Luís Alberto Pires Amaro

RESILIENT ARTIFICIAL NEURAL NETWORKS

1 2 9 0 UNIVERSIDADE D COIMBRA

1 2 9 0

UNIVERSIDADE D
COIMBRA

Luís Alberto Pires Amaro

RESILIENT ARTIFICIAL NEURAL NETWORKS

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Raul Barbosa and Professor Nuno Lourenço and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2020

Faculty of Sciences and Technology

Department of Informatics Engineering

Resilient Artifical Neural Networks

Luís Alberto Pires Amaro

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Raul Barbosa and Professor Nuno Lourenço and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

<09> <2020>

1 2 9 0

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

This thesis was supervised by Professor Raul Barbosa and Professor Nuno Lourenço at the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra and is part of the AI4EU project.

This page is intentionally left blank.

Abstract

Artificial Neural Networks (ANNs) are used daily to help humans solve complex problems in real-life situations. They are present in technology that is key to our society in the present, such as Credit card fraud detection, medical devices, and they will be present in technology that will be key to our society in the future, such as Autonomous Vehicles (AV). A system is considered safety-critical if in case of failure leads to a loss of life or substantial damage both in property or environment. There are currently a large number of Safety-Critical Systems (SCSs) that rely on ANNs. For this reason it is crucial to ensure that the ANNs have a level of robustness/resilience adequate to the fact that there are human lives that depend on the results produced by them.

In this work we have two main objectives. The first one is to study the impact that the Dropout technique has on the resilience/robustness of ANNs. The second one is to develop a new technique and study its impact on the resilience/robustness of ANNs. The new technique is named Stimulated Dropout. To achieve these goals we are going to train multiple neural networks with different dropout and stimulated dropout probabilities. The networks were trained and tested using two different databases. During the tests of these neural networks we will perform memory bitflips, and to understand the impact of both techniques in the resilience of ANNs, the results were analyzed using three parameters, the number of SDCs, accuracy, and training time.

Our results show that adding Dropout and Stimulated Dropout to the the neural networks will decrease the number of SDCs, which means that both techniques have a positive impact on the resilience/robustness of ANNs. For the Mnist database, when we have a network trained without any of the techniques, we have a percentage of SDCs of 6.77%, whereas with a dropout probability of 80% the percentage of SDCs will be 3.76%, about 45% less. While in the case of the Stimulated Dropout, the lowest percentage of SDCs happens when we have a probability of 20%, where we have a percentage of SDCs of 5.15% about 24% less. As for the Fashion Mnist database, when we have a network trained without any of the techniques, we have a percentage of SDCs of 7.93%, whereas with a dropout probability of 80%, we have a percentage of 4.8%, about 46% less. While in the case of the Stimulated Dropout, the lowest percentage of SDCs happens when we have a probability of 50%, where we have a percentage of SDCs of 4.22%, about 47% less.

In the case of Dropout, we observed that there is a tradeoff between the accuracy and the number of SDCs. This because, the lowest number of SDCs in both databases happens when we have a dropout probability of 80%, and if we compare the accuracy of the different dropout probabilities, we can see that the accuracy decreases as the dropout probability increases. Regarding the Stimulated Dropout, we observed that there is a tradeoff between the training time and the number of SDCs. This because, the training time of the networks trained with different stimulated dropout probabilities is far higher that the training time of the network trained without any of the techniques.

Keywords

Artificial Neural Networks, Dropout, Stimulated Dropout, Hardware Fault Injection, Silent Data Corruption

This page is intentionally left blank.

Resumo

Diariamente estamos dependentes de Redes Neurais Artificiais para resolver problemas complexos. Estas estão persentes na tecnologia que é crucial para a nossa sociedade no presente, tal como detecção de fraude de cartão de crédito, dispositivos médicos e estarão presentes em tecnologia que será crucial para a nossa sociedade no futuro, como por exemplo, veículos autônomos. Redes Neurais Artificiais são parte de Sistemas Críticos de Segurança, sistemas esses que têm várias vidas humanas que dependem deles. Um sistema é considerado crítico de segurança se em casa de falha levar à perda de vidas ou causar danos substanciais tanto em propriedades como no ambiente. Atualmente, há um grande número de Sistemas Críticos de Segurança, que dependem de Redes Neurais Artificiais. Por este motivo, é fundamental garantir que estas tenham um nível de robustez/resiliência adequado ao facto de existirem vidas humanas que dependem dos resultados produzidos por estes sistemas.

Neste trabalho, temos dois objetivos principais. O primeiro é estudar o impacto que a técnica de *Dropout* tem na resiliência/robustez das Redes Neurais Artificiais. O segundo é desenvolver uma nova técnica e estudar o impacto que esta tem na resiliência/robustez das Redes Neurais Artificiais. A nova técnica é chamada de *Stimulated Dropout*. De forma a atingir os objectivos propostos, foram treinadas várias redes neurais com diferentes probabilidades de *Dropout* e *Stimulated Dropout*. As redes foram treinadas e testadas usando duas base de dados diferentes. Durante os testes das redes neurais iremos realizar bitflips na memória, e de forma a entender o impacto de ambas as técnicas na resiliência das Redes Neurais Artificiais, os resultados foram analisados usando três parâmetros, o número de *Silent Data Corruptions* (SDCs), *accuracy* e o tempo de treino.

