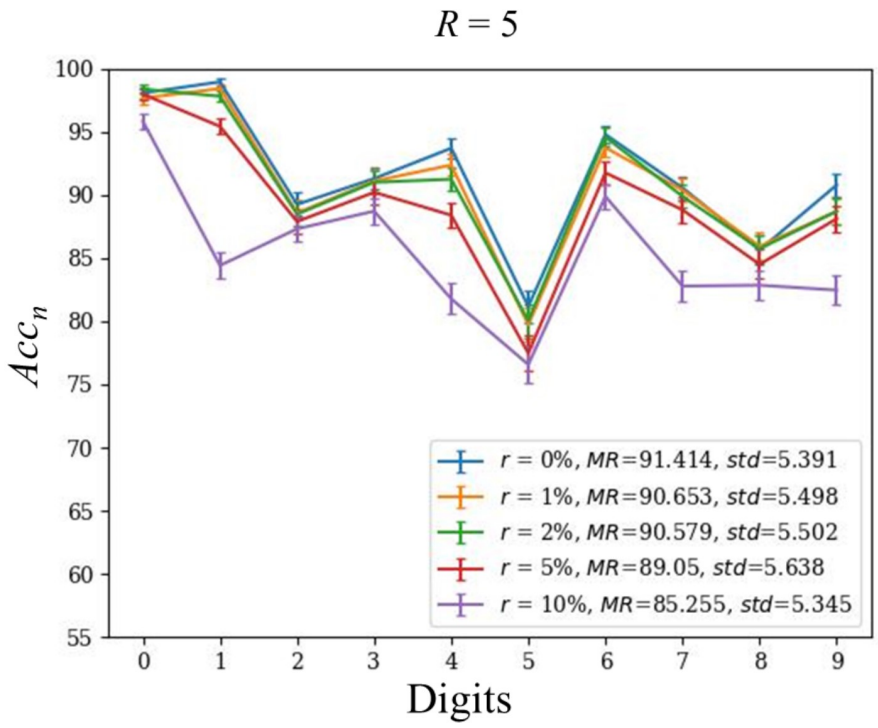


Figure 3.29: Acc_n for all the digits, with a network trained with a training dataset with a ratio R=5% of switched pixels. The network is tested with the testing dataset with a rate of 0%, 1%, 2%, 5% and 10% of noise injected.

In Fig.3.30, the differences of the Acc_n between the curve with the highest MR in Fig.3.29 (standing for $r = 0\%$) and the various other curves are displayed. Seventeen out of the twenty Acc_n values from $r = 1\%$ and 2% have uncertainties matching with $r = 0\%$, which is one less than the number in Fig.3.28. Besides, five Acc_n values from the curve for $r = 5\%$ match the uncertainty with $r = 0\%$, which is the highest number reached until now. Even though the results deteriorated for the lower injections of noise, we notice the higher ones tend to fit the behaviour of the curves with the smaller ratio implementation r. The difference with the curve representing the proportion of noise $r = 10\%$ has been plotted because of its relevancy. Indeed, we denote the difference between $r = 0\%$ and $r = 10\%$ tends to hit values smaller than 10%, and even for Acc_2, the uncertainties match.

In the same way it has been done for $R = 2\%$, the plots showing the Acc_n for a ratio of noise $R = 5$ with a noisy dataset, the behaviour of the curves shown by the difference between the curves of such a plot, the difference between the two methods of noise injection of a given ratio $R = 5\%$ and the behaviour given by the difference between the curves are all given in Appendix A for better visibility, along with the explanations.

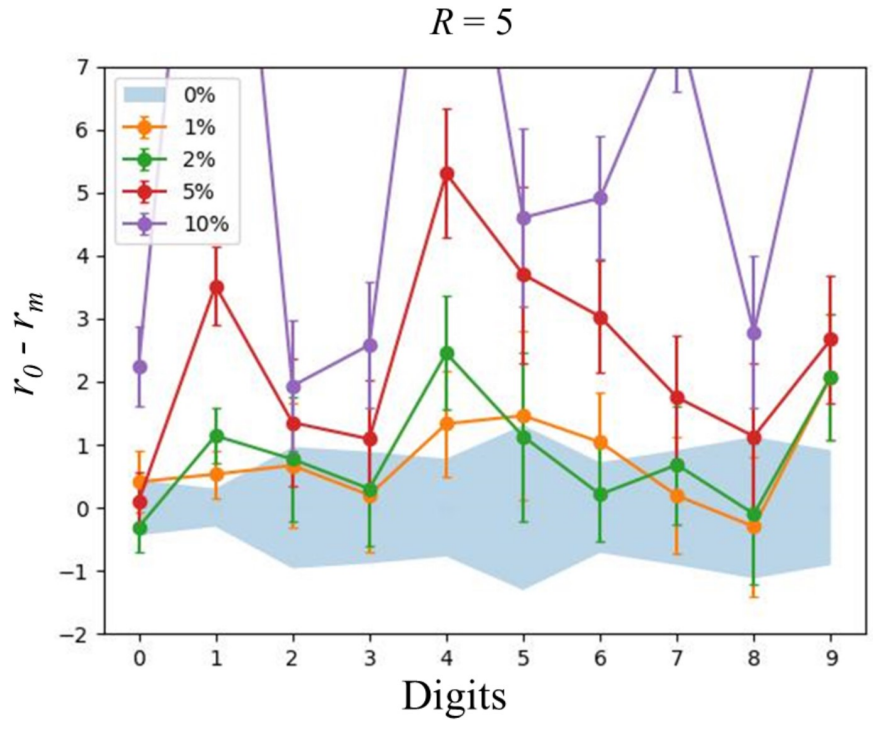


Figure 3.30: Difference of Acc_n with a training dataset with $R = 5\%$ between values for $r = 0\%$ and the values for $r = 1\%$, 2%, 5% and 10%.

***R* = 10%**

In Fig.3.31, curves representing Acc_n for a ratio of noise injected in the training set *R* = 10% are plotted, while the noise implemented into the testing set varies with *r* = 0%, 1%, 2%, 5% and 10%. Comparing it to the results obtained with *R* = 5%, the MR has decreased by more than 1% for the lowest amount of noise in the testing set (*r* = 0%, 1% and 2%), while the standard deviation has once again increased. For *r* = 5%, the MR decreased slightly, by less than 1%, and the standard deviation also increased. Meanwhile, we observe that the MR for *r* = 10% has slightly improved by less than a percent, while the standard deviation increased. We traded accuracy for homogeneity in the results by increasing the noise ratio *R* from 5% to 10%.

The observation that the best results are reached with $r = R$ is verified with the MR but not with the standard deviation with the condition *R* = 10%.

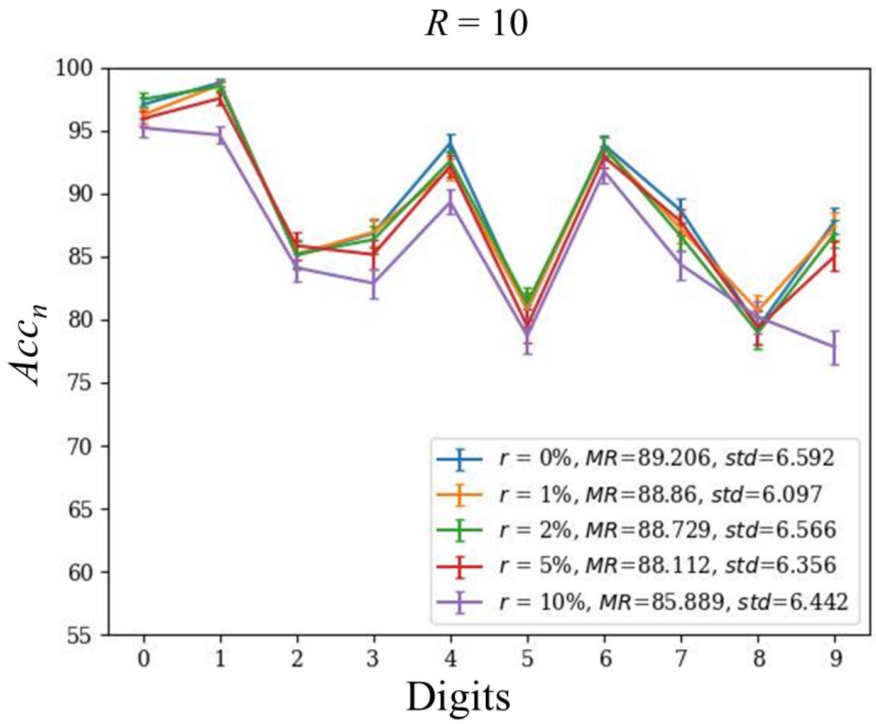


Figure 3.31: Acc_n for all the digits, with a network trained with a training dataset with a ratio R=10% of switched pixels. The network is tested with the testing dataset with a rate of 0%, 1%, 2%, 5% and 10% of noise injected.

In Fig.3.32, the differences of the Acc_n between the curve with the highest MR in Fig.3.29 (standing for *r* = 0%) and the various other curves are displayed. Nineteen out of the twenty Acc_n values from *r* = 1% and 2% have uncertainties matching with *r* = 0%, which

is the best result observed until now. Besides, three Acc_n values from the curve for $r = 5\%$ do not match the uncertainty with $r = 0\%$, which is the lowest number reached. While the result for the curve representing $r = 10\%$ has reached its closest performance with three Acc_n values matching uncertainties, the curve is overall following the pattern of the other values of r. Indeed, nine out of the ten points are below the 5% of proximity.

In the same way it has been done for $R = 2\%$ and $R = 5\%$, the plots showing the Acc_n for a ratio of noise $R = 10\%$ with a noisy dataset, the behaviour of the curves shown by the difference between the curves of such a plot, the difference between the two methods of noise injection of a given ratio $R = 10\%$ and the behaviour given by the difference between the curves are all given in Appendix A for better visibility, along with the explanations.

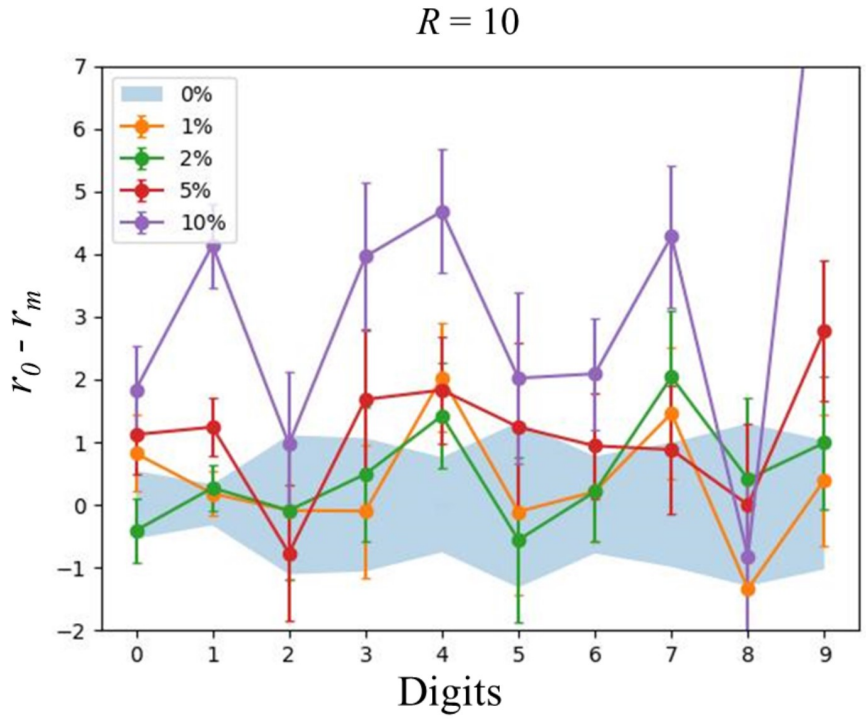


Figure 3.32: Difference of Acc_n with a training dataset with $R = 10\%$ between values for $r = 0\%$ and the values for $r = 1\%$, 2%, 5% and 10%.

Conclusions

While, for a network trained without noise, we observe the best predictions results when no noise is injected into the testing images, we also notice the results worsening as the rate of noise injected increases, regarding both the MR and the standard deviation.

Meanwhile, what is observed from the plots introduced in this section is the fact that the predictions resulting for a given *r* generally reach a peak of performance for $r = R$. However, this feature is not observed in certain cases, such as for $R = 5\%$, for example. In order to globally understand the behaviour of the network regarding the injection of noise in the training and testing sets, we plot the MR and the mean of the standard deviation as a function of *R* for a given *r*. The graphics displaying it with the MR is shown in Fig.3.33, and Fig.3.34 presents the one with the standard deviation.

