**FACULTY AT SOGAMOSO CAMPUS**

**SCHOOL OF ELECTRONIC ENGINEERING**

**THESIS PROJECT PROPOSAL, MODALITY: ACTIVE PARTICIPATION IN A**

**RESEARCH GROUP RECOGNIZED BY THE DIN**

**DATE OF SUBMISSION:** 05 - 08- 2024

**PROJECT IDENTIFICATION**

**TITLE:** High-level fault injector framework for convolutional neural networks to evaluate their reliability.

**PROPONENT(S)**: David Esteban Gonzalez Becerra **CÓDIGO:** 201812189

**ADVISORS: Ing. Luis Ariel Mesa Mesa, Ph.D(c).**

**Ing. Wilson Javier Pérez Holguín, Ph.D**

**AREA OF SPECIALIZATION: Digital Electronic**

**RESEARCH GROUP: Group of Research in Robotics and Industrial Automation - GIRA**

**P.I.I.I. NAME: Reliability Assessment of Deep Neural Networks Implemented in Parallelized Hardware Platforms**

**RESEARCH LINE: Design and Test**

1. **BACKGROUND** 
   1. **Background**

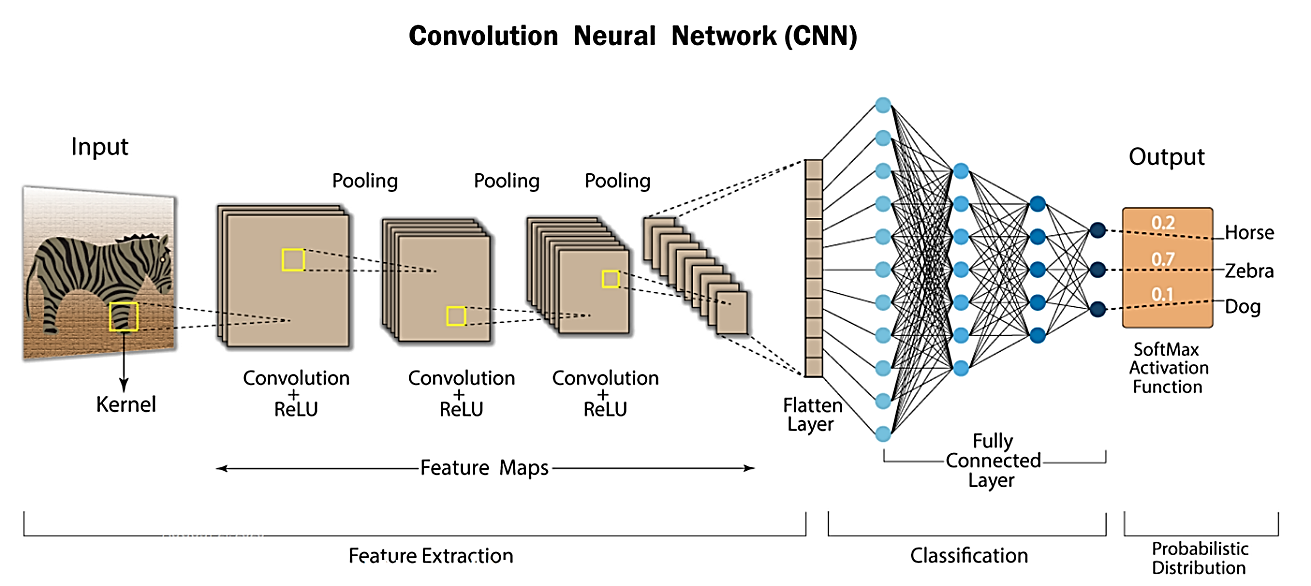
**Convolutional Neural Networks**

Artificial intelligence, particularly Artificial Neural Networks (ANNs), has experienced exponential growth in its application across various fields, ranging from image recognition to decision-making in autonomous navigation systems. Although these neural networks are highly resilient due to their characteristics, they can present vulnerabilities that undermine their reliability, affecting their use in applications termed "mission-critical" applications. A mission-critical application is one where the occurrence of a failure is not tolerable as it may impact the integrity of human life, the environment, and/or the equipment itself.