Os resultados mostram que o número de SDCs vai decrescer em redes treinadas com *Dropout* e *Stimulated Dropout*. Na base de dados *Mnist*, quando temos uma rede treinada sem nenhuma das técnicas, temos uma percentagem de *SDCs* de 6.77%, enquanto que com uma probabilidade de *dropout* de 80%, a percentagem é de 3.76%, cerca de 45% a menos. No caso do *Stimulated Dropout*, a menor percentagem de SDCs ocore quando temos uma probabilidade de 20%, onde temos uma percentagem de SDCs de 5.15%, cerca de 24% a menos. Quanto a base de dados *Fashion Mnist*, quando temos uma rede treinada sem nenhuma das técnicas, temos umas percentagem de SDCs de 7.93%, equanto que com uma probabilidade de 80%, temos umas percentagem de 4.8%, cerca de 46% menos. No caso do *Stimulated Dropout*, a menor percentagem de SDCs ocorre quanto temos uma probabilidade de 50%, onde temos uma percentagem de SDCs de 4.22%, cerca de 47% menos.

No caso do Dropout, observamos que existe um *tradeoff* entre a *accuracy* e o número de SDCs. Isto porque, o menor número de SDCs em ambas as bases de dados acontece quando temos uma probabilidade de *dropout* de 80%, sendo que a *accuracy* decresce à medida que a probabilidade de *dropout* aumenta. Relativamente ao *Stimulated Dropout*, observamos um *tradeoff* entre o tempo de treino e o número de SDCs. Isto porque, o tempo de treino das redes treinadas com diferentes probabilidade de *stimulated dropout* é consideravelmente maior do que o tempo de treino da rede treinada sem nenhuma das técnicas.

Palavras-Chave

Redes Neurais Artificiais, Dropout, Stimulated Dropout, Injeção de Falhas De Hardware, Silent Data Corruption

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

ANNs Artificial Neural Networks. v, 1–3, 5, 8, 13–15, 17, 19, 20, 23, 29, 32, 35, 37, 38

AV Autonomous Vehicles. v, 1, 5, 6

CNNs Convolution Neural Networks. 8

FI Fault Injection. 6

HWFI Hardware Fault Injection. 2, 6, 14, 15, 17

SCSs Safety-Critical Systems. v, 1, 5, 13, 37

SDC Silent Data Corruption. 7, 11, 15, 18

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

From medical devices to the mechanism that controls the plane we fly on, we are daily dependent on systems that rely on a computer. These are called Safety-Critical Systems (SCSs), and it is crucial to ensure that they have a level of robustness/resilience adequate to the fact that there are human lives that depend on them. Autonomous Vehicles (AV) are an example of SCSs, since a failure in them can lead to catastrophic damage. For this reason, one of the barriers to automated vehicles is their certification, as they have to have a level of robustness/resilience appropriate to the fact that there are human lives that depend on the results produced by the system. AV make use of Artificial Neural Networks (ANNs), and these networks are essential to the AV functioning, since they are responsible for detecting pedestrians and obstacles. Therefore, to ensure that these systems are resilient, we need to ensure that ANNs are resilient.

In a hypothetical scenario where we have an AV driving on the road with traffic and pedestrians on the sidewalk, what happens if there is a hardware failure? In such situations, there are several ethical questions that we have no answer to, namely which one would best prioritize the passenger life and ignore everything else or always choose the scenario that causes the least damage to human lifes. This is just an example of a scenario among many others, and in these types of circumstances, there is no correct answer since it involves the loss of human life. That is why we have to do as much as we can to prevent these situations from happening, and the only way to do it is to make our systems more robust/resilient.

AV depend on computers such as Tesla FSD [25] and Nvidia Pegasus [16]. A failure in these can have severe consequences since this hardware is used to perform tasks associated with neural networks. ANNs are loosely inspired by the human brain, both on the way they work and how they are structured. They are composed of small units called neurons, and these neurons are organized in layers. The neurons in a layer will be connected to the neurons in the following layer, where the output of some neurons is the input of others. As in the human brain, where a axon connects neurons, in the ANNs, neurons are also connected, allowing them to exchange information between them. There are several types of neurons, with different objectives. An example of a type of neurons is the dropout [24]. Dropout has been introduced as an overfitting prevention technique. That is, overfitting happens when the network adapts too much to the training data, which results in a model that does not generalize well.

Although dropout was initially presented as an overfitting prevention, we hypothesise that it can also improve the resilience of ANNs. In concrete, our main question is to see if a network trained with dropout units is more resilient than a network without dropout.

To achieve this we will train multiple neural networks with the same architecture. The difference between them lies in the fact that some networks will be trained with different dropout probabilities, and others will be trained without dropout. After training, all the networks will be tested with examples from the MNIST database and the Fashion Mnist database. During the tests, Hardware Fault Injection (HWFI) will be injected into the hardware, specifically, memory bitflips. A bitflip is merely changing a bit from 1 to 0 or 0 to 1. After performing the fault injection, the results will be analyzed, allowing us to understand whether the dropout influenced the resilience of the neural networks. Besides, we will introduce a new technique based on dropout, the stimulated dropout. Briefly, as mentioned before, neurons exchange information among themselves, and in the case of ANNs, this information is a real number. These real numbers will be the output of a neuron and the input of a neuron in the next layer. In the case of dropout there is a probability that this value will be replaced by 0. In the case of stimulated dropout instead of replacing this value with 0, we will do a bitflip of this value.

1.1 Objectives

In this thesis, we have two main goals, the first one is to explore dropout as a technique to increase the resilience/robustness of ANNs. The second one is to develop a new technique based on dropout and study its impact on the resilience/robustness of ANNs. This new technique is named stimulated dropout.

- To study the impact of dropout on the resilience/robustness of ANNs, we have to train and test multiple neural networks, more specifically four. These networks will have the same architecture, one of which will be trained without dropout, and the other three will be trained with different dropout probabilities, 20%, 50% and 80%. The neural networks will be trained and tested using two datasets, the MNIST dataset [15], and the Fashion MNIST dataset [27]. The next step will be to perform fault injection in the neural networks during the test phase, specifically memory bitflips. To do that we will use the ucXception tool [3], this tool is capable of emulating soft errors, a single bitflip, that affect CPU registers on a Linux-based operating system.