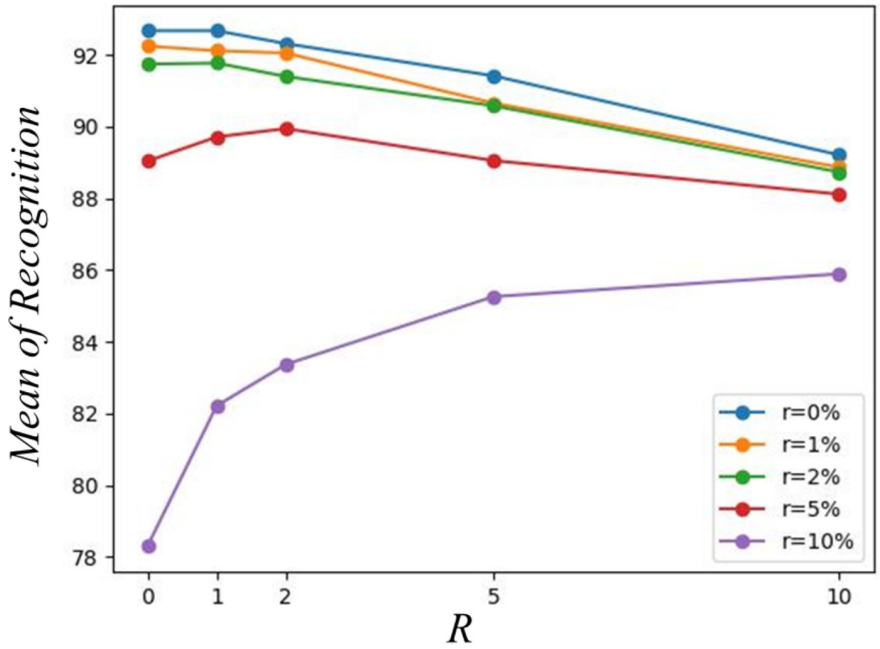


Figure 3.33: Mean of recognition for all *R* at given *r*.

One can observe from the plot in Fig.3.33 that the curves tend to converge towards a single point, with different starting positions and convergence factors. Regarding the behaviour of the curves, the means of recognition differentiated by the noise *r* could be approximated as overdamped oscillators with different factors towards a convergent decreasing linear function [74]. This approximation could be strengthened with a deeper study over it.

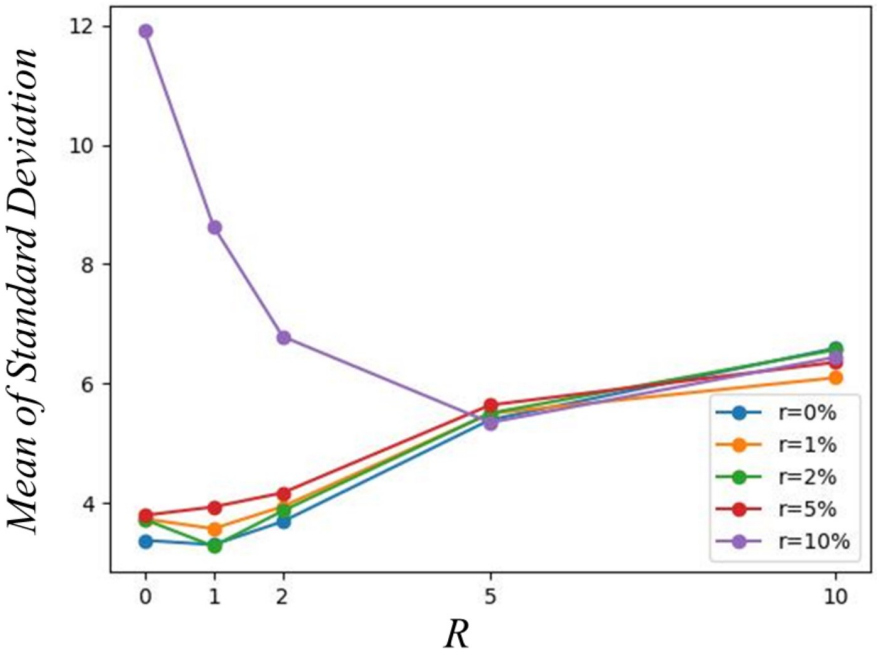


Figure 3.34: Standard deviation for all *R* at given *r*.

The curves plotted in Fig.3.34 tend to verify the approximation of the curves by overdamped oscillator models [74], as they converge towards a non-linear function, with a stronger factor, but different parameters. Again, this approximation could be the subject of a deeper study.

Even though the standard deviations aggregate for $R \geq 5\%$, we observe from both previous plots that the ratio *R* is of no consequence in the ranking of the robustness regarding *r*. The smaller is the noise injected in the tested images *r*, the more accurate are the results. Besides, the best results with a high rate of noise $r = 10\%$ are obtained when the network is also trained with the same high ratio $R = 10\%$, for the upcoming studies, the ratio of noise implemented in both training and testing set will be set at 10%.

4 Optimisation of neural networks

In Sec.3.4, we observed the impact of injecting noise into the images provided on the performances of the network. Since particles can affect electronics, and potentially influence network performance, as seen in Sec.2.1, it is essential to first examine the direct consequences of particle interactions with electronics on network performances. Subsequently, efforts must be made to optimise these small networks to enhance performance.

4.1 Experiments at Legnaro facility

4.1.1 The CN accelerator

For inducting primary tests with the Zybo board, irradiation with particles have been conducted at the accelerator Van de Graaff accelerator CN in Laboratori Nazionali di Legnaro (LNL). It is a vertical accelerator, housed in a tower near the north-east border of the laboratory. About 7 meters tall, at its top is placed the high-voltage terminal (in the past up to about 7 million volts, since 2006 up to 5.5 million volts, operational on weekdays mainly during the day), supported by a column (observed in Fig.4.1) and, inside it, crossed by the accelerating tube, along which the electrostatic voltage is uniformly distributed, from the 5.5 MV of the head down to ground potential (0 V) at ground level.

The whole setup is contained in a large metal container (the 'tank'), filled with a mixture of gas (N_2+SF_6) with insulating properties at a pressure of 12-14 bar. Inside the accelerator head (called the 'high-voltage terminal'), there is space for the particle source (positive ions of materials such as H, He, ...) and the equipment that feeds it (voltage generators, power supplies, equipment to maintain the beam channel in a vacuum).

While the particles are inside the terminal, they are protected by its Faraday cage and do not feel its voltage.

Inside the 7 MV terminal, with the beam source,are the pre-acceleration system, and all the instrumentation connected to them. However, as soon as they are conducted downward by deflection and electrostatic focusing systems and exit the terminal zone, they feel the electric field distributed along the tube and are accelerated up to a final energy (at the tube exit) equal to E=qV, where q is the charge state of the ion (i.e., it is a number representing how many electron charges the ion is away from neutrality) and V is the voltage between the terminal and ground.

At this point, all that remains is to conduct the accelerated particles towards the target and measurement point using deflectors and magnetostatic lenses. With the CN, beams of protons, deuterium, and helium (single or double charge) are accelerated for studies in fundamental and applied physics. The main application fields are: materials science, radiobiology, radiation-matter interaction, radiation damage, and dosimetry. For fundamental physics, for example, cross-sections and/or excitation functions for nuclear reaction channels that are still poorly investigated are studied, as well as neutron/gamma spectrometry. Moreover, it is possible to produce, from a specially equipped and appropriately shielded target station, beams of medium-intensity neutrons, capable of exploiting the maximum authorised proton current (3μA) [75].

Figure 4.1: Column of the CN accelerator being placed inside the tank, picture taken by Andrea Alessio, from [75].

4.1.2 Setup and experience

Neutron beam production involves bombarding a Beryllium target with a deuterium beam. Using a 4 MeV beam with a current of 100 nA on the Beryllium target maintained at 0°C, radiations levels can range from 10^9 to $5 \cdot 10^{11}$ neutrons.cm^{-2}. These values are obtained considering the total Yield.sr^{-1}.μC^{-1} from Fig.4.2, which depicts the total neutron yield at 0°C. The following third-order polynomial is used to represent the data:

$$Y_n(E_d) = 10^9 \cdot (-0.3819 + 0.1740E_d - 0.00137E_d^2 + 0.01131E_d^3), \tag{4.1}$$

where the unit of Y_n is neutrons.sr^{-1}.μC^{-1} and the unit of E_d is in MeV [76]. From this data, we extract 1.02.10^9 neutrons.sr^{-1}.μC^{-1} for a deuteron energy of 4.0 MeV. Taking into account a hardness factor of 1.146 at E_d = 4.0 MeV, a charge of 3.60.10^2 μC per hour, and experimental positions ranging from 35 mm to 200 mm for the aperture of the beam, predictions from LNL suggest fluxes ranging from 10^9 to $5 \cdot 10^{11}$ neutrons.cm^{-2}.

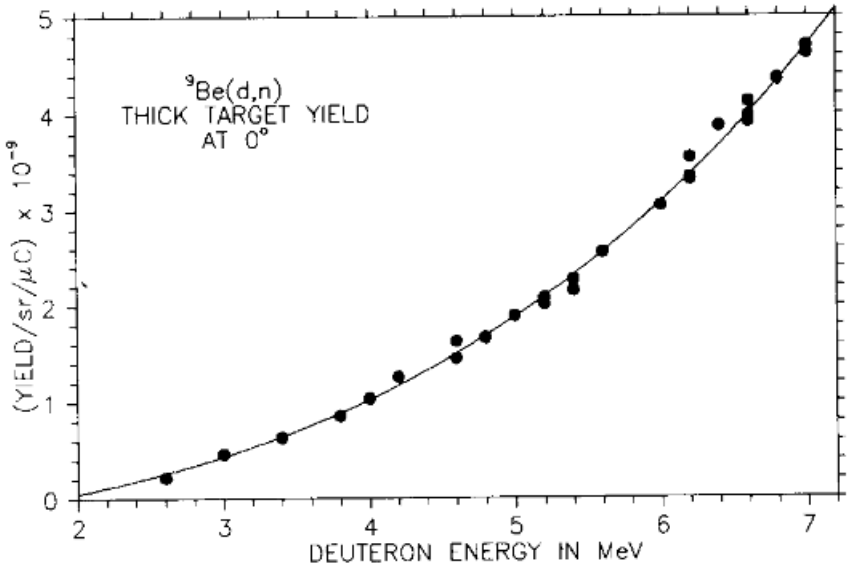


Figure 4.2: The ^{9}Be(d,n) thick-target yield at 0°C, from [76].

The experiment took place from November 21st to 23rd, 2022 at LNL. Over the course of three days, the Zybo board was exposed to the neutron beam, which was previously shielded with 8 cm of plastic to prevent particle interactions with electronics not involved in neural networks computing, as illustrated in Fig.4.3. The board was positioned at various distances from the beam, ranging from 8 to 51.5 cm from the beam, a picture illustrating the setup is displayed in Fig.4.4.

Figure 4.3: Image of the Zybo Z7-10 board shielded by a 8 cm plastic shield. The column in the centre is a hole dug aimed to let the beam interact with the electronics involved in the computations related to the network.

The experimental procedure involved training a neural network with the same properties as described in Sec.3.2 on a separate machine (training it on the Zybo board or another machine has given the exact same results; for speed constraints, the training is done on machines with more efficient resources). The trained network was then imported onto the Zybo Z7-10. The FPGA was tasked with testing the network using a testing set, while being irradiated by the neutron beam. Subsequently, comparisons between the results obtained under irradiation and those obtained in a non-radiation environment were performed.

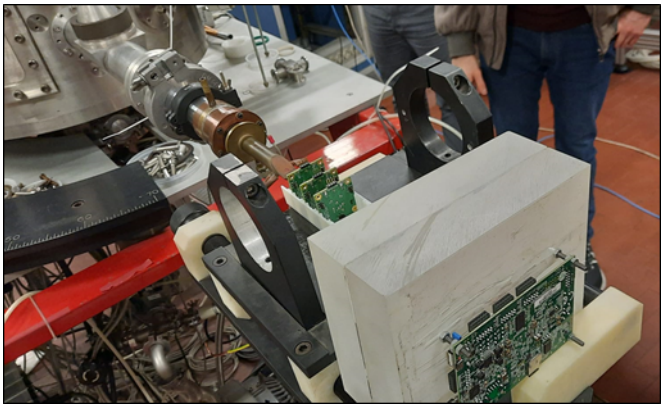


Figure 4.4: Image of the Zybo Z7-10 board shielded by a 8 cm plastic shield. The column in the centre is a hole dug aimed to let the beam interact with the electronics involved in the computations related to the network.

For positions within a distance of 30 cm from the beam, the board crashed prematurely, rendering it unable to handle any communication, thus no testing tasks were manageable. Between 30 and 35 cm from the beam, differences were observed between tests conducted under irradiation and those conducted without. However, no differences were found above 35 cm. To illustrate the observed differences, we present the data in a format similar to Tab.3.1 and compute Tab.4.1.

Prevision / **Input**	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	+2	0	0	-1	0	+1	0	0	-2	0
9	0	0	0	0	-1	-1	0	+2	0	0

Table 4.1: Table gathering all the differences between the test made under radiations and the predictions made out of radiations.

Despite the small discrepancy between the two predictions, the neutron beam, even if composed of uncharged particles, interacts with electronics and affects network performance. This observation is directly related to modifications in the output layer values. Indeed, since the network predicts the digit corresponding to the highest value among the ten output values during a forward propagation, predictions are altered if these values are modified. Approximately 5.9% of output vectors were modified, primarily for digits 8 and 9. While most modifications did not affect predictions, some resulted in changes to the final results, as shown in Tab.4.1. The largest modification recorded reached 5% in the output layer. Thus, a greater discrepancy between the correct value and the highest remaining value indicates greater robustness in the network. This will be further discussed in Section 4.2.1.