Convolutional Neural Networks (CNNs) are a deep learning (DL) architecture developed by (LeCun & others, 1989) to process information in the form of numerical arrays (Pérez Cerdeira, 2021). Currently, they are used to identify patterns, making them ideal for various tasks such as image recognition, segmentation, and classification, among others. Their performance has reached outstanding levels, driving significant advances in various applications, such as medical diagnosis, scientific data analysis, process control, autonomous vehicle navigation, and more.

The architecture of a CNN consists of multiple layers that apply filters to the input dataset to propagate the results to subsequent layers. This technique facilitates the detection and extraction of relevant features, enriching the learning process and effective pattern representation in the network. Each layer performs convolution and pooling operations to capture important details and reduce dimensionality, allowing the network to learn progressively and address complex tasks efficiently. (Moreno, 2019) adds that the convolutional layers of a neural network divide the input matrix into smaller subsets through mathematical operations with a kernel or reduced-dimension filter.

CNNs, according to (Pérez Cerdeira, 2021), represent a revolutionary approach in neural networks by replacing multiplications and one-dimensional weights used in conventional networks, such as fully connected or multilayer perceptrons, with convolutions and filter masks. Similarly, (Moreno, 2019) describes CNNs as a set of layers designed to emulate the visual cortex of the human brain and detect various features in inputs. Figure 1 shows a general representation of the architecture of a CNN.

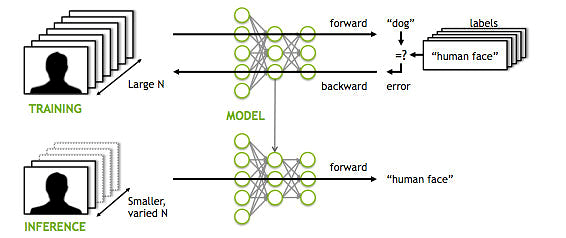


*Figure 1. General Architecture of a CNN.*

*Fuente:* [*https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5*](https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5)

The management of a neural network is divided into two important aspects: training and inference. Training is the process by which the network "learns" from a set of labeled input data (dataset). With each new piece of data introduced, the network progressively adjusts its internal parameters, known as weights and biases, with the goal of minimizing a loss function. This adjustment (minimization of the loss function) is achieved by comparing the difference between the network's predictions and the actual values defined by the labels in the training dataset. On the other hand, inference is the process where the trained network is actually put into operation. New input data is fed into the network, and it must make the respective prediction or classification of classes. This is illustrated in Figure 2.

For the implementation of a CNN, there can be two levels: the hardware level and the application level. The former involves the hardware platform running the neural network, while the latter includes only the application level.



*Figure 2. Training and Inference Processes of an ANN.*

*Fuente:* [*https://mitxpc.com/pages/ai-inference-applying-deep-neural-network-training*](https://mitxpc.com/pages/ai-inference-applying-deep-neural-network-training)