- Regarding the stimulated dropout technique, the first step will be to implement it. The main idea behind this technique is that instead of what happens in the dropout technique, where there is a probability of the output value of the neuron to be 0, in the case of the stimulated dropout technique, there is a probability of the output value being changed through a bitflip. After developing it, the process will be equal to what was done to study the impact of the dropout technique on the resilience/robustness of ANNs. We will train and test four networks with the same architecture, one of them trained without stimulated dropout and three of them trained with different probabilities of stimulated dropout, 20%, 50% and 80%. The networks will be trained and tested using two datasets, the MNIST dataset [15], and the Fashion MNIST dataset [27]. The last step will be to perform fault injection in the neural networks during the test phase. To do that we will use the ucXception tool [3], this tool is capable of emulating soft errors, a single bit-flip, that affect CPU registers on a Linux-based operating system.

We will analyze the impact that both techniques have on the resilience/robustness of neuronal networks, comparing the results of networks trained without dropout and stimulated

dropout, with the results of networks trained with different dropout probabilities and different probabilities of stimulated dropout. Also, the fact that we have different datasets helps us understand the effect that different datasets with varying levels of complexity have on the neural networks.

1.2 Contributions

The main contributions of this work are summarised below:

- Dropout as a technique to improve the resilience/robustness of ANNs. We also showed that different dropout probabilities have different impacts on the resilience of ANNs. The lowest number of SDCs happens when we have a higher probability of dropout. Being that when we have a dropout probability of 80%, it is also possible to observe an accentuated decrease in the accuracy. With this, it is possible to observe a tradeoff between the accuracy and the number of SDCs.
- We developed a new technique called stimulated dropout. This technique can be used to improve the resilience/robustness of ANNs. Different stimulated dropout probabilities have different impacts on the resilience/robustness of ANNs. We observed that the training time is far higher than the training time of ANNs trained with dropout or without any technique. With this, it is possible to observe a tradeoff between the number of SDCs and the training time.
- The number of SDCs is related to the complexity of the dataset. The number of SDCs in the results obtained using the Mnist dataset is lower than the number of SDCs in the results obtained using the Fashion Mnist dataset.

1.3 Document Structure

In the first chapter we introduced the problem addressed by this work and presented the main objectives. The structure of this thesis is organized as follows.

- Chapter 2: In this chapter, we give an overview of state of the art on Fault Injection, Safety-Critical Systems, and Deep Learning. Is also given a description of the tools used, as well as the ANNs implemented and the type of faults injected into these networks. There are two sections that are subdivided; Fault injection is subdivided into, Hardware Fault Injection, Soft Errors, and ucXception; Deep Learning is subdivided into, Convolutional Neural Networks, Dropout,Stimulated Dropout and Pytorch. The last section of this chapter serves as a complement to the previous ones, since the previous sections only described research that was directly related to the work, that is, that had an influence on the practical part. In this section will be presented articles that aim to improve the resilience of neuronal networks through various techniques.
- Chapter 3: This chapter presents the approaches taken to achieve the goals of this thesis. It details the experimental methodology, including the datasets, the architecture of the neural networks and the fault injection process.
- Chapter 4: In this chapter we present and discuss the results of the experiments performed during this thesis. We analyze the performance of dropout and stimulated

dropout using three parameters, namely the number of SDC's, accuracy and training time. This chapter ends with a reflection on the results obtained.

- Chapter 5: The document ends with a reflection about what was accomplished and discusses possible directions for future work.

Chapter 2

Background

We aim to study the impact that dropout has on the resilience of ANNs. To achieve that, we need to understand fundamental concepts in the areas of Fault Injection, Safety-Critical Systems and Machine Learning. In section 2.1, we explain why it is so important to ensure reliability in SCSs. In section 2.2, concepts related to Fault Injection are introduced. At the end of this section we give a brief overview of the tool used to inject faults. In section 2.3, we introduce concepts related to Deep Learning. In this section we will also introduce a new technique called Stimulated Dropout. At the end of section 2.3, we give a brief overview of the tool used to implement the neural networks. In section 2.4, will be presented articles that aim to improve the resilience of neural networks through various techniques.

2.1 Safety-Critical Systems

A system is considered safety-critical if in case of failure leads to a loss of life or substantial damage both in property or environment. There are currently a large number of systems that rely on a computer, and the number of such systems will tend to increase in the future. Within these systems, the ones that concerns the most are the safety-critical ones since a failure in them could lead to irreversible damage [12].

Safety-Critical Systems (SCSs) are present in multiple areas, some of which are crucial for our survival. From medical devices (e.g., patient monitors) to the computer that controls the planes, we are daily dependent on SCSs. That is why we need to ensure that those systems are reliable. The higher the dependability, the higher the reliability. For this reason, SCSs are developed under certain guidelines and according to safety standards. These standards are used to regulate the systems, ensuring user safety. The ISO-26262, " Road Vehicles – Functional Safety", is an example; it defines a car-specific international standard. It defines a common vocabulary and focuses on safety-critical components allowing the automotive industry to measure how safe their car are.

Self-driving cars, also known as AV, are cars capable of navigating without any human input. This description is only valid for level 5 of automation, according to the Society of Automotive Engineers (SAE) [20], since, on the other levels, there is human input. These cars depend on computers such as Tesla FSD (full self-driving) and Nvidia Pegasus. This hardware is used so that the system can detect pedestrians and avoid obstacles. These systems are SCSs since a failure in them can lead to irreversible damage, like what happened in 2018. During a test, an AV operated by Uber killed a woman in Arizona [26]. These

lead to a major setback since, after the incident, the safety measures that had been studied and document are now being reconsidered.