4.2 Characterisations of a network

4.2.1 Confidence level

In order to gauge the robustness of the network, the MR cannot be the sole parameter considered. As witnessed in the Legnaro experiment, the 10-value array output undergoes modifications when the device is exposed to neutrons. To understand the significance of this outcome, it is essential to first explain how it is obtained.

In a supervised learning, after the network is trained, weights and biases are fixed. When an input is provided for testing, all the values are processed by the trained network, following 3.1 until reaching the output layer. The output layer, being of the same dimension as the number of classes (10 in the case of MNIST), each index is associated with a class. The highest value in the final vector corresponds to the index of the guess made by the network. For clarification, an illustrated example is pictured in Fig. 4.5

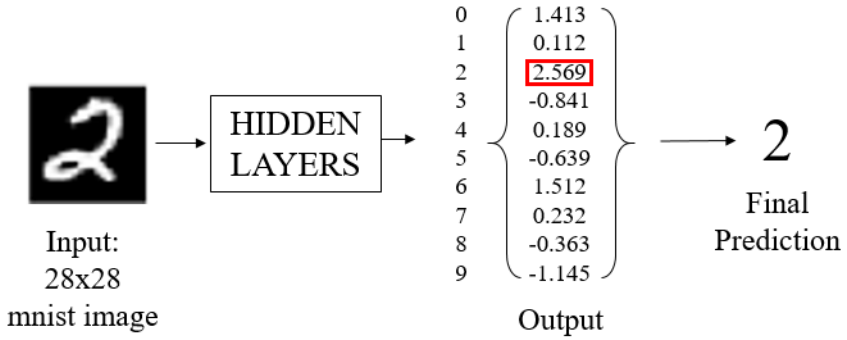


Figure 4.5: This scheme describes how a network predicts the class corresponding to the input provided. A picture of a 2-digit is provided and then processed through the hidden layers.The output array, where the highest value is chosen as the final prediction, is obtained out of the processing. In this case, the highest value is 2.569 at index 2, corresponding to the class "2". The network provides a correct answer for this input.

Thanks to the analysed results from the Legnaro experiment, we have witnessed damages induced by particles hitting the chip repercuss into the output arrays through the values. Examples of the consequences of such impacts can be observed in Table 4.2, where the perturbations modify an initially incorrect prediction into a correct one in one case, and conversely in another by changing the highest value of each output array. The final prediction has been modified by the radiation.

The robustness of the network is directly related to the confidence of the network before any interaction with neutrons. The higher the value of the correct prediction regarding the highest one among the remaining corresponding digits, the more robust it becomes.

	Image of an "9" number 4673			Image of an "8" number 2152		
Digits	Array	Array with chip under radiations	Difference	Array	Array with chip under radiations	Difference
0	0.25	0.23	-0.02	0.32	0.33	+0.01
1	0.27	0.29	+0.02	0.26	0.25	-0.01
2	0.27	0.25	-0.02	0.36	0.35	-0.01
3	0.20	0.20	0	0.47	0.46	-0.01
4	0.43	0.45	+0.02	0.24	0.25	+0.01
5	0.31	0.33	+0.02	0.36	0.37	+0.01
6	0.33	0.31	-0.02	0.22	0.21	-0.01
7	0.35	0.35	0	0.26	0.27	+0.01
8	0.37	0.37	0	0.46	0.47	+0.01
9	0.46	0.44	-0.02	0.31	0.32	+0.01

Table 4.2: Difference between the digit-normalised output arrays when the chip is exposed to radiation and when it is not. These results are obtained with the 4673$^{\text{th}}$ for an image representing a "9" and the 2152$^{\text{th}}$ for an image representing a "8" from the MNIST test set. In the case of image number 2152, the network initially predicted incorrectly the number "3" before irradiation and "8" during radiation exposure. Reciprocally, the network's prediction was correct for image 4673 with the prediction being for "9" and became incorrect and predicted a "3" due to radiation. The green lines represent the index of the digit predicted during the irradiation, while the red ones represent the prediction before the irradiation.

Oppositely, in the case where the network provides an incorrect prediction, the closer the value for the correct digit is to the highest one, the greater the likelihood that damages will lead the network to produce the correct prediction.

In order to quantify this phenomenon, we will introduce a new parameter called Confidence Level (CL), a metric more relevant than MR in our situation.

To do so, we start by working separately on digits for the same reasons as discussed in Sec3.4. We aim to have a perfect comprehension on the work on each digit and prefer independence between them. Each of the ten classes representing the MNIST digits, have a label that we express with the index $i \in \{0, 1, ..., 9\}$. We call M_i the total amount of images in the testing set for an i-digit. For each provided j-image in M_i, to be tested, an output layer vector $\mathbb{V}_{ij}$ is provided. We can define the set of its ten components as

$$\mathbb{V}_{ij} = \{V_{ij}^{\ell},\ \ell = 0, \cdots, 9\}\ ,$$

only the highest V_{ij}^{ℓ} is associated with the prediction ℓ.

We introduce the parameter Ω_i such as:

$$\Omega_i = \bigcup_{j \in M_i} \mathbb{V}_{ij}, \tag{4.2}$$

where Ω_i is a set gathering all the values of $\mathbb{V}_{ij}$ for a given *i*. Since the values in $\mathbb{V}_{ij}$ result from Eq.3.1, $min(\Omega_i)$ and $max(\Omega_i)$ are obtained empirically. Examples of arrays $\mathbb{V}_{ij}$ before any normalisation are displayed in Table 4.3.

Digits	0_7699	1_9556	4_3133
0	0.97488785	-1.5613939	-1.5613939
1	-1.8512547	0.10530555	-1.199068
2	-1.1266026	-0.47441596	-1.4889287
3	-1.0541375	-1.7063242	-1.2715331
4	-1.5613939	-1.199068	-0.6193464
5	-1.1266026	-1.3439983	-1.4889287
6	-1.1266026	-1.3439983	-1.0541375
7	-1.3439983	-1.1266026	-1.4164635
8	-1.1266026	-0.32948554	-0.76427674
9	-1.2715331	-1.7787894	0.10530555

Table 4.3: Table displaying examples of output vectors $\mathbb{V}_{ij}$ generated by the network. The highest value in each array is selected as the predicted value according to the network. For the first and second images, the predictions are correct because the highest values are respectively 0.97488785 and 0.10530555 and corresponding to the indices of the correct digit. However, the prediction in the last example is incorrect because the highest value, 0.10530555, corresponds to the digit 9 and not 4.

The values in Table 4.3 are the results of successive operations based on the one introduced in 3.1. As a result, negative and values close to 1 are obtained.

The next step in building the CL is to normalise the values while maintaining independence between each digit. Thus, for each i, we create a function f_i that normalises over the smallest and highest value in Ω_i:

$$f_i(x) : \mathbb{R} \to \mathbb{R}_{\geq 0} \quad f_i(x) = \frac{x - min(\Omega_i)}{max(\Omega_i) - min(\Omega_i)}. \tag{4.3}$$

.

Now, to quantify the effects of the radiations on the results, we need to distinguish between two cases:

- First, when the network makes a correct prediction but radiation makes it incorrect by switching the highest value of the vector, represented with image 4673 in Table 4.2.

The value more likely to be chosen after modification is the one closest to the correct value before any perturbation. Since the alteration is not constant, we attribute more importance to greater differences between the value of the correct digit and the second highest one, making the network more robust.

- The second case is when the network makes an incorrect prediction but radiation amends it, represented with image 2152 in Table 4.2. Unlike the previous situation, the smaller the difference between the higher value and the correct number corresponding value, the higher the chances are to correct the prediction.

To summarise, in both the cases, the difference between the value of the correct digit and the highest among the remaining ones should be as large as possible to enhance the network's robustness.

When the mean is calculated over the total number of images for each digit, it can be formulated as the Confidence Level for a digit *i*:

$$\mathrm{CL_i} = \frac{1}{M_i} \sum_{j=1}^{M_i} f_i(V_{ij}^i) - f_i(max_{\ell \neq i}(\mathbb{V}_{ij})), \tag{4.4}$$

. where Len_i is the number of images in the testing set for the digit *i*.

The mean of 10 CL_i can be calculated to obtain the CL:

$$\mathrm{CL} = \frac{1}{10} \sum_{i=0}^{9} \frac{1}{M_i} \sum_{j=1}^{M_i} f_i(V_{ij}^i) - f_i(max_{\ell \neq i}(\mathbb{V}_{ij})) \tag{4.5}$$

. expressing the robustness of the network to the perturbations induced by cosmic rays. Examples of CL_i and the overall CL are visualised in Table 4.4. It's notable that CL_5 is the lowest Confidence Level at 0% noise and is close to be the lowest at 10%. This is because the MR is the smallest for this digit in both implementations of noise. However, we notice CL_4 is the smallest for 10% noise despite having a relatively high MR. Most predictions for digit 4 are close to the second-highest values and could be influenced by radiation, leading to incorrect predictions.

The CL is a very strong metric that will lead us in the search for optimisation of the architecture of the network in terms of robustness to perturbations.

4.2.2 Weights visualisation

In Sec.3.1, we introduced the concept of weights in neural networks and discussed their function. Motivated by the desire to optimise finite-dimensional neural networks, along

Digits	CL_i for 0% noise	CL_i for 10% noise
0	0.4759615	0.4443430
1	0.5897323	0.4044362
2	0.3962992	0.2990106
3	0.4146421	0.3143130
4	0.4204332	0.2110623
5	0.3600811	0.2192392
6	0.4936567	0.3581108
7	0.4069328	0.3039969
8	0.3999368	0.2685075
9	0.4103263	0.2568289
Total	0.4392423	0.3099597

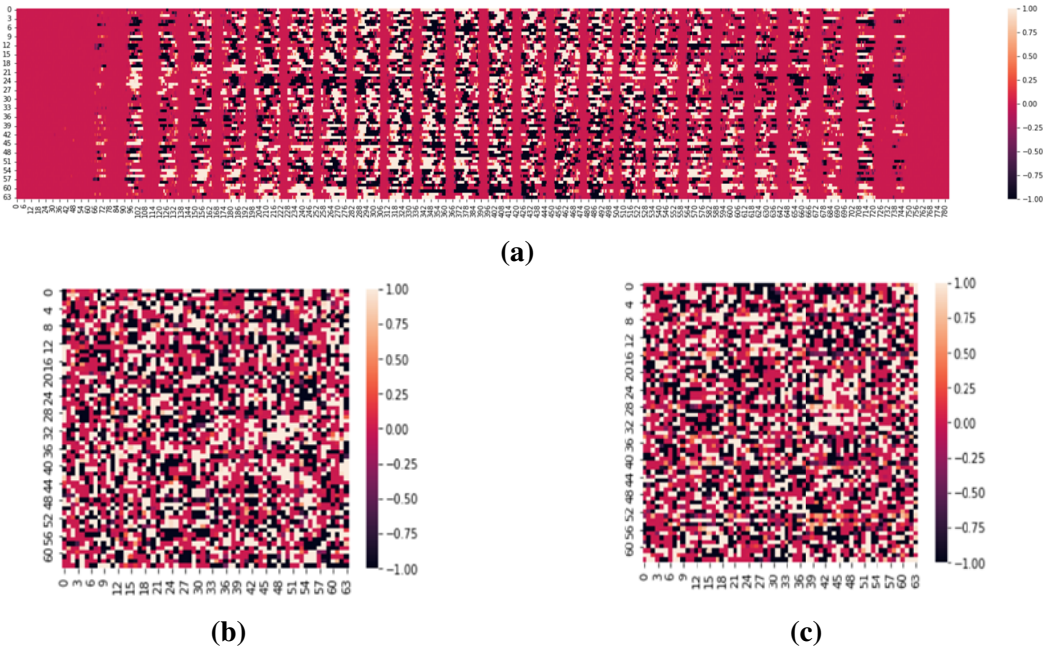
Table 4.4: List of the CL_i for the network at respectively 0% and 10% noise in the testing and training sessions. The Total line corresponds to the CL. We denote all the values are higher with smaller amount of noise injection

with works such as the one presented in [77], it becomes necessary to visualise the weights comprising the trained network. Each neuron in a layer is connected to all neurons in the following and layer, with each connection representing a weight. Therefore, the number of weights between two layers is given by:

$$N_{W_{ij}} = N_i \cdot N_j, \tag{4.6}$$

where N_k represents the number of neurons in layer *k*. As demonstrated in [77], one way to assess the status of weights in a network is by plotting the weights between each layer in a 2D diagram. We have done so, and the resulting plots are depicted in Fig.4.6. While Figures 4.6b and 4.6c do not exhibit any particular pattern or distinct regions of weights, Figure 4.6a reveals a repetitive behaviour.

To further investigate this pattern, we plot the values of the weight for a single row of Fig.4.6a onto a 28x28 pixels 2D plot, as shown in Fig.4.7. The 2D plotting follows the same process as reverting the flattened image to its original shape. Specifically, we analyse the weights for the first and fifth neurons of the first hidden layer. It becomes evident that each neuron exhibits minimal activity in areas corresponding to the positions of black pixels in the MNIST dataset. Conversely, regions where gray and white pixels appear in the training images display more pronounced weight values. Neurons located in the centre circle of 28 pixels in diameter demonstrate significantly higher activity, with absolute values at their peak. Notably, neurons with values of -1 and +1 are closely intertwined, suggesting that further investigation into the variance of weights would be pertinent.