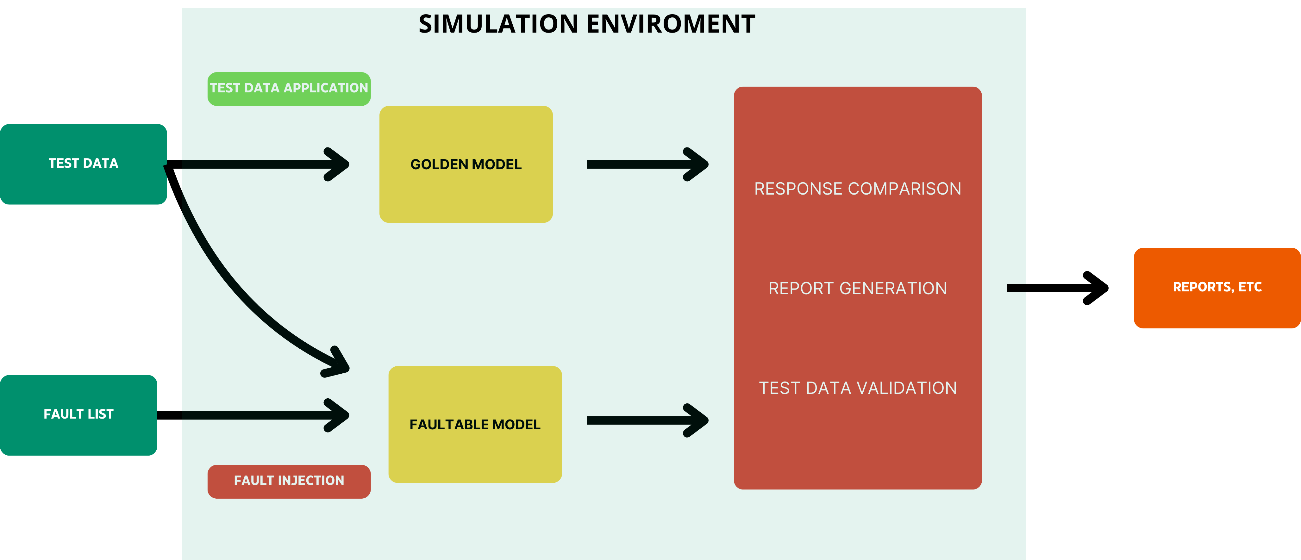
(Moreno, 2019) indicates that in the field of artificial intelligence, there are various development environments, among which Python and Matlab stand out. For Python, there are platforms like TensorFlow or PyTorch that allow for the construction, training, and evaluation of machine learning models through a series of libraries.

**Reliability of a Convolutional Neural Network**

Reliability in a neural network refers to the network's ability to produce consistent and accurate results across a wide range of situations and input data associated with the application. For mission-critical applications, this aspect is crucial, as a failure can trigger undesirable adverse effects (Hernández et al., 2022).

CNNs are inherently deep learning architectures with high resilience due to their generalization capabilities, noise tolerance, and adaptive learning. However, their reliability can be compromised by events such as programming errors, training data errors, overfitting, data preprocessing errors, errors in network architecture, incorrect hyperparameters, optimization problems, inadequate hardware requirements, code bugs, hardware errors, malicious attacks, and others.

Reliability assessment allows experts and users of these tools to make them more robust against vulnerabilities; however, identifying specific situations that can cause failures is not an easy task. Various approaches and techniques can be used to intentionally inject faults into the network to uncover critical points. In this scenario, the performance of the network subjected to a controlled fault attack is typically evaluated against a fault-free model of the same network, referred to as the "Golden Model," as illustrated in Figure 3. Reliability assessment can also aim at the application level (CNN model independent of technology) and the entire system, which includes both the network model and the hardware platform.



*Figure 3. Fault Injection and Reliability Assessment Scheme.*

One technique for evaluating the reliability of a system is fault injection. In the context of Convolutional Neural Networks (CNNs), this involves deliberately introducing errors, noise, or unusual conditions into the input, model, or training process to assess how the CNN responds to these situations. These tests help identify vulnerabilities, improve robustness, and better understand the behavior of the CNN under adverse conditions. Examples of fault injection in CNNs include introducing noise into input data, perturbations in weights, removal of layers or nodes, altering architecture, adversarial attacks, data rotation or translation, disrupting network connectivity, and simulating lighting and environmental conditions.

The literature shows that different types of CNN architectures have been proposed to address various needs. These architectures have considered variations such as input image size, type of operations, number of layers, type of activation functions, data representation, among others. For example, (Bengio & Haffner, 1998) proposed a CNN architecture called LeNet-5, which, for processing 32x32 pixel images, included seven layers, not counting the input layer, each containing trainable parameters called weights and biases. In (Krizhevsky et al., 2017), another network model called AlexNet was proposed, which processes 224x224 images through an architecture of five convolutional layers and three fully connected layers. The output of the last fully connected layer feeds into a softmax classifier with 1000 outputs, producing a distribution over the same number of class labels. Some CNN architectures have proposed including an extensive number of convolutional layers, such as VGG16 and VGG19, where the number represents the model's depth (Renza Torres & Ballesteros, 2023), aiming to improve feature extraction and accuracy. On the other hand, other initiatives like the MobileNet model presented in (Howard et al., 2017) have proposed performance improvements based on less extensive and computationally less intensive architectures.