Even though this was an isolated incident, it raises some concerns, since society is far less tolerant of a death caused by a machine. After the initial hype, there was a deceleration in the technological development as there are several issues raised after the Arizona incident. This is referred to in a paper published roughly one year before the event [22], where the authors expose the fact that the automotive industry is more concerned about who will get the first AV on the road rather than trying to improve the technology. They focus on scalability, as it will be tough to fix AV at scale and safety assurance. For the success of AV, it is crucial to assure users that they are safe, so in [22] is proposed the Responsibility-Sensitive Safety (RSS). It is a mathematical model that formalizes the Duty of Care law.

2.2 Fault Injection

Fault Injection (FI) evaluates the dependability of the system, that is, " delivery of service that can justifiably be trusted, thus avoidance of failures that are unacceptably frequent or severe" [13] . The faults can be injected in both software (code or data corruption) and hardware (logical or electrical faults). Many times is challenging to identify the cause of an error/failure, particularly in large and/or complex systems. To understand it, we recreate the failure scenario injecting artificial faults into the system. After that, we monitor their effects [11].

A system consists of multiple components, and a failure in one of them can cause permanent or transient faults depending on their duration. The permanent faults can be both solid or elusive. Elusive development faults and transient physical faults can be grouped as intermittent faults [1], as we can see in figure 2.1.

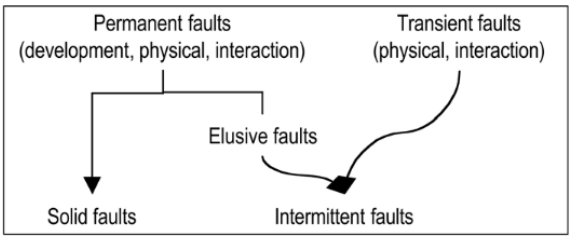

Figure 2.1: Solid versus intermittent faults [1]

Permanent faults stay present after their occurrence. They are easy to replicate since they always occur when a particular set of conditions exists. Transient faults appear for a short period and disappear after a new write masks them in the affected location, for example, a write to a CPU register. Intermittent faults appear, disappear, and reappear in a seemingly erratic manner [5].

2.3 Hardware Fault Injection

HWFI consists of injecting faults into the target system hardware. It uses external hardware to inject the fault, and thence there are no modifications and intrusion to the system under test. Although, there are some setbacks, the fact that we rely on the available debugging and testing features of the hardware, for example, setting a breakpoint [5].

There are two categories of hardware-implemented fault injection methods, hardware fault injection with contact and hardware fault injection without contact. In the first one, the injector has physical contact with the target system, while in the second one, there is no physical contact. Instead, it relies on an external source to produce some physical phenomenon [11].

After fault injection the application will produce an output, this output will be classified into three classes, as in [4].

- No Effect - the fault has zero impact on the output;
- Silent Data Corruption (SDC) - the application behaves normally but the output is erroneous;
- Crash/Hang - there is an error and the application closes unexpectedly.

2.4 Soft errors

As mentioned earlier, faults can be permanent, transient, and intermittent. In this work, we will be focusing on the last ones. Since they will appear and disappear randomly, it makes it hard to deal with this type of faults. The errors caused by intermittent faults are usually termed soft errors [1].

A soft error, is an error that will not cause significant damages to the system. It only affects the data that is being processed. They can be generated, for example, by a bit-flip in a CPU register. We can do a single bit-flip or multiple bit-flips, but as shown in [21], the percentages of SDCs are roughly the same for the single bit-flip model and the multiple bit-flip model.

Although the crash of the program is the one that causes the biggest damage, they are easy to detect, which makes that we can find ways to prevent it. The biggest problem comes from SDCs since although the system is apparently functioning normally, its output will be wrong. This will have a negative impact on the system reliability as there can be a catastrophic failure at any time [21].

The reliability of a system is crucial to its success, so you need to ensure that it is resilient to SDCs. The issue is that sometimes quality attributes such as performance often outweigh the reliability of a system, so it is necessary to figure out ways to improve the reliability of the system without compromising other attributes.

Choosing a suitable programming language is a viable solution to this problem. As shown in [4] non-compiled languages (Java, JavaScript, and Python) are more resilient against SDCs than compiled languages (C, C++, Rust, and Haskell).

2.5 ucXception

To do the fault injection, we are going to use a tool developed by Frederico Cerveira, ucXception [3]. This tool's choice is because it is capable of emulating soft errors, a single bitflip, affecting the CPU registers in Linux-based operating systems. In addition to this, it was developed by a CISUC researcher, Frederico Cerveira, which made it easier to learn how to use it.

The fault injection tool "takes advantage of the Linux kernels ptrace functionality to suspend the execution of a running process, extract its register values, perform a bit-flip and resume execution" [4]. Briefly, there will be injected faults, in this case bitflips, in twenty-eight memory registers. Each one of those twenty-eight has sixty four bits, so for each simulation will be made one thousand seventy ninety two bitflips.

2.6 Deep Learning

Learning from our experiences is part of growing as a human being. Throughout our lives, we learn, improve in many aspects, due to past experiences and Machine Learning is the same. A computer can learn to do a certain task without being programmed for it, just based on examples given to it. This allows computers to learn by themselves without human intervention, just by analyzing the given data and founding patterns in it, allowing the computer to make a decision based on the examples provided.

Deep Learning is a subfield of Machine Learning that uses Artificial Neural Networks (ANNs), which aim to mimic the behavior of the human brain, both in the way it functions and in the way it is structured. Deep learning, in comparison with traditional Machine Learning algorithms, has better performance if the dataset size is large, but it requires more computational resources to train in a reasonable amount of time. Another benefit is the ability to extract features from raw data automatically.