(a) **(b)** **(c)**

Figure 4.6: 2D visualisation of the weights between **a** the input layer (columns) and the first hidden layer (rows), **b** the first (columns) and the second (rows) input layers, and **c** the second (columns) and third (rows) hidden layers.

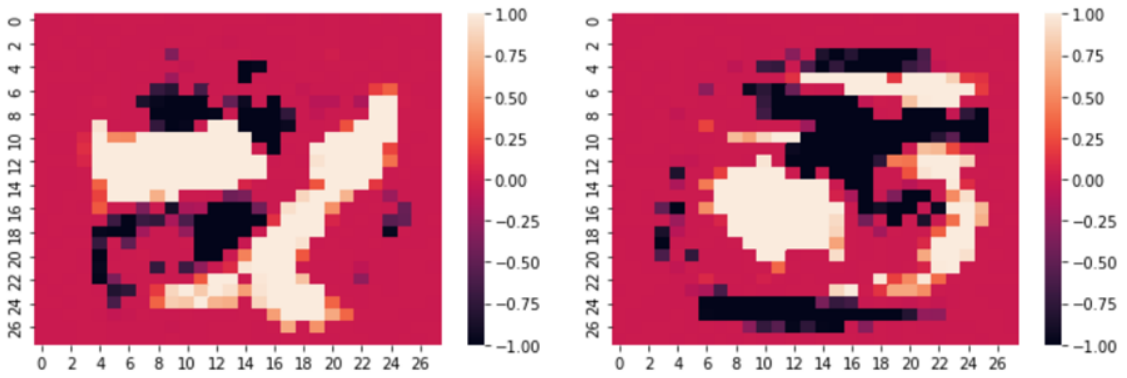
4.2.3 Inverse temperature

Besides having metrics characterising the performance of the network, we could use theoretical parameters to help us optimise the network. In Sec.4.2.2, we discussed the interest in studying the variance of weights between the hidden layers, and to do so, we utilise the inverse temperature.

For the present case, we consider supervised learning focused on classification. The network we will investigate is made up of five components: input, output and three hidden layers. The input and output layers are fixed, regarding the numbers of neurons composing them. This is because the size of the samples inside the training set and the classification task are strong constraints and cannot be modified. The hidden part of the network is considered a Deep Boltzmann Machine (DBM) and can be described by the following Hamiltonian [78]:

$$H(\sigma) = -\sqrt{\frac{2}{N}} \sum_{p}^{K-1} \beta^{(p)} \sum_{(i,j)\in L_p \times L_{p+1}} J_{ij}^{(p)} \sigma_i \sigma_j \,. \tag{4.7}$$

Each σ_i represents one neuron of the p^{th}layer L_p while, in the approximation of centred Gaussian learned weights, we can define $W_{ij}^{(p)} = \sqrt{\frac{2}{N}}\,\beta^{(p)} J_{ij}^{(p)}$ as the connections obtained after the training procedure. In particular, each pair of layers is connected by a weight matrix $\boldsymbol{W}$, where entries $W_{ij}^{(p)}$ are associated with an inverse temperature $\beta^{(p)}$.



(a) **(b)**

Figure 4.7: 2D visualisation of the weights between the input layer and the first hidden layer for **a** the first and **b** fifth neurons of the first hidden layer.

An estimate of the temperature $\beta^{(p)}$ can be added in order to help its study and its integration into the relevant parameters of the network. It turns out that the variance of the weight matrix of each pair of layers is linked to the inverse temperature [79], [80]. We therefore evaluate the inverse temperature of a couple of layers as:

$$W_2 = \frac{1}{N_p N_{p+1}} \sum_{i=1}^{N_p} \sum_{j=1}^{N_{p+1}} (W_{ij}^2 - \bar{W}) \sim \beta_p^2 \,. \tag{4.8}$$

where $\bar{W}$ stands for the mean value of the weights W_{ij} and N_p describes the number of neurons on the p^{th} layer.

In our analysis, we denote L_p as the $p-$th layer that composes the network, among the total number $N = 3$, and we consider two pairs of them. The first pair (L_1,L_2) is associated to the inverse temperature β_1, and the second pair (L_2,L_3) is associated to the inverse temperature β_2. For example, a training process ended up with having the couple of values (β_1,β_2) being (8.438,8.301). We refer to the region associated with β_1 as the cold area (due to the higher value of the inverse temperature), while the region associated with β_2 is referred to as the hot area.

4.2.4 Spectral radius

Together with such a temperature parameter, another important thermodynamic quantity can be introduced. Now that each pair of layers has an associated an inverse temperature β_p,

one can build the following matrix:

$$\begin{pmatrix} 0 & \alpha_2\beta_1^2 & 0 \\ \alpha_1\beta_1^2 & 0 & \alpha_3\beta_2^2 \\ 0 & \alpha_2\beta_2^2 & 0 \end{pmatrix},$$

where the quantities $\alpha_p = N_p/N$ correspond to the form factors of the elements of the network: the relative size of the number of neurons in one layer (N_p) with respect to the total number of neurons in the network (N). The spectral radius (SR) ($\rho(\boldsymbol{\beta}, \boldsymbol{\alpha})$) of this matrix turns out to be a function of the whole set of inverse temperatures $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_p, \cdots)$ and the form factors $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_p, \cdots)$. When $\rho(\boldsymbol{\beta}, \boldsymbol{\alpha}) > 1$, a phase where the network correctly classifies is possible.

4.3 Modification of a network's topology

4.3.1 Movement conjecture

As introduced in Sec.3.2, due to the limited processing capacities provided by the hardware, the computational power of the model must be constrained to a small amount. This computational power is defined as the number of neurons (N) available to fill each layer of a Deep neural network.

Despite its small dimension, this network can achieve satisfactory results regarding accuracy and computation speed. Since our goal is to achieve the highest accuracy, we not only optimise the parameters (as this greatly depends on the datasets used) but also perform architecture optimisation, as presented in [81]. To guide us toward the most accurate topology, we use MR, CL, SR and the inverse temperature.

With reference to Fig.3.4, our aim is to find the correct number of neurons that enter each one of the layers, compatible with the task that has to be performed. Then, with $\alpha_p = N_p/N$, we indicate the fraction of neurons populating the L_p layer, while $N = \sum_p N_p$ is the total amount of neurons we can move. In the infinite computational power limit ($N \to \infty$), it is possible to show [78] that moving the neurons inside the couple of coldest layers L_{p^*} and L_{p^*+1}, maintaining their form factors of the same amount $\alpha_{p^*} = \alpha_{p^*+1} = 1/2$, one can avoid problematic working regimes (Annealed region).

Taking inspiration from this result, we try to apply a variation of the previous rule, adapting it to the finite dimensional case of our classification problem. We are going to prove that we can produce an improvement of the classification performance, if we choose to move the neurons according to the following

Conjecture 1 *Denote with*

$$\beta_p^* = \max_{p\in\{1,2,\dots,K-1\}} \beta_p$$

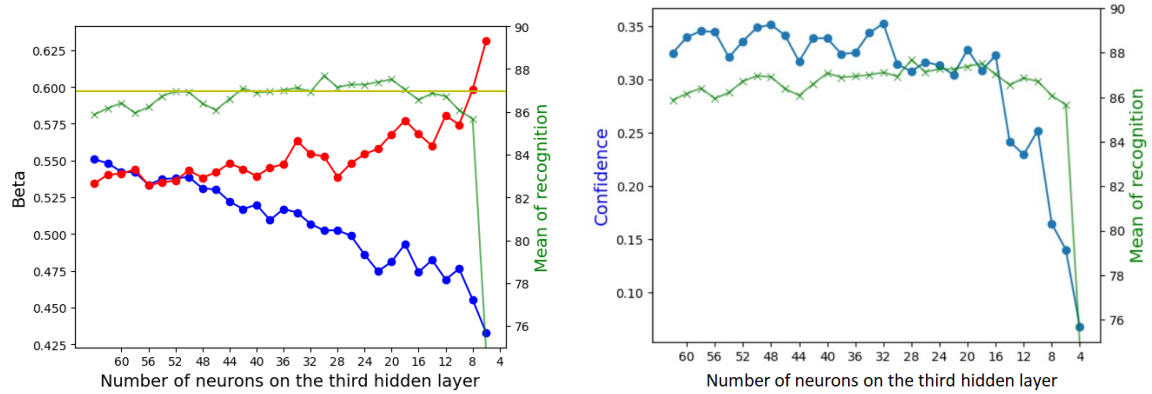
the maximum of the inverse temperature between each couple of layers. Then if the neurons are moved in the layers L_{p^*} and L_{p^*+1} , such that

$$\lambda_{p^*} = \lambda_{p^*+1}$$

i.e. the width of the coldest couple of layers is of the same amount, then we assist to an improvement of the classification metrics, MR (Eq.3.8) and CL (Eq.4.5).

4.3.2 Hot to cold shift

As introduced in 4.2.3, the pair of inverse temperatures for the network (β_1,β_2) is (8.438,8.301). Since ($\beta_1 > \beta_2$) and the conjecture suggests moving neurons from the hotter area toward the colder one, we redistribute two neurons from the last hidden layer to the first two layers until no more neurons can be distributed as depicted in Fig.4.9, ensuring that $\alpha_1 = \alpha_2$. We also compute MR and CL, the raw performance metrics for each of the configurations, and plot them in Fig.4.8.



(a) **(b)**

Figure 4.8: Plots showing the raw performance metrics characterising the network, with the inverse temperature when a movement of neurons is performed on the network. **a** shows MR and inverse temperature function of the number of neurons on the last hidden layer. The yellow line stands for the local maximum of MR around the thermal equilibrium. **b** shows MR and CL for the same configurations.

We notice that the temperatures in the two different regions converge toward a thermal equilibrium and then diverge. Due to the finite size of the problem, we do not reach the condition $\beta_1 = \beta_2$. MR in the meantime tends to slowly increase from 86% to 87% until

a drop when the number of neurons on the last hidden layer coincides with the number of neurons on the output layer, probably due to a bottleneck effect. We also notice CL begins fluctuating between 0.30 and 0.36 before dropping and fluctuates around 0.30 afterward, and finally falls the same way MR did. We remark both parameters have a local maximum matching with the thermal equilibrium and the beginning of the divergence of the inverse temperatures.

The quantitative evaluation of the thermal equilibrium is computed by leveraging the various samples that determine the uncertainties of the metrics. We compare the relative distance $\Delta = \frac{\beta_1 - \beta_2}{(\beta_1 + \beta_1)/2}$ of the temperatures of the hidden layers, with their uncertainties under repetitive training. If Δ is less than the considered uncertainty, we assume $\beta_1 = \beta_2$.

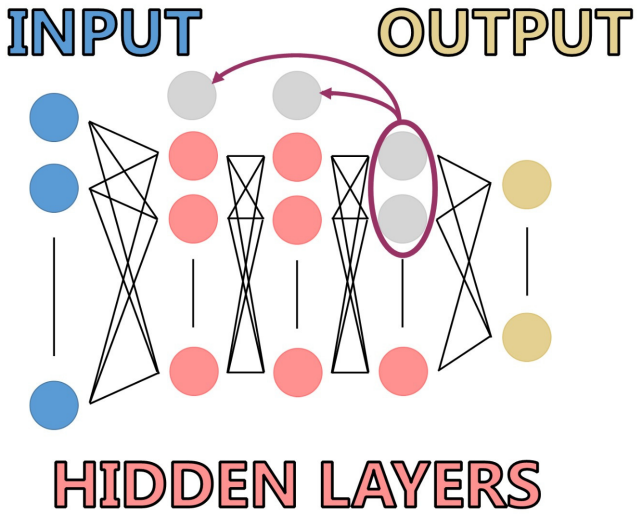
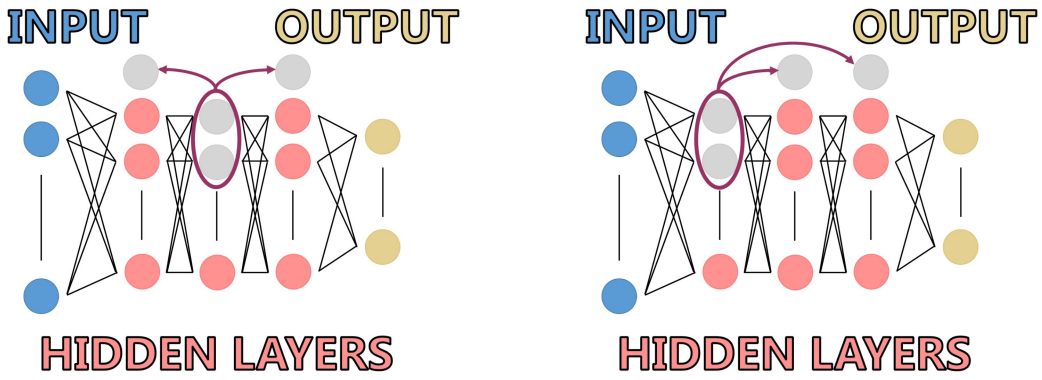


Figure 4.9: Scheme describing the movement of neurons performed with the network architecture.