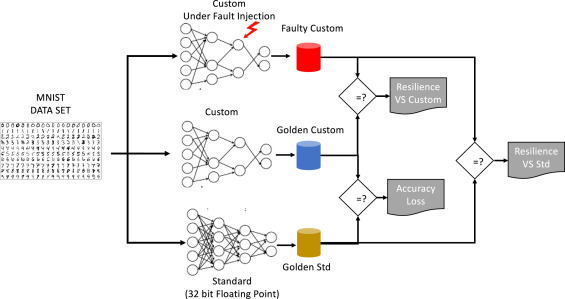
Other approaches have sought to optimize performance and reduce the size of some CNN architectures by applying techniques such as eliminating less significant neurons and/or quantizing weights and biases. In (Molchanov et al., 2017) and (Sarkar, 2020), a method for optimizing CNNs called pruning is proposed. Pruning is a technique that involves identifying and removing unnecessary connections in a neural network to simplify it. (Molchanov et al., 2017) focused on the concept of transfer learning to adapt previously trained networks to specialized tasks and reduce resource usage such as memory. (Sarkar, 2020) proposed a new CNN pruning technique based on incremental pruning, achieving higher accuracy with fewer prior trainings, leading to reduced times compared to other techniques studied previously.

Considering that this project aims to address the evaluation of CNN reliability at the application level, it is important to determine the aspects related to this topic when considering the development environment. In (Kaur & Bahl, 2014), an overview of reliability is provided, considering the differences between software and hardware. For software, it should be noted that its complexity tends to be high due to its intangible and abstract nature, making its verification and validation challenging. Similarly, (Bengio & Haffner, 1998) presents two approaches for thoroughly evaluating software reliability: the first is based on testing plan evaluation, ensuring that the system contains the specified functionality, while the second is based on evaluating the number of errors and the rate of detection and correction.

In the Group of Research in Robotics and Industrial Automation (GIRA) at the Universidad Pegadógica y Tecnológica de Colombia (UPTC), various projects have been developed aimed at addressing new trends in artificial intelligence, based on areas such as system design and testing, robotics, advanced digital design, biotechnology, and hardware accelerators. In this context, this work is framed within the projects "Development of a Mobile Robotic Platform Based on FPGA for Teaching Robotics and Embedded Systems" (SGI 2847) and "Reliability Assessment of Deep Neural Networks Implemented in Parallelized Hardware Platforms". The latter project is being executed by Engineer Luis Ariel Mesa, and it aims to support its execution through the evaluation of the reliability of a Convolutional Neural Network implemented at a high level. In this sense, the project will address only the application level of the neural network, using a development environment such as Python or Matlab.

* 1. **State of art**

One technique for evaluating reliability is fault injection, which involves intentionally introducing errors into the system to observe its behavior in response to these perturbations and to identify and quantify failure modes that are not detectable through traditional testing. In (Ruospo et al., 2020) and (Ruospo et al., 2021), fault injection studies are conducted on CNNs using software with two types of data representation for weights: floating-point and fixed-point. One of their main objectives is to identify the optimal combination of data type, bit reduction, and reliability. The results of these studies indicate that, for a CNN, using fixed-point data provides the best balance between memory usage and reliability. Figure 4 shows a general overview of the fault injection procedure conducted by (Ruospo et al., 2021).

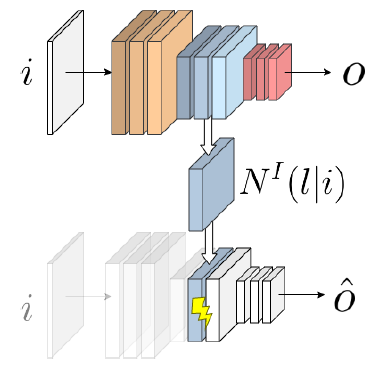


*Figure 4. Fault Injection Scenario Proposed in the Work of (Ruospo et al., 2021)*

In the literature, some tools are available for evaluating the reliability of CNNs implemented in high-level languages like Python, which emulate hardware and software faults. These tools are based on injecting errors into machine learning models to detect potential vulnerabilities. Fault injection can be performed before or during the inference process, specifically in the learning operators. Some of the tools used for this purpose include TensorFI, PyTorchFI, ARES, and Fidelity.