Deep learning has led to a technological leap forward in the artificial intelligence community, making it possible to solve problems that seemed impossible to solve. In order to understand it, an example adapted from [14] will be given. Suppose we have an image, which is represented through an array of pixels. In the first layer, the network learns particular features, such as the presence or absence of edges at certain locations in the image. The second layer finds specific arrangements of edges. The third layer groups the arrangements found in the previous layer. These groups correspond to parts of familiar objects. Later layers will combine these parts to detect objects. The main point here is that in deep learning, these layers require no human intervention; they are learned from data [14].

2.7 Convolutional Neural Networks

As said before ANNs are used to mimic the human brain behaviour. Convolution Neural Networks (CNNs) are a type of ANNs that are used in image recognition. The structure of the CNNs are also inspired in the human brain, in this case in the visual cortex, the part of the brain that is responsible for processing human visual stimuli. CNNs consist of the input layer, the output layer, and between those there are the hidden layers. The hidden layers can be multiple and of different types, convolutional layers, normalization layers,fully connected layers and pooling layers.

The critical aspect of CNNs is the convolutional layers since instead of this layer being connected to every single pixel in the image, it is only connected to the pixels in their receptive fields. The next layer will only be connected to the pixels within the rectangle produced by the first layer. The fact that instead of receiving pixel by pixel as input and just focusing on the features of the image will speed up the process, furthermore with this architecture, as we move through the hidden layers we can see that the neural network will only focus on the higher-level features, for example, the first hidden layer will take into

account low-level features which will together be the input of the second layer making the next hidden layer take into account higher-level features than previous layers [9].

In figure 2.2 we have the neural network used in this work. Briefly, in the first stage we have four layers two convolution layers each one followed by one pooling layer and at the second stage we have three linear layers.

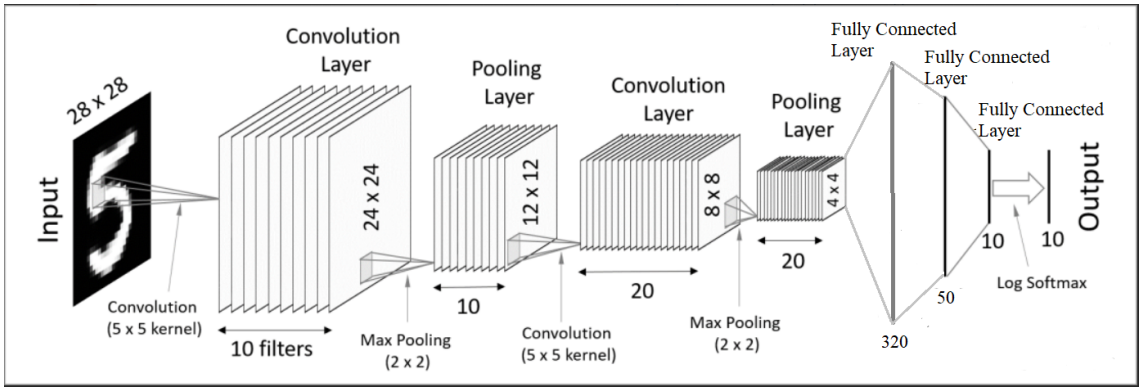

Figure 2.2: Convolutional Neural Network used for handwritten digit classification [23]

As input, we receive a 28*28 figure that presents in the MNIST database [28]. As mentioned before, instead of going pixel by pixel, we extract features of the image with a five by five kernel is used. This kernel is used to extract specific features of the input image. In practice, the kernel is going to be a five by five matrix that will be multiplied by the input, also a five by five matrix, it will generate an output that will be given as input to the next layer, thereby enhancing certain parts of the image that are more interesting rather than doing a thorough search for it. In order to extract the features from the whole image, we will multiply the kernel by several arrays of the same size, and the values in these arrays will vary according to the position. This position will depend on the stride, for example, if we have the stride with a value of one, and we are multiplying the kernel by an array that originates from position zero to position five in the next iteration we will multiply the kernel by an array originating from position one to position six. In the first convolution layer, we are going to have ten filters (kernel), this will highlight some features of the input images, as an edge. Each filter it's applied to a subregion of the input image. In this case the input is a five by five matrix (the size of the kernel). Hereupon each convolution layer is the matrix of the output of the set of filters.

The pooling layers, in this case, max-pooling layers, will divide the input matrix into a two by two matrix and compute the maximum of each region. This will help to extract higher-level features that the ones extracted in the previous layers. In the pooling layers, we have the same amount of filters as in the previous convolutional layers. In the end, we will have a 4 * 4 output that is going to the input of the following layer. The final layers are fully connected layers; in this case all neurons of the layer are connected to all the neurons of the previous layer. The output of a layer is the result of the multiplication of the link added to the bias. At the one, there will be ten classes; each one represent a number from zero to nine. A probability is associated with every neuron, the one with the highest value is the output, the sum of the probability of the neurons is equal to one. To get these values the softmax function was used to map the non-normalized output into the output classes.

2.8 Dropout

One of the main problems with neural networks is overfitting, that is, the network adapts too much to the training data, making network performance optimal for the training data but decreasing for new data. To address this problem, dropout has been introduced as an overfitting prevention technique in [24].

In figure 2.3 we have the dropout neural net model on the left side we have a network with two hidden layers and at the right side we have a thinned version of the network after the dropout model was applied. Briefly, if we have a layer that has a dropout probability of 0.5, this means that all neurons in that layer have a 50% probability of not being present during a given training iteration. That said, instead of training a complete network we will train smaller networks in each iteration.

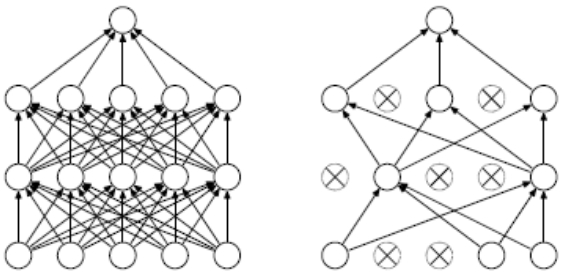

Figure 2.3: Dropout Neural Net Model [24]

There are several benefits that come from using dropout, reduces overfitting and improves other regularization methods. In addition there is also an improvement in the performance neuronal networks on supervised learning in different tasks such as vision, speech recognition, document classification and computational biology as proved in [24].