4.3.3 Different movements

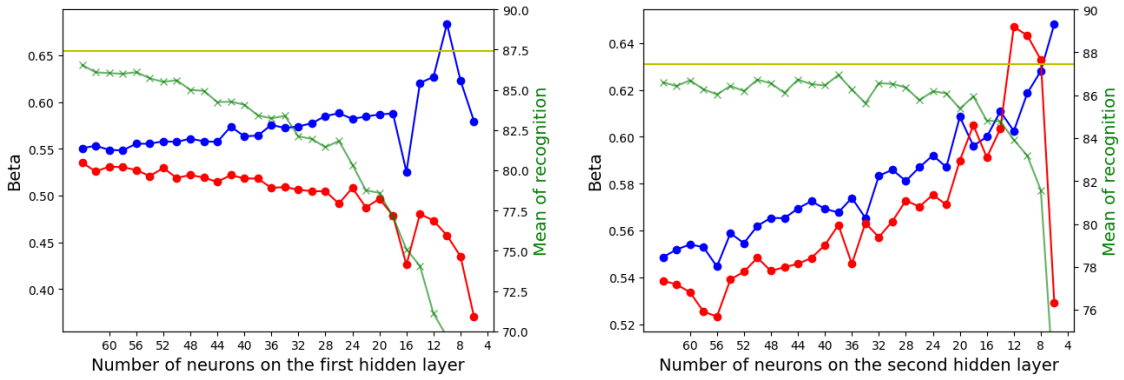
Conjecture 1 stipulates the classification performance should improve if neurons are moved from the hotter area towards the colder one. This conjecture tends to be verified for MR and partially for CL, since some regions of fluctuations improve it. To verify if another movement of neurons could give us better results, which would not fit with the conjecture, we move neurons from the first hidden layer to the two last ones, corresponding to a movement of neurons from the colder area toward the hotter one, as observed in Fig.4.10a. Additionally, we move neurons from the middle hidden layer towards the first and third ones, as observed in Fig.4.10b. The results of these movements are plotted in Fig.4.11, and we notice that shifting the neurons from the colder area toward the hotter one worsens the results, while reducing the number of neurons in the middle layer keeps the results relatively constant

with fluctuations, and then drops due to the bottleneck effect. The addition of the horizontal (yellow) line introduced in Fig.4.8 shows the difference of MR with respect to the movement proposed by Conjecture 1. If a movement enhanced the performance of the network, it should follow the instructions given by Conjecture 1.



(a) **(b)**

Figure 4.10: Scheme describing the other movements of neurons performed to the network. **a** shows the motion of neurons from the hot area toward the cold area. **b** describes the movement of neurons from the middle hidden layer toward the two outer ones.



(a) **(b)**

Figure 4.11: Plots showing MR and inverse temperatures when a movement of neurons is performed on the network. **a** shows results for movements of neurons from the first hidden layer toward the two other ones. **b** shows results for movements of neurons from the second hidden layer toward the two outer ones. The yellow vertical line is the same as in Fig.4.8a, for comparison with movement from the hot area toward the cold one.

4.3.4 Region of interest

As introduced in Sec.4.3.2, although MR shows a slight improvement, it comes at the expense of reducing the confidence in classification accuracy. By conducting multiple

training runs for each network, we gather statistics that confirm the stability of our results. We consistently find the local maximum around the same network topology, at the thermal equilibrium and the beginning of the divergence of the temperatures, as depicted in Fig.4.12

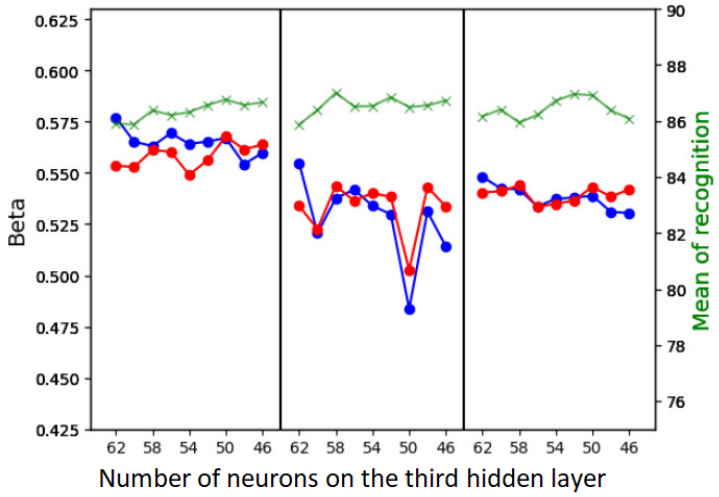


Figure 4.12: Scheme describing the movement of neurons performed with the network architecture.

In Fig.4.12, one can observe this particular region for training with different seed, and the region always present a local maximum of MR. We therefore plot the different results obtained where a mean and a statistic study is performed, and plot it in Fig.4.13, where CL and SR are also displayed. The region presenting the local maximum is observable in the gray area and present $< 1\%$ uncertainties for both CL and MR.

In our situation, this region is optimal because we value more low uncertainty and the general maximum for CL rather than the general maximum of MR observable at 24 neurons on the last hidden layer, with higher uncertainty. Since the particles interacting with the electronic systems affect the CL, it must be prioritised since CL value is in range for being shifted with particle radiations. However, if one looks to increase the raw performance of the network with increasing MR, then going for 24 neurons in the last hidden layer is optimal in this singular situation. A notable result is that CL and SR have very similar behaviour, even if the plot in Fig.4.13 show them at different values.

4.4 Variation of conditions

Since all the previous work has been performed utilising the same characteristics, such as the datasets, the number of classes and the architecture of the code, we want to visualise how the network would react to the modifications of these characteristics.

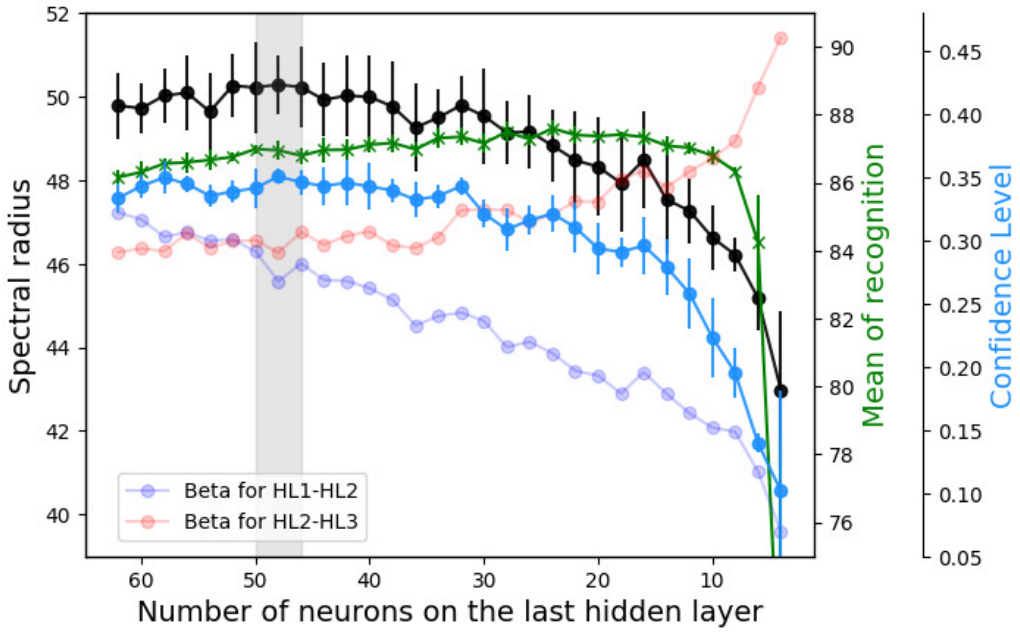
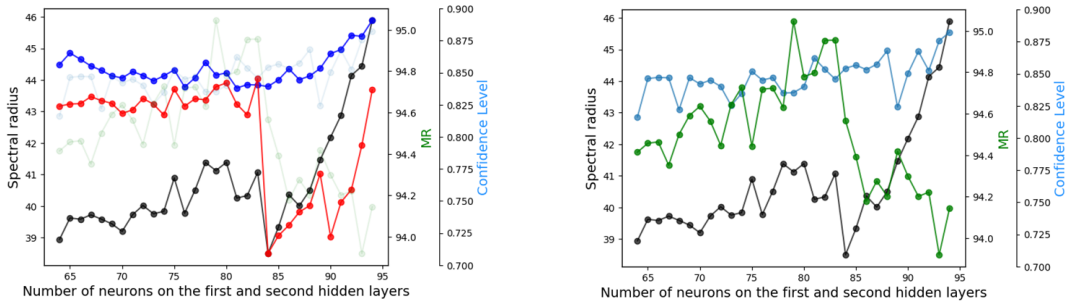


Figure 4.13: Mean and statistics of the different parameters for the movements of neurons from the hot area toward the cold one. On average, the local maximum is around the configuration $(72 \pm 1, 72 \pm 1, 48 \pm 2)$ (in gray).

4.4.1 Reduction of the number of classes

The first modification is to reduce the number of classes of the MNIST dataset. Instead of setting every digit in its own class, we sort them into two classes, being odd/even digits. The output layer of the network is then changed from 10 to 2 neurons. We perform the same exercise done previously by moving neurons from the hotter area toward the colder one, the hot area corresponding to the last two layers. The results can be visualised in Fig.4.14 where a mean of three different trainings is displayed. We observe MR and CL have very high values compared to the results with the standard MNIST, and the peak of performance of MR corresponds to a local maximum of SR. Once again, the interesting region is noticed around the superposition of temperatures, being the peaks of MR and global peaks of CL. In the meantime, SR and CL keep increasing and peak at the very last architecture model, since no bottleneck effect is expected any more because of the low number of neurons in the output layer. We notice MR drops above 84 neurons on each layer of the cold area, and SR has its global minimum at 85. In this case, for particle ambient optimisation, we value more having high levels for MR since CL is high enough for not being influenced by particle injecting variations in the output layer.



(a) **(b)**

Figure 4.14: Plots showing the different parameters of the network for a odd/even classification. **a**, Inverse temperatures and SR for moving neurons from the last hidden layer. **b**, SR, MR and CL for moving neurons from the last hidden layer.

4.4.2 Fashion MNIST dataset

The second modification is made to the dataset, from MNIST to fashion MNIST [82], a dataset with pictures of clothes, being of the same colours and dimensions as standard MNIST. One can notice the total number of images in this set is homogenous among the classes, thus, in this case, Accuracy and MR are the same metrics. The results of the study with this set is displayed in the graphic in Fig.4.15, where the mean of the results of five different studies is displayed. When CL and MR are significantly smaller compared to the results obtained with standard MNIST, we notice again the same interesting area around the superposition of the temperatures for CL and MR with very low statistics, and a general maximum for CL. In the situation of optimising the architecture for particle ambient purposes, we definitely value more the architecture reaching the maximum CL, being of 54 neurons on the last hidden layer, and having the smallest uncertainties. We also notice a plateau for MR, SR and CL between 62 and 28 neurons on the last hidden layer. Once again, CL and SR show similar behaviours.

4.4.3 CIFAR10 dataset

The third modification has been performed on the dataset, the colours and dimensions of the images, and the architecture of the code have been modified. The dataset is CIFAR10 [83], a set of RGB images of dimension 32x32 pixels gathering images of various animals and human means of transport. Since the CIFAR10 set provides this kind of images, the number of input is almost multiplied by 4 compared to MNIST entries, and the number of parameters is drastically increased. Having a fully-connected network is not relevant any more when the number of parameters is this high, since the Zybo Z7-10 must support it.

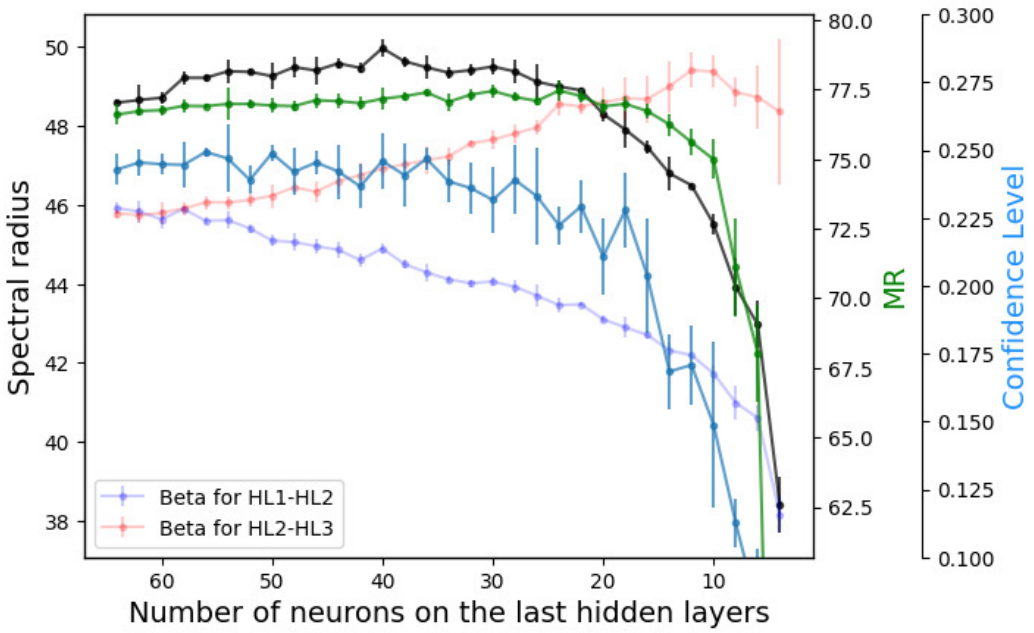


Figure 4.15: Mean and statistics of the different parameters for the movements of neurons from the hot area toward the cold one for the Fashion MNIST.