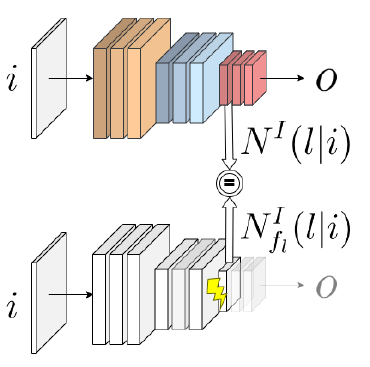
TensorFI is a highly configurable fault injection (FI) tool characterized by its flexibility, ease of use, and portability. This tool integrates easily with TensorFlow to assess the resilience of CNNs against potential faults (Chen et al., 2020). Another tool to mention for this task is the PyTorchFI library, which allows users to make perturbations in neural network weights and/or neurons in the convolutional operations of a DNN either dynamically or statically (Mahmoud et al., 2020).

With these tools, frameworks have also been developed to enhance the performance and effectiveness of fault injection in convolutional neural networks. One such framework, called SCI-FI (Smart, aCcurate and unIntrusive Fault-Injector), demonstrates how to reduce fault execution times by employing two distinct techniques, which can be used separately or in combination. The first technique, called Delayed Start, involves saving the complete execution of the Golden Network. This way, if faults need to be applied from a specific layer of the network, it is not necessary to re-execute the entire network. Instead, the network is executed from the layer where the batch of faults was injected (Gavarini et al., 2023). A diagram of the technique in action can be seen in Figure 5.



*Figure 5. Use of the Delayed Start Technique on a Convolutional Neural Network. (Gavarini et al., 2023).*

The second technique, called Fault Dropping, refers to the elimination of faults that could occur within the network during execution, similar to what happens when using the ReLU activation function, which prevents the propagation of negative weights. This method involves comparing the Golden Network with the Faultable Network, layer by layer. If at any point the output feature maps of the two networks become identical, inference is immediately halted, as it is concluded that the faults will not impact the output of the neural network (Gavarini et al., 2023). A visualization of the technique can be seen in Figure 6.



*Figure 6. Use of the Fault Dropping Technique on a Convolutional Neural Network. (Gavarini et al., 2023)*

1. **PROBLEM IDENTIFICATION**

However, as the use of CNN-based systems has expanded into critical applications and safety environments, there is growing concern about their reliability and robustness. Despite their successes, deep learning models, including CNNs, can face significant challenges in maintaining consistent and accurate performance under diverse and challenging conditions.

Various factors can compromise the reliability of CNNs. For example, the presence of unwanted data, such as errors in training data or even inaccuracies in weights, can trigger serious failures in predictions, with potentially significant repercussions in critical applications, such as incorrect medical diagnoses or autonomous vehicle navigation. Additionally, errors in the inference process can stem from various factors, ranging from software failures to vulnerabilities in the hardware used (Hernández et al., 2022).

In this context, research into the reliability and robustness of CNNs has become essential to ensure their applicability in mission-critical applications. Addressing these challenges requires not only the development of more accurate and resilient models but also the implementation of comprehensive evaluation and validation strategies that allow for the identification and mitigation of potential weaknesses. Ultimately, advancing the understanding and improvement of CNN reliability will not only enhance their performance in current tasks but also open new opportunities for their implementation in fields where safety and precision are crucial.

* 1. **Research Question**

How to evaluate the reliability of a CNN implemented on a Python-based development environment considering the network architecture and data representation?

* 1. **Hypothesis**

The controlled injection of faults into a previously trained neural network will have a significant impact on its reliability and performance in mission-critical applications. It is anticipated that the deliberate manipulation of factors such as the weights of the most sensitive neurons will result in a decrease in the precision and stability of the neural network.