2.9 Stimulated Dropout

As mentioned earlier, if we have a certain probability of dropout, for example, 50%, associated with a neural network layer, all neurons of that layer are 50% likely not to be present during a given training iteration. Before introducing stimulated dropout, it is essential to understand how dropout works in practice. In a neuronal network, neurons of different layers are connected. The neurons of a layer will be connected to the neurons of the following layer, and the output of a neuron will be the input of a neuron in the next layer. When it is determined that a neuron will not be present, in practice, what happens is that the output of this neuron will be 0, and with this, the neuron will not have any impact.

Neurons of different layers exchange information with each other, the output of a neuron is the input of other neurons, this information is a real value. In the case of dropout, there is a probability that this value will be replaced by 0. In the case of stimulated dropout instead of replacing this value with 0, we will do a bit-flip of this value. This value can be a 64 bit IEEE 754 double precision floating point number, or a 32 bit IEEE 754 single precision floating point number. With stimulated dropout, we will emulate errors when training the neural networks. With this we intend to increase the resilience/robustness of the neural networks, thus reducing the number of SDCs during their testing.

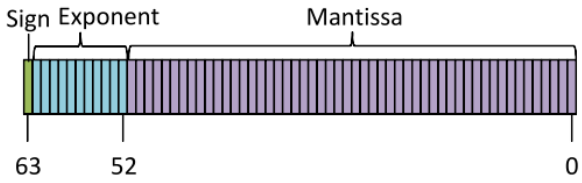

Figure 2.4: Double Precision IEEE 754 Floating-Point Standard [8]

The 64 bit IEEE 754 depicted graphically in figure 2.4, where 52 bits represent the mantissa, 11 bits represent the exponent, and one bit represents the sign. Bit 0 being the least significant of the mantissa and bit 52 the most significant. Bit 53 will be the least significant of the exponent, and bit 62 will be the most significant. The more significant the bit is, the greater the change on the number will be. The bits of the exponent are more significant than the bits of the mantissa, meaning that if we flip a bit in the exponent is will have a more substantial impact on the number than if we flip a bit in the mantissa. But the most significant change will happen when we flip bit 63 as it will change the sign of the value. In a first approach, we choose a random flip a random bit 0 and 63. Still, after several experiments, we realized that if the random bit were the bit 63, the impact it would have would be too significant, since the accuracy would undergo a significant change. In both databases, Fashion Mnist and Mnist, the accuracy would decrease to around 10%.

With this, we will only flip bits between 0 and 62, since we never flip the bit 63 during the training. This choice will not impact the number of SDCs, since even while testing the bit 63, it has no greater impact than other bits. Since any change in these bits makes such a difference too large in the output. This will cause the program to detect the error, and the output will be classified as Crash.

There is also a probability associated with stimulated dropout. For example, if the probability is 50%, there is a 50% chance that we flip a bit in the value. This does not mean that this value has changed since if the bit is of the mantissa usually it has no impact on the value. We decided to flip any bit between 0 and 62, even if it belongs to the mantissa because we observed during the tests that if we flip any bit, regardless if its a bit of the mantissa of the exponent, it is capable of resulting in a SDC.

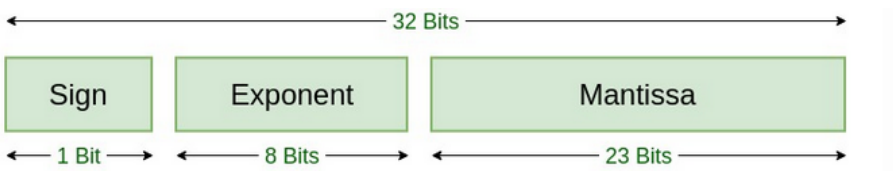

Figure 2.5: Single Precision IEEE 754 Floating-Point Standard [17]

The 32 bit IEEE 754 depicted graphically in figure 2.5, where 23 bits represent the mantissa, 8 bits represent the exponent, and one bit represents the sign. Bit 0 being the least significant of the mantissa and bit 22 the most significant. Bit 23 will be the least significant of the exponent, and bit 31 will be the most significant. As was the case in the case of 64 bits, here we also observed that if the random bit were the bit that represents the sign, the impact it would have be be too significant. For this reason, we also never flip the bit 31. With this, in the final version of stimulated dropout we flips bits 0 and 30, and between 32 and 62. In [19], the authors developed an algorithm that had the most significant negative impact on the accuracy with the smallest number of bit-flips. Being that here the authors

also concluded that flipping random bits could overwhelm the functionality of the neural networks. Citing the authors, "flipping the bits in exponent part of floating-point value can increase the weight to an extremely large value, this leading to the exploded output". In our case, we only don't flip the bits that represent the sign. When we are doing certain operations with floating-point numbers, as in our case, certain operations end up being invalid, and sometimes the value turned out to be NaN (Not a number). The value Nan is used to represent a value that does not represent a real number. NaNs was introduced by the IEEE 754 floating-point standard in 1985 and have the following format, the sign is either 0 or 1, exponent its all 1 bits, and the fraction can be anything except all 0 bits (all 0 bits represents infinity). According to the IEEE standard, comparisons involving NaN values are always false. So one way one to solve this problem was trough a comparison, value != value, its only true if the value its Nan.