Thus, we decide to implement a convolutional network based on MobileNet [56] [57] and reproduce the study made previously.

With this configuration, it appears the cold and hot areas are inverted, and implies the cold area is for the couple of layers (L_2,L_3) and the hot one corresponds to the pair (L_1,L_2). Therefore, we move neurons from the first hidden layer toward the two others in order to follow the instructions of Conjecture1. Because of the modification of the model of the implementation of the network, it was necessary to modify the code utilised. We have not been able to obtain data from the output layer, and CL is not computable. We therefore plot the inverse temperatures, SR and MR, in Fig.4.16. We notice MR is very low compared to the other datasets results and to the global CIFAR performance, even if other resources are involved in studies such as [84]. SR is also very low compared to the studies made previously, increasing until reaching 28 neurons on the first hidden layer and then dropping. Finally, we notice MR globally decreases, which is in opposition with the prevision of Conjecture1. Results with CIFAR and CNN do not coincide with the results obtained previously.

4.4.4 Discussion

Inspired by [78], we formulated Conjecture 1 proposing movements of neurons within the architecture to optimise finite-dimensional deep neural networks. It suggests moving

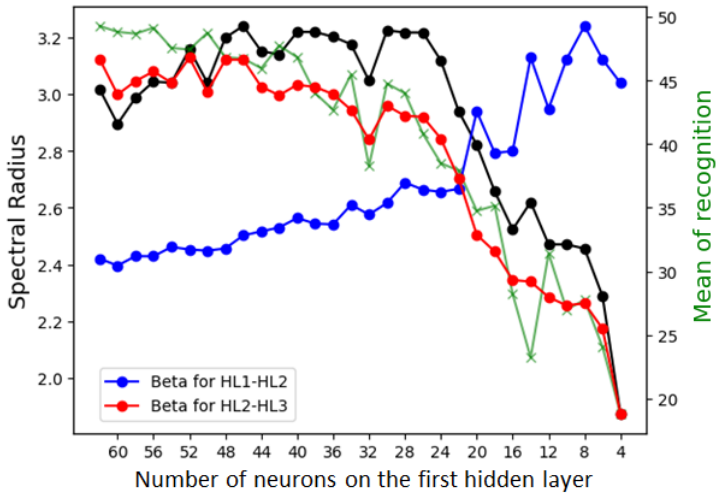


Figure 4.16: Mean and statistics of the different parameters for the movements of neurons from the hot area toward the cold one for the Fashion MNIST.

neurons from the hotter area towards the colder one, while maintaining the condition of homogeneous shifts between the two layers composing the cold area.

We applied these modifications to the network already created and studied its performance modifications through row metrics and theoretical parameters linked to the variance of the weights. We observed the Conjecture is verified for fully-connected networks, under the conditions in which we conducted the tests, primarily focusing on MR, but also considering CL.

Since the purpose of the project is not only to optimise the architecture of finite-dimensional deep neural networks, but also to study their behaviour regarding camera and electronics noise, we identified regions of interest for the architectures optimising the results. Specifically, for low MR and low CL, we prioritise the highest value of CL with low uncertainties, while MR becomes more relevant for high values of CL.

The conjecture holds true in our situations of fully-connected networks, but we observed limitations with more complex datasets coupled with CNNs. Future studies may focus on generalising the conjecture and identifying regions of interest.

Finally, we found that CL and SR exhibit relatively similar behaviours, and a deeper study of the correlations between these two parameters would be of interest for future studies. With a more profound and comprehensive analysis, SR could become a strong candidate for estimating CL, focusing solely on the variance of the weights.

Conclusions

This Ph.D. thesis addressed various aspects of science, including the study of electronics interacting with particles, the effects of particle-induced noise on inputs in the performance of neural networks, and the optimisation of deep networks with finite dimensions.

Studying the effect of particle interaction with electronics is crucial for space missions, as these interactions can lead to significant unwanted consequences. In Chapter 1, we initially investigated the interactions of a microcontroller with particles, focusing on the electronics' response to these interactions. During the tests conducted under the radiation of heavy ions and neutrons, we observed a generally good resistance of the DUTs . Out of the 15 samples tested, 73% did not crash during the process, despite the irradiation fluxes being much higher than those witnessed in LEO, the orbit of interest. Although numerous resets were forced on the electronics due to the interactions, the results, considering LEO characteristics, were satisfactory. However, suggestions for enhancing the reliability of the tests were proposed, such as relocating the tension regulator from inside the DUT to outside.

Thales Alenia Space provided us with a Versal AI Core Series VCK190es1 for studying its resistance to particle interaction, its resource requirements, and its capabilities in utilising artificial intelligence for classification tasks. Since we cannot afford to damage the board during particle tests, it is necessary to switch the project to a cheaper and more easily available option, the Zybo Z7-10.

The Zybo Z7-10, an FPGA, encompasses all the criteria required for the study: the ability to embed neural networks, features electronic components enabling connection with additional components like a video camera for real-time camera processing studies and classification by the networks, and it comes at an affordable price, allowing for risks during radiation experiments.

To embed the neural networks on this FPGA, we introduced the framework composed of PYNQ, FINN, and Brevitas projects, proposed by Xilinx for such tasks. This framework not only facilitates the implementation of the network into the electronics but also enables the mapping of each of the different electronic components forming each network

element. Although not studied in this thesis, this feature allows for a deeper understanding of the consequences of particle interactions on the network at the electronics level.

Future prospects in this area include studying the map of damaged components during irradiations and utilising these results to model particle interactions with electronics. This would enable the study of particle effects with models for better comprehension and preparation before experiments. Further steps would involve having a real-time map of the electronics and visualising particle interactions with the resulting consequences on the electronics. If real-time communication with the electronic components is possible, one could simply mitigate the bit flips caused by charged particles by reversing the undesired effects.

Chapter 3 focused on the study of noise interacting with detectors, such as video cameras, along with its consequences on network performance. Before conducting this study, we introduced neural networks and the dataset used throughout the study of noise injection, the MNIST dataset. We particularly explained the features exploited and studied in this thesis for better understanding of the choices made later on. Performance metrics for characterising a network were defined, and the motivations behind choosing these particular metrics were explained. We also introduced the method of implementing noise into the images provided for training and testing.

We studied different proportions of noise injected into the image, distinguishing between different ratios of noise in the training session or the testing sessions. To study this effect, we examined the Mean of Recognition and the Standard Deviations of the results, focusing on the results obtained with the same ratio of noise in both the training and testing sets. We were interested in the network's performance and the behavior of the curves, identifying similar patterns.

We plotted the final results together and found that the behavior of the network with noise injection not only contradicts the previous conjecture stating that the best results of one proportion of noise are obtained when both ratios are the same for the testing and training sets, but also suggests that the Standard Deviation and Mean of Recognition tend to exhibit an overdamped oscillator towards a non-linear function behavior. This phenomenon requires further study for a better understanding and for developing prediction models.

Finally, the needs of implementing a network into embedded systems induce reduced resources and oblige networks to have small dimensions. In Chapter 4, we introduced the need of optimising the networks while constraining the resources, the dimension of the layers, and thus the computing resources.

Chapter 4 addressed the need for implementing networks into embedded systems, which entails reduced resources and necessitates networks with small dimensions. We

conducted an experiment at LNL with a Zybo Z7-10 board, in which a neural network was implemented. These tests provided an overview of the Zybo's resistance to neutrons and mainly highlighted the issues of particles interacting with a network implemented into the electronics. The results showed that radiation consequences modify the values in the output layer, altering predictions when the network is not "confident enough".

Based on these results, we formulated the "Confidence Level," a raw metric observing the level of confidence of the network. This allows us to quantify the network's robustness to radiation consequences and provides an option for optimisation satisfaction regarding real-life situations.

We also studied the weights composing the networks and their variance, which lead to the construction of a conjecture about architecture optimisation preferences. We opted to move neurons from one hidden layer to the other two layers, as our networks consist of three hidden layers. This movement is based on the inverse temperature of the various areas of the network, and the conjecture suggests moving neurons from hot layers towards cold ones.

We observed that this conjecture is verified in some cases with fully-connected networks, and we also found specific architectures optimising the Confidence Level and the Mean of Recognition. Different architectures are proposed based on individual needs, enhancing one parameter or another, with varying uncertainties.

The Spectral Radius was also defined, using the variance of the weights, and appears to be a good indicator of the Confidence Level. A correlation study between these two parameters must be performed in the future, as it would provide performance suggestions based solely on the weights of one's network.

The focus was on the elements of the network related to the weights, while investigating bias variances could provide even deeper understanding of network performance.

A

In Sec.3.4, besides introducing how we decided to implement the noise in the images to reproduce the interference we will witness during space missions, we also study its effects on the performance of the network. A comprehensive study has been presented, exploiting the results in the case *R*=1% of noise implemented into the training set. Several plots have been shown: the *Acc_n* for all digits $n = 0, \cdots, 9$ with two different ways of implementing the noise during the forward propagation, the difference between the curves with the smaller amount of noise in the testing set *r* for both of the cases presented previously, and the difference between the two approaches while also plotting the subtraction of the curves in this case. For better clarity, not all of these plots have been inserted for all the noise implementations in the training set *R*. This appendix aims to show these plots for $R = 2\%, 5\% \text{ and } 10\%$.

R=2%

First of all, if we compare the plots presented in Fig.3.27 and A.1a we observe once again that the results are generally better when the network is trained with a set modified with the first method, in terms of MR and standard deviation. It emphasises the fact that the network gets used to the pattern of noise implementation, and that the second method is more robust in avoiding the consistency of the noise. Looking at the graphic in Fig.A.1b and comparing it with the one observed in Fig.3.28 reveals that the curves for the smaller amount of noise *r* are even more close from one to each other with the first method of noise implementation. In order to compare both of the methods, we subtract the curves from the first method with the ones from the second and plot it, as shown in Fig.A.1c. When compared with the graphic in Fig.3.24, we observe that adding noise significantly increased the difference of Acc_n between the two methods, and the standard deviation of it is also enhanced. This means that raising the ratio of noise makes the network exploit the repetitive position of noisy pixels even more than previously. This phenomenon is not verified for

$r = 10\%$. Finally, the subtraction of the curve for $r = 2\%$ with the other ones is shown in Fig.A.1d. Even if the curve for $r = 5\%$ has a MR closer to 0 in FigA.1c, it is observed that some of the digits show high discrepancies, for better comparison, it has been chosen to compute the subtraction with $r = 2\%$ instead. The results observed are almost the same as for Fig.3.26, all the points have matching uncertainties for the smallest *r*, and five points match for $r = 10\%$, which is two more compared to the results observed in Fig.3.26.

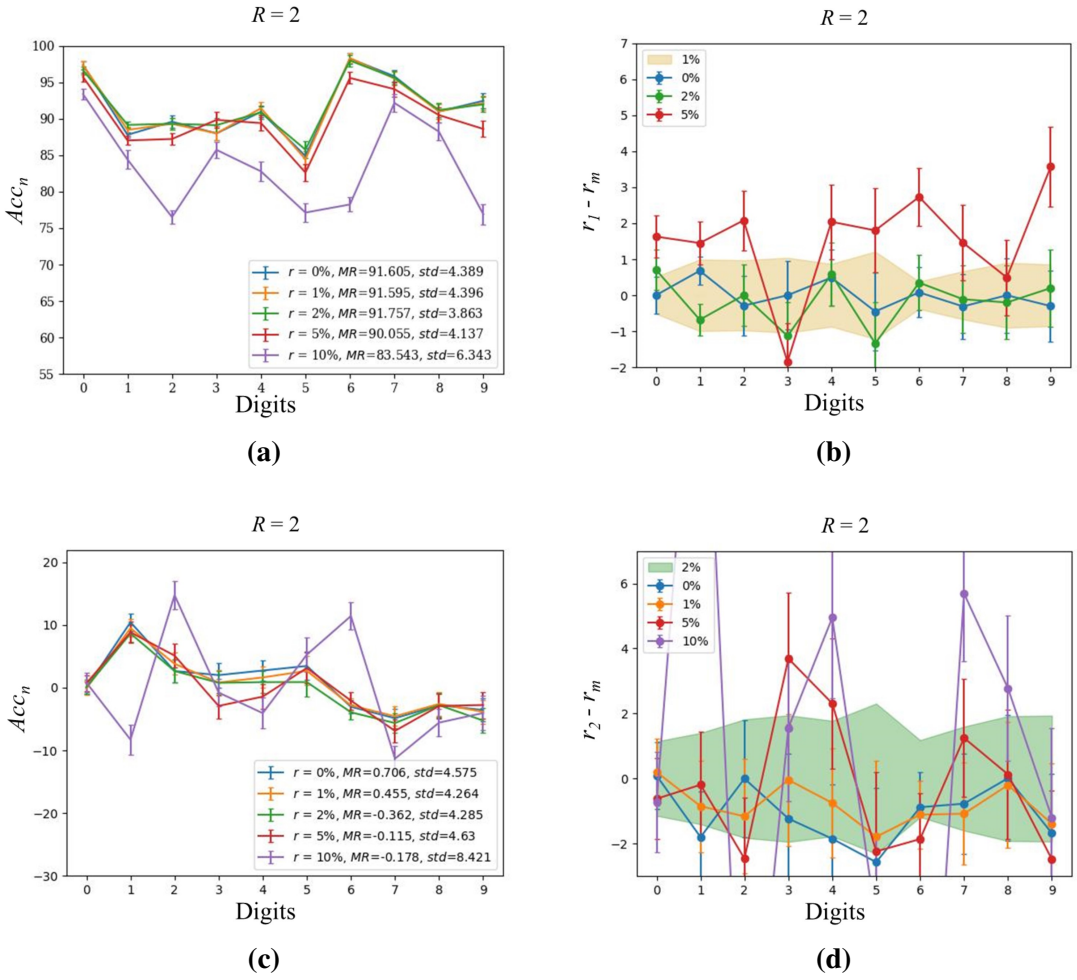


Figure A.1: Plots comparing the results of noise implementation with a ratio *R=2%* when using the first methods and confronting it with the second method. **a**, Acc_n for all digits with the first method of noise implementation. **b**, subtraction of the curves presented in **a** for $r = 0\%, 2\% and 5\%$ with the curve for $r = 1\%$. **c**, difference between the curves obtained with the second method and the first one. **d**, subtraction of the curves presented in **c** for $r = 0\%, 1\%, 5\%$ and 10% with the curve for $r = 2\%$

R=5%

Again, we start by comparing the plots presented in Fig.3.29 and A.2a. We observe the curves are significantly narrower with the first method, making the MR smaller for the smaller *r* and larger for the larger *r*. The first method seems to improve the behaviour of the converging curves with the augmentation of noise. Looking at the graphic in Fig.A.2b and comparing it with the one observed in Fig.3.30, we observe the previous statement tends to be verified. Indeed, twenty-nine out of the thirty points for the smaller *r* have matching uncertainties, while only twenty-two were observed with the other method.