Additionally, the hypothesis is that both the magnitude and nature of the injected faults will influence the degree of deterioration in the network's reliability. It is expected that the results of this study will show substantial differences in performance between the network with fault injection and the fault-free network, supporting the importance of investigating reliability in critical application contexts.

Ultimately, it is projected that the findings of this research will provide empirical evidence supporting the need to develop risk mitigation strategies and improve the reliability of neural networks used in mission-critical applications. This, in turn, will contribute to advancing the understanding of fault tolerance mechanisms in artificial intelligence.

1. **JUSTIFICATION**

In this context, the present thesis proposal aims to evaluate, at a high level, the reliability of a neural network through controlled fault injection and compare its behavior with an identical fault-free network. The use of Python-based environments such as TensorFlow and PyTorch, among others, is considered for this purpose.

Creating a specific test setup for a Convolutional Neural Network (CNN) would be a valuable tool for improving the robustness of any neural network of that architecture. A well-designed test environment will allow a thorough evaluation of all the components that make up the network, from the dataset to the architecture and internal characteristics. This setup will provide a systematic platform for conducting exhaustive tests on each part of the neural network, ensuring that every aspect, from data integrity to model parameter robustness, is carefully analyzed.

Through a rigorous process of fault injection in a previously trained neural network, its performance can be compared with the same fault-free model. This approach will allow the exploration of various types of faults, such as the alteration of the weights of the most sensitive neurons, and evaluate their impact. By conducting a detailed analysis of performance metrics, such as accuracy, the study will investigate how these faults affect the reliability of the neural network and its ability to maintain acceptable performance.

The results of this study will provide valuable information for improving the reliability of neural networks in mission-critical applications. Additionally, this work is expected to contribute to advancing the understanding of fault tolerance mechanisms in neural networks and promote the development of risk mitigation strategies in AI-based systems.

1. **GENERAL OBJECTIVE**

Design a framework for the reliability evaluation of a CNN implemented on a Python-based environment considering the network architecture and data representation.

* 1. **SPECIFIC OBJECTIVES**

1. Identify types of CNN architectures, their data representation, and reliability evaluation mechanisms for high-level systems.
2. Develop a testing environment that includes implementation, training, and tools to evaluate the reliability of CNN’s.
3. Execute tests to assess the reliability of selected CNN's and collect the data.
4. Analyze and validate the obtained results.
5. **METHODOLOGY**

This project aims to apply theoretical and experimental knowledge to address a real problem related to the reliability of CNNs, classifying it as applied and experimental research. In this investigation, controlled tests will be conducted to evaluate the reliability of CNNs by applying controlled fault injection. The reliability and performance of the network will be analyzed based on different fault conditions. This experimental methodology allows us to establish causal relationships and accurately measure the effect of independent variables, in this case, fault injection, on the dependent variables, which are the reliability and performance of the CNN.

1. **ACTIVITIES**

The activities that will enable the execution of the various specific objectives are outlined in this section.

a. To achieve the first specific objective, “Identify types of CNN architectures, their data representation, and reliability evaluation mechanisms for high-level systems,” the following activities are proposed:

1. Investigate the different types of existing neural network architectures.
2. Explore the data representations that can occur in neural networks and examine their effects.
3. Review the various methods for evaluating the reliability of convolutional neural networks.

b. To achieve the second specific objective, “Develop a test environment that includes implementation, training, and tools to evaluate the reliability of CNNs,” the following activities are proposed:

1. Implement 3 types of network architectures (Alexnet, VGG, LeNet-5).
2. Perform training and inference of the selected architectures.
3. Implement the “Pruning” method in the already trained architectures.
4. Investigate the format in which the trained network data is delivered and visualize this data.

c. To achieve the third specific objective, “Conduct tests to evaluate the reliability of selected CNNs and collect data,” the following activities are proposed:

1. Implement code that allows access to the information of each layer and modify at least one data point in them.
2. Perform inferences on the trained networks: Golden model and the fault model.
3. Collect the information generated during the inference of both models.

d. To achieve the fourth specific objective, “Analyze and validate the obtained results,” the following activities are proposed:

1. Analyze the obtained data and compare the results of the Golden model with those of the fault model.
2. Conclude the obtained results.
3. **TIMELINE**

To achieve this objective, we have designed a series of activities to be carried out in specific stages of the project. These activities are aimed at addressing critical aspects of the neural network's reliability, including performance evaluation.