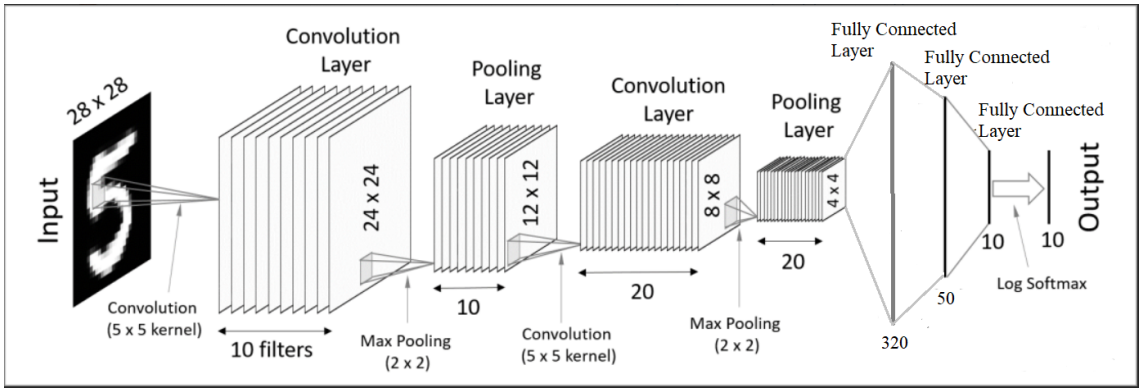

Figure 2.6: Neural Network Architecture [23]

The training time was a big problem, since in a first approach we changed every output of each neuron of the neural network. From the outset, we realized that it wasn't a reliable approach since the training time will be to high. To understand why the training time was so high lets look at the architecture of neural network 2.6. With this, the number of values that we would change at each training iteration, it would be equal to the Batch Size * Number of filters * Size of the Layer for the convolution layers and it would be equal to the Batch Size * Size of the Layer for the fully connected layers. We choose a batch size equal to 64, since it would ensure us better accuracy and less training time. With the architecture represent in 2.6 we would have:

- Convolution Layer 1 - 64*10*24*24 = 368 640 values
- Convolution Layer 2 - 64*20*8*8 = 81 920 values
- Fully connect Layer 1 - 64*320 = 20 480 values
- Fully connect Layer 2 - 64*50 = 3 200 values

This is the total number of values at each training iteration. Each database has a training set of 60,000 examples, with a batch size of 64 this results in 938 training iterations for each epoch, being that we have 10 epochs. For this reason it was not possible to change each one of these values, so we decided to change a number of random values. In the final version of stimulated dropout, we flip a bit in one third of the outputs of each neuron in the convolution layers and in half of the outputs of each neurons in the fully connected layers. This allows to reduce the training time, while maintaining consistence in the results. Briefly, in the final version of stimulated dropout, we flip a bit in:

- Convolution Layer 1 - 122 880 random values
- Convolution Layer 2 - 27 307 random values
- Fully connect Layer 1 - 10 240 random values
- Fully connect Layer 2 - 1 600 random values

2.10 Pytorch

As shown in [4], non-compiled languages (Java, JavaScript, and Python) are more resilient against SDCs than compiled languages (C, C++, Rust, and Haskell). There are multiple machine learning frameworks, but because the choice of a programming language is relatively restricted since python is more resilient to SDCs than C/C++, the decision was Pytorch since it is one of the only frameworks with a C++ frontend [18]. Pytorch only had a python frontend since it is the most common language when it comes to machine learning. Although Pytorch was built on a C++ backend, the C++ frontend was only added roughly two years ago. Since we are using something relatively new and there is not much documentation, it will cause some delay as the learning curve will be longer compared to using python.

The base code was taken from [10]. This is a repository containing c++ examples provided by Pytorch. In figure 2.2, we can see the neuronal network produced by this code. Some changes were made to the base code. The two main ones were the fact that in our case we split this class into two, one that consists only of training the neuronal network and second only the test of the same, this is because in our case it is only relevant to study the impact of failures on the test case. The second change was that we introduced dropout and stimulated dropout on both convolution layers.

2.11 Related Work

As deep learning systems are being widely adopted in SCSs, malicious attacks could lead to irreversible damage. It is crucial to find techniques that improve the resilience of these systems. Although I have not found other works that use dropout with the same purpose, different techniques are used to enhance the resilience of ANNs. In this chapter, will be presented some articles that served to demonstrate that ensuring the resilience of ANNs is pivotal in SCSs.

In self-driving vehicles it is important to guarantee that the system is not vulnerable against noisy or even maliciously manipulated sensory inputs. In [6], this problem is addressed, and is proposed a solution to increase the resilience of the ANNs. Regarding the noisy sensory inputs the authors solve that by defining a maximum amount of input that is tolerated. To find the sensor perturbation that is still tolerated, perturbation is added to the input, a image from the MNIST database, until the classifier gives the wrong output. If the image was classified to be five after the perturbation the same image is likely to be classified as a zero or a three but unlikely to be a five. After this process is done the authors find the smallest possible perturbation that makes the classifier loose confidence. Although this work is not directly related to ours, the goal is the same improve the resilience of ANNs. In our case we use dropout to achieve this goal and in this case the authors define a maximum amount of input that is tolerated. This is interesting since this solution do

not change the architecture of the neural network, this means that we could use the two techniques simultaneously to see the impact that it would have on the resilience of ANNs.

In [2] the authors do physical fault injection attack on Deep Neural Networks, more specifically, they performed practical laser fault injection using a diode pulse laser in order to disturb the circuits. In their study they target four activation functions: Rectified Linear Unit (ReLU), softmax, sigmoid and tanh. The goal of this study was to achieve misclassification, by attacking these four activation functions during the testing phase. At the end the authors conclude that the attacker to achieve a success rate superior to fifty percent the attacker should inject faults in at least 75% of the neurons in a layer for ReLu and in at 50% of the neurons in a layer for sigmoid and tanh. In the case of the softmax, after the fault injection there were two possible output: either the output contained invalid values or there was no output. The authors conclude that introducing faults into softmax is harder compared to other functions, since there was a lack of valid outputs. With this article the authors demonstrate that by injecting faults into the hidden layers of the network we can achieve a misclassification. The neural network used in this study had no dropout, would be interesting to study the impact that dropout would have on this results.