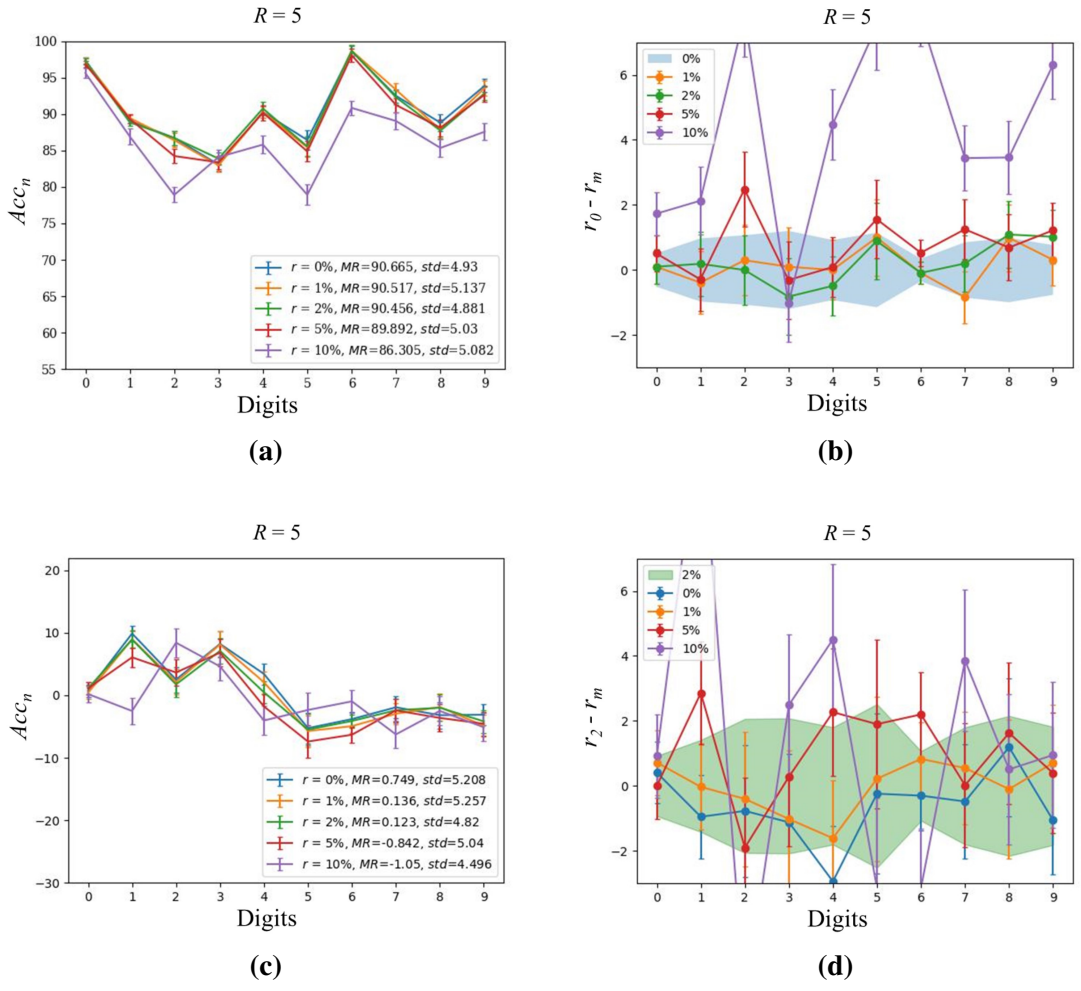


Figure A.2: Plots comparing the results of noise implementation with a ratio *R=5%* when using the first methods and confronting it with the second method. **a**, Acc_n for all digits with the first method of noise implementation. **b**, subtraction of the curves presented in **a** for $r = 1\%, 2\%, 5\%$and10% with the curve for $r = 0\%$. **c**, difference between the curves obtained with the second method and the first one. **d**, subtraction of the curves presented in **c** for $r = 0\%, 1\%, 5\%$ and 10% with the curve for $r = 2\%$

In order to compare both of the methods, we subtract the curves from the first method with the ones from the second and plot it, as shown in Fig.A.2c. When compared with the graphic in Fig.A.1c, we observe that adding noise significantly increased the difference of Acc_n between the two methods for $r = 5\%$ and decreased it for $r = 1\%$ and $r = 2\%$, while it slightly increased for $r = 0\%$, fluctuating. However, the standard deviation of the difference increased for all of these four results. A remarkable result is observed for $r = 10\%$ where the MR increased importantly but the standard deviation almost halved. Visually speaking, we notice the single peaks observed in Fig.A.1c tend to disappear, and the curve seems to get closer to the behaviour of the other curves. This last statement is observed in Fig.A.2d where the number of points for the curve representing $r = 10\%$ increased from five in Fig.A.1d to seven points having matching uncertainties with the curve $r = 2\%$. Regarding the other curves, a perfect correspondence has been reached, with thirty points out of thirty have matching uncertainties.

R=10%

Recurrently, we start by comparing the plots presented in Fig.3.31 and A.3a. For the first time, the results worsen with the first method compared to the second one, either regarding the Mr or the standard deviation. This is not true for the curve representing $r = 10\%$ where both the Mr and the standard deviation have better results. It seems the network is overflowed with noise, and is not able to exploit the redundancy of the perturbations when presented fewer amount of noise. However, it is still true for $r = R$. The curves in Fig.A.3a seem really tight, and to observe this, we plot the subtraction of the curve representing the results for $r = 1\%$ with the other in Fig.A.3b. When we compare it with the results observed in Fig.3.32, we witness the curves in Fig.A.3a are tighter, observed with the number of points having uncertainty matching with the one of the highest MR from Fig.A.3a, $r = 1\%$. Indeed, this number has been increased from twenty-six to twenty-nine. For the case $r = 10\%$, we enhanced the number from three points to six matching uncertainties. Now we want to compare both methods the way it has been done previously, and it observed in Fig.A.3c. As stated previously, it confirms only the curve for $r = 10\%$ is better in the second method, all the other have large differences and large standard deviation. Fig.A.3d shows this difference between the curve for $r = 10\%$ and the others is mainly witnessed for the lower digits, while from digit "4", all the points are matching with the uncertainty for the curve corresponding to $r = 1\%$.

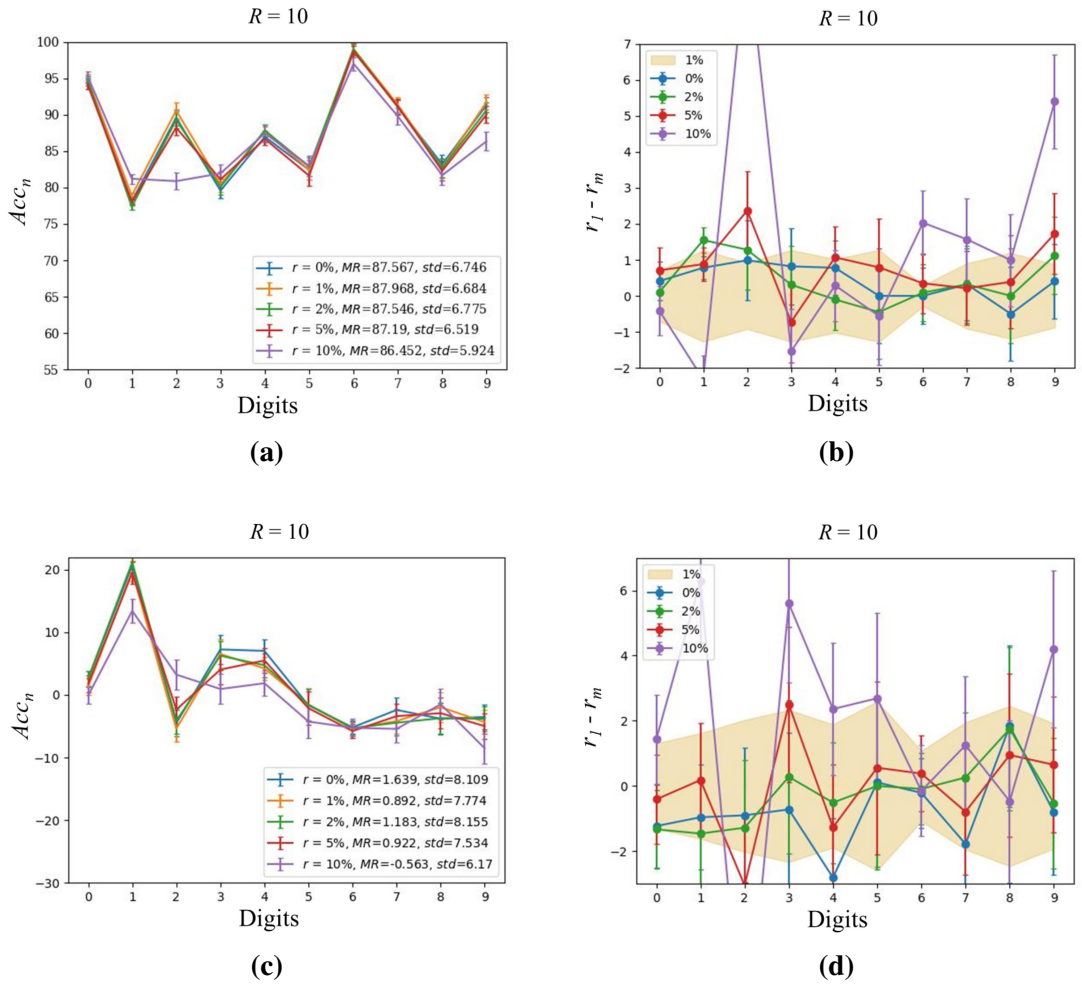


Figure A.3: Plots comparing the results of noise implementation with a ratio *R=10%* when using the first methods and confronting it with the second method. **a**, Acc_n for all digits with the first method of noise implementation. **b**, subtraction of the curves presented in **a** for $r = 0\%, 2\%, 5\%\text{and}10\%$ with the curve for $r = 1\%$. **c**, difference between the curves obtained with the second method and the first one. **d**, subtraction of the curves presented in **c** for $r = 0\%, 2\%, 5\%$ and 10% with the curve for $r = 1\%$

Conclusion

The first method is globally showing better results than the second one, induced by the network getting used to the repetition of the pattern of noise injection. Although this is true for $R = 1\%$, $R = 2\%$ and $R = 5\%$, it is not for $R = 10\%$ for the smaller amount for noise *r*. These results comfort us in choosing the second method as the more relevant for simulating the noise in the detecting of images in outer space.

Bibliography

[1] James Alfred Van Allen, George H Ludwig, Ernest Clark Ray, and Carl E McIlwain. Observation of high intensity radiation by satellites 1958 alpha and gamma. *Journal of Jet Propulsion*, 28(9):588–592, 1958.

[2] Karen C. Fox. Nasa's van allen probes spot an impenetrable barrier in space. https://web.archive.org/web/20200306135737/https://www.nasa.gov/content/goddard/van-allen-probes-spot-impenetrable-barrier-in-space. Accessed: 2014-11-26.

[3] Background: Trapped particle radiation models. *SPENVIS*. URL https://www.spenvis.oma.be/help/background/traprad/traprad.html.

[4] EG Stassinopoulos and James P Raymond. The space radiation environment for electronics. *Proceedings of the IEEE*, 76(11):1423–1442, 1988.

[5] Rossi B. *Cosmic Rays*. New York: McGraw-Hill, 1964.