Below, we present a table of activities that details the key steps of this project, including estimated start and end dates. This plan will be developed over a determined period, and we expect the activities to lead us to a comprehensive and rigorous evaluation of the reliability of our neural network.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stage | Activity | 2024 | | | | | | | | | | | | | | | |
| Month 1 | | | | Month 2 | | | | Month 3 | | | | Month 4 | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| a | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| b | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| c | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| d | 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

*Table 1. Activity timeline.*

**Bibliografía**

Bengio, Y., & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition*. *86*(11).

Chen, Z., Narayanan, N., Fang, B., Li, G., Pattabiraman, K., & DeBardeleben, N. (2020). Tensorfi: A flexible fault injection framework for tensorflow applications. *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, *2020*-*Octob*, 426–435. https://doi.org/10.1109/ISSRE5003.2020.00047

Gavarini, G., Ruospo, A., & Sanchez, E. (2023). SCI-FI: A Smart, aCcurate and unIntrusive Fault-Injector for Deep Neural Networks. *Proceedings of the European Test Workshop*, *2023*-*May*, 1–6. https://doi.org/10.1109/ETS56758.2023.10173957

Hernández, A. Z., Jesús, H., Santiago, J., & Lee, M. M. (2022). *Métricas de rendimiento para evaluar el aprendizaje automático en la clasificación de imágenes petroleras utilizando redes neuronales convolucionales .* *2215*, 4624–4637.

Howard, A. G., Chen, B., & Wang, W. (2017). *MobileNets : Efficient Convolutional Neural Networks for Mobile Vision MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications*. *April 2017*.

Kaur, G., & Bahl, K. (2014). *Software Reliability , Metrics , Reliability Improvement Using Agile Process*. *1*(3), 143–147.

Krizhevsky, B. A., Sutskever, I., & Hinton, G. E. (2017). *ImageNet Classification with Deep Convolutional Neural Networks*. https://doi.org/10.1145/3065386

LeCun, Y., & others. (1989). Generalization and network design strategies. *Connectionism in Perspective*, 143–155.

Mahmoud, A., Aggarwal, N., Nobbe, A., Vicarte, J. R. S., Adve, S. V., Fletcher, C. W., Frosio, I., & Hari, S. K. S. (2020). PyTorchFI: A Runtime Perturbation Tool for DNNs. *Proceedings - 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN-W 2020*, 25–31. https://doi.org/10.1109/DSN-W50199.2020.00014

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). *P RUNING C ONVOLUTIONAL N EURAL N ETWORKS*. *2015*, 1–17.

Moreno, A. (2019). Autor: Álvaro Artola Moreno Tutor: José Antonio Pérez Carrasco. *Universidad de Sevilla*, 80. https://hdl.handle.net/11441/89506

Pérez Cerdeira, I. J. (2021). *Aceleración hardware para inferencia en redes neuronales convolucionales*.

Renza Torres, D., & Ballesteros, D. M. (2023). *Fundamentos de visión por computador utilizando aprendizaje profundo*. 1–166.

Ruospo, A., Bosio, A., Ianne, A., Sanchez, E., Torino, P., & Lyon, E. C. De. (2020). *Evaluating Convolutional Neural Networks Reliability depending on their Data Representation*. 672–679. https://doi.org/10.1109/DSD51259.2020.00109

Ruospo, A., Sanchez, E., Traiola, M., Connor, I. O., & Bosio, A. (2021). Microprocessors and Microsystems Investigating data representation for efficient and reliable Convolutional Neural Networks. *Microprocessors and Microsystems*, *86*(December 2020), 104318. https://doi.org/10.1016/j.micpro.2021.104318

Sarkar, S. (2020). *An Incremental Pruning Strategy for Fast Training of CNN Models*. 371–375.