[6] JA Simpson. Elemental and isotopic composition of the galactic cosmic rays. *Annual Review of Nuclear and Particle Science*, 33(1):323–382, 1983.

[7] E.R Benton and E.V Benton. Space radiation dosimetry in low-earth orbit and beyond. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 184(1):255–294, 2001. ISSN 0168-583X. doi: https://doi.org/10.1016/S0168-583X(01)00748-0. Advanced Topics in Solid State Dosimetry.

[8] W Suparta and S K Zulkeple. Spatial analysis of galactic cosmic ray particles in low earth orbit/near equator orbit using spenvis. *Journal of Physics: Conference Series*, 495(1):012040, apr 2014. doi: 10.1088/1742-6596/495/1/012040. URL https://dx.doi.org/10.1088/1742-6596/495/1/012040.

[9] G.D. Badhwar, W. Atwell, B. Cash, M. Weyland, V.M. Petrov, I.V. Tchernykh, Yu.A. Akatov, V.A. Shurshakov, V.V. Arkhangelsky, V.V. Kushin, N.A. Klyachin, E.V. Benton, A.L. Frank, E.R. Benton, L.A. Frigo, V.E. Dudkin, Yu.V. Potapov, N. Vana, W. Schoner, and M. Fugger. Intercomparison of radiation measurements on sts-63. *Radiation Measurements*, 26(6):901–916, 1996. ISSN 1350-4487. doi: https://doi.org/10.1016/S1350-4487(96)00082-0. URL https://www.sciencedirect.com/science/article/pii/S1350448796000820. Space Radiation Results Of The Long Duration Exposure Facility.

[10] Gautam D Badhwar. Radiation dose rates in space shuttle as a function of atmospheric density. *Radiation measurements*, 30(3):401–414, 1999.

[11] SÉbastien Bourdarie and Michael Xapsos. The near-earth space radiation environment. *IEEE Transactions on Nuclear Science*, 55(4):1810–1832, 2008. doi: 10.1109/TNS.2008.2001409.

[12] Guenther Reitz, Thomas Berger, and Daniel Matthiae. Radiation exposure in the moon environment. *Planetary and Space Science*, 74(1):78–83, 2012. ISSN 0032-0633. doi: https://doi.org/10.1016/j.pss.2012.07.014. Scientific Preparations For Lunar Exploration.

[13] Firmato il contratto asi-thales alenia space per la prima missione italiana di in orbit-servicing. https://www.asi.it/2023/05/firmato-il-contratto-asi-thales-alenia-space-per-la-prima-missione-italiana-di-in-orbit-servicing/. 2023-05-16.

[14] Paolo Branchini, Andrea Fabbri, Sacha Cormenier, Marco Bernardini, Giovanni Romanelli, Enrico Preziosi, Roberto Senesi, Carla Andreani, C. Frost, Carlo Cazzaniga, Toni Fabio Catalano, and Mario Buffardo. Irradiation tests for commercial off-the shelf components with atmospheric-like neutrons and heavy-ions, 2023.

[15] Gianluca Furano, Stefano Di Mascio, Tomasz Szewczyk, Alessandra Menicucci, Luigi Campajola, Francesco Di Capua, Andrea Fabbri, and Marco Ottavi. A novel method for see validation of complex socs using low-energy proton beams. In *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 131–134. IEEE, 2016.

[16] Md Mahbubur Rahman, Divya Shankar, and Shreya Santra. Analysis of radiation environment and its effect on spacecraft in different orbits. In *Conference: international Astronautical Congress (IAC2017)*, pages 8073–8079, 2017.

[17] *RM0342 - SPC56XL70xx Reference manual*. STMicroelectronics, October 2013.

[18] UK Research, Science Innovation, ISIS Neutron Technology Facilities Council, and Muon Source. How isis works - in depth. https://www.isis.stfc.ac.uk/Pages/How-ISIS-works--in-depth.aspx, . Accessed on: September 15, 2021.

[19] UK Research, Science Innovation, ISIS Neutron Technology Facilities Council, and Muon Source. Chipir technical information. https://www.isis.stfc.ac.uk/Pages/Chipir-technicalinformation.aspx, . Accessed on: September 15, 2021.

[20] C Andreani, A Pietropaolo, A Salsano, G Gorini, M Tardocchi, A Paccagnella, S Gerardin, CD Frost, S Ansell, and SP Platt. Facility for fast neutron irradiation tests of electronics at the isis spallation neutron source. *Applied physics letters*, 92(11), 2008.

[21] Avner Haran, Nir M Yitzhak, Eran Mazal-Tov, Eitan Keren, David David, Nati Refaeli, Enrico Preziosi, Roberto Senesi, Carlo Cazzaniga, Christopher D Frost, et al. Ultralow power system-on-chip sram characterization by alpha and neutron irradiation. *IEEE Transactions on Nuclear Science*, 68(11):2598–2608, 2021.

[22] M Bagatin, Simone Gerardin, Alessandro Paccagnella, C Andreani, Giuseppe Gorini, and CD Frost. Temperature dependence of neutron-induced soft errors in srams. *Microelectronics Reliability*, 52(1):289–293, 2012.

[23] UK Research, Science Innovation, ISIS Neutron Technology Facilities Council, and Muon Source. Chipir. https://www.isis.stfc.ac.uk/Pages/ChipIR.aspx, . Accessed on: October 12, 2021.

[24] Davide Chiesa, Massimiliano Nastasi, Carlo Cazzaniga, Marica Rebai, Laura Arcidiacono, Ezio Previtali, Giuseppe Gorini, and Christopher D Frost. Measurement of the neutron flux at spallation sources using multi-foil activation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 902:14–24, 2018.

[25] Matteo Cecchetto, Pablo Fernández-Martínez, Rubén García Alía, Rudy Ferraro, Salvatore Danzeca, Frédéric Wrobel, Carlo Cazzaniga, and Christopher D Frost. See flux and spectral hardness calibration of neutron spallation and mixed-field facilities. *IEEE Transactions on Nuclear Science*, 66(7):1532–1540, 2019.

[26] James F Ziegler, Matthias D Ziegler, and Jochen P Biersack. Srim–the stopping and range of ions in matter (2010). *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 268(11-12):1818–1823, 2010.

[27] Texas Instruments. Slvk046 application report - heavy ion orbital environment single-event effects estimations. https://www.ti.com/lit/pdf/slvk046, May 2020.

[28] VDS Dhaka, NV Tkachenko, H Lemmetyinen, E-M Pavelescu, J Konttinen, M Pessa, Kai Arstila, and Juhani Keinonen. Room-temperature self-annealing of heavy-ion-irradiated ingaas/gaas quantum wells. *Electronics Letters*, 41(23):1304–1305, 2005.

[29] David M Hiemstra and Ewart W Blackmore. Let spectra of proton energy levels from 50 to 500 mev and their effectiveness for single event effects characterization of microelectronics. *IEEE Transactions on Nuclear Science*, 50(6):2245–2250, 2003.

[30] AMD. Versal ai core series vck190 evaluation kit. https://www.xilinx.com/products/boards-and-kits/vck190.html.

[31] *VCK190 Evaluation Board User Guide (UG1366)*. AMD, March 2023.

[32] *Versal Adaptive SoC System Software Developers Guide (UG1304)*. AMD, October 2023.

[33] *Versal ACAP VCK190 Base Targeted Reference Design (UG1442)*. AMD, January 2021.

[34] Digilient. Zybo z7. https://digilent.com/reference/programmable-logic/zybo-z7/start.

[35] *Zybo Z7 Board Reference Manual*. Digilent, February 2018.

[36] Advanced Micro Devices. Pynq: Python productivity for adaptive computing platforms. https://pynq.readthedocs.io/en/latest/#.

[37] Terence Sanger and Pallavi N. Baljekar. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[38] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[39] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. Multi-column deep neural networks for image classification. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 02 2012. doi: 10.1109/CVPR.2012.6248110.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[41] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57–83, 2002. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(01)00129-1. URL https://www.sciencedirect.com/science/article/pii/S0004370201001291.

[42] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, Feb 2017. ISSN 1476-4687. doi: 10.1038/nature21056. URL https://doi.org/10.1038/nature21056.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[44] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003. URL http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[45] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9: 249–256, 01 2010.

[46] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[47] Haixia Xiao. Weather phenomenon database (weapd), 2021. URL https://doi.org/10.7910/DVN/M8JQCR.

[48] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine*

Learning, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[49] Yangfan Zhou, Xin Wang, Mingchuan Zhang, Junlong Zhu, Ruijuan Zheng, and Qingtao Wu. Mpce: A maximum probability based cross entropy loss function for neural network classification. *IEEE Access*, PP:1–1, 10 2019. doi: 10.1109/ACCESS.2019.2946264.

[50] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212, Apr 2022. ISSN 2198-5812. doi: 10.1007/s40745-020-00253-5.

[51] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. URL https://api.semanticscholar.org/CorpusID:205001834.

[52] Pankaj Mehta, Ching-Hao Wang, Alexandre G R Day, Clint Richardson, Marin Bukov, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Phys Rep*, 810:1–124, March 2019.

[53] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. Multi-column deep neural networks for image classification, 2012.

[54] IBM. Convolution neural networks. https://www.ibm.com/topics/convolutional-neural-networks.

[55] Mayank Mishra. Convolutional neural networks, explained. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, august 2020.

[56] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[57] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[58] Ph. Busson, R. Nóbrega, and J. Varela. Modular neural networks for on-line event classification in high energy physics. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 410(2):273–283, 1998. ISSN 0168-9002. doi: https://doi.org/10.1016/S0168-9002(98)00234-4. URL https://www.sciencedirect.com/science/article/pii/S0168900298002344.

[59] Tuse Asrav and Erdal Aydin. Physics-informed recurrent neural networks and hyperparameter optimization for dynamic process systems. *Computers & Chemical Engineering*, 173:108195, 2023. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2023.108195. URL https://www.sciencedirect.com/science/article/pii/S0098135423000649.

[60] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1):4308, Jul 2014. ISSN 2041-1723. doi: 10.1038/ncomms5308.

[61] Giorgio Apollinari, O Brüning, Tatsushi Nakamoto, and Lucio Rossi. High luminosity large hadron collider hl-lhc. *arXiv preprint arXiv:1705.08830*, 2017.

[62] Wahid Bhimji, Steven Andrew Farrell, Thorsten Kurth, Michela Paganini, Prabhat, and Evan Racah. Deep neural networks for physics analysis on low-level whole-detector data at the lhc. *Journal of Physics: Conference Series*, 1085(4):042034, sep 2018. doi: 10.1088/1742-6596/1085/4/042034. URL https://dx.doi.org/10.1088/1742-6596/1085/4/042034.

[63] Dalya Baron. Machine learning in astronomy: A practical overview. *arXiv preprint arXiv:1904.07248*, 2019.

[64] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 65–74, 2017.

[65] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016.

[66] Alessandro Pappalardo. Xilinx/brevitas, 2023. URL https://doi.org/10.5281/zenodo.3333552.

[67] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[68] Xilinx. Finn. https://finn.readthedocs.io/en/latest/.

[69] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[70] Abbas El Gamal and Helmy Eltoukhy. Cmos image sensors. *IEEE Circuits and Devices Magazine*, 21(3):6–20, 2005.

[71] Matthew I Plewa and Justin Vandenbroucke. Detecting cosmic rays using cmos sensors in consumer devices. In *Academic High Altitude Conference*, volume 2015. Iowa State University Digital Press, 2015.

[72] Pankaj Mehta, Ching-Hao Wang, Alexandre G R Day, Clint Richardson, Marin Bukov, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Phys Rep*, 810:90–92, March 2019.

[73] Henrik Svensmark and Eigil Friis-Christensen. Variation of cosmic ray flux and global cloud coverage—a missing link in solar-climate relationships. *Journal of Atmospheric and Solar-Terrestrial Physics*, 59(11):1225–1232, 1997. ISSN 1364-6826. doi: https://doi.org/10.1016/S1364-6826(97)00001-1. URL https://www.sciencedirect.com/science/article/pii/S1364682697000011.

[74] Pieter P. The damped harmonic oscillator. https://tttapa.github.io/Pages/Arduino/Audio-and-Signal-Processing/VU-Meters/Damped-Harmonic-Oscillator.pdf.

[75] INFN-LNL. Cn. https://www.lnl.infn.it/cn/.

[76] James W Meadows. The 9be (d, n) thick-target neutron spectra for deuteron energies between 2.6 and 7.0 mev. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 324(1-2): 239–246, 1993.

[77] Chelsea Voss, Nick Cammarata, Gabriel Goh, Michael Petrov, Ludwig Schubert, Ben Egan, Swee Kiat Lim, and Chris Olah. Visualizing weights. *Distill*, 6(2):e00024–007, 2021.

[78] Diego Alberici, Pierluigi Contucci, and Emanuele Mingione. Deep boltzmann machines: rigorous results at arbitrary depth. In *Annales Henri Poincaré*, volume 22, pages 2619–2642. Springer, 2021.

[79] Jérôme Tubiana and Rémi Monasson. Emergence of compositional representations in restricted boltzmann machines. *Physical review letters*, 118(13):138301, 2017.

[80] Giancarlo Fissore. *Generative modeling: statistical physics of Restricted Boltzmann Machines, learning with missing information and scalable training of Linear Flows*. PhD thesis, université Paris-Saclay, 2022.

[81] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[82] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[83] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[84] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.