Currently there is a high demand for hardware optimized implementations of machine learning models. For this reason, companies focus on making cost-effective hardware and neglecting hardware security. This is addressed in [7] where the authors made an framework for inserting malicious hardware trojans, in order to make known the seriousness of this problem. The authors used the MNIST and CIFAR-10 dataset as inputs, and inserted the faults into a convolutional neural network. At the end the authors discuss potential defenses against the hardware Trojan attack on neural networks. The results presented in the article validated the premise that factories prioritizing efficiency can lead to hardware security flaws. This results validate the theme of our work, since as demonstrated in the article these systems are vulnerable to HWFI

Chapter 3

Approach

The main goal of this work is to study the impact of dropout and stimulated dropout on the resilience of ANNs. To achieve it, we trained seven similar ANNs. All of them had the same architecture, the only difference was the introduction of dropout and stimulated dropout in the layers. With this, we have one that was trained without dropout or stimulated dropout, three trained with different probabilities of dropout, and three trained with varying probabilities of stimulated dropout. Each network was trained with examples from the MNIST database [15] and the Fashion Mnist database [27]. Both databases have a training set of 60,000 examples and a test set of 10,000 examples. During the tests, we use the ucXception tool to perform HWFI, more specifically bitflips in the CPU registers. The results were grouped into three classes:

- No Effect - the fault has zero impact on the output;
- SDC - the application behaves normally but the output is erroneous;
- Crash/Hang - there is an error and the application closes unexpectedly;

3.1 Databases

In this works we are going to use the Mnist databse [15] and the Fashion Mnist database [27].

The Mnist database is a database of handwritten digits. It contains 60,000 training images and 10,000 testing images. These images are white and black 28*28 images. The Mnist database is extremely common in Machine Learning, which is usually the first example that people try to solve when introduced to Machine Learning. Briefly, you are given a 28*28 image of an handwritten digit, and your model have to classify it into one of ten classes (0,1,2,3,4,5,6,7,8,9). We have chosen to use this database for various reasons, from the outgiven, the fact that we can visualize the problem helps us understand it in a much faster way. The database is also very big, and it is well divided into training sets and testing sets, and there are any missing values or wrong labels, which make it easier since we don't need to preprocess any of the data. Finally, you can build a simple model and get high accuracies.

The Fashion Mnist database is very similar to the Mnist database, but instead of handwritten digits, in this case, we have clothing. As in the Mnist database, the Fashion Mnist database also contains 60,000 training images and 10,000 testing images. These images

are white and black, 28*28 images. The fact that both datasets share the same image size and structure of training and testing splits made it easier to implement the Fashion Mnist dataset into the already developed model. Comparing both databases, Fashion Mnist and Mnist, the first is a bit more challenging to train a model to, while in the second you could build a simple model and get accuracies as high as 99 percent, that's not going to be the case with Fashion Mnist, it's going to be more challenging to get high accuracy.

In this work we will use datasets with different levels of complexity, with the same model we will have higher accuracies in the Mnist dataset in relation to the Fashion Mnist dataset. With this we will be able to observe if with different datasets we will have different results or if these will be similar regardless of the dataset used. In other words, whether the dataset will affect the total number of sdc's, or on the other hand if different datasets will affect the impact that sdc's have on accuracy. Besides, this will help us validate the results since these will be obtained from two datasets instead of just one, which will bring more consistency to the results.

3.2 Convolutional Neural Networks

To implement the neural networks, we used Pytorch C++ Frontend. As the neuronal network architecture will be the same in all cases, it was important to ensure that this has good accuracy in both datasets. For this reason, we decided to use an already tested model with good accuracy on both datasets. With this model, we had an accuracy of 98.68% for the Mnist dataset and 88.31% for the Fashion Mnist dataset.

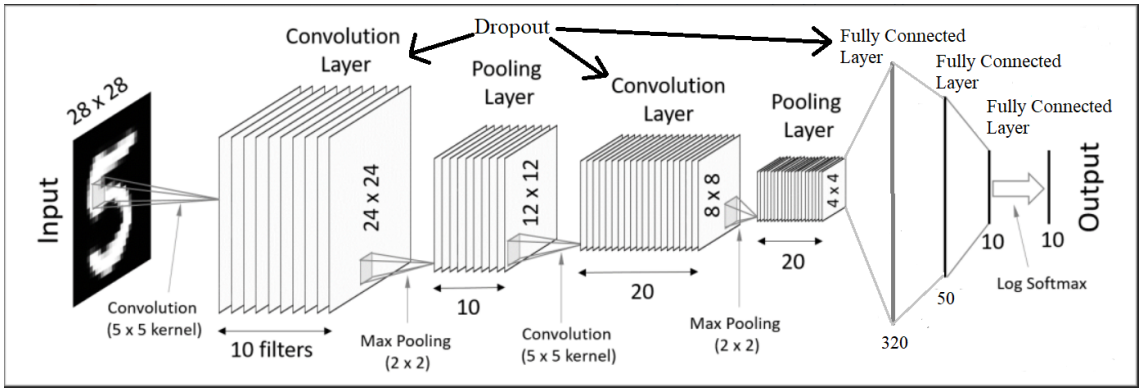

Figure 3.1: Convolutional Neural Network with Dropout[23]

The next step was to add dropout to the layers of the neural network, as shown in figure 3.1. The dropout probability will range between four values 20%, 50%, and 80%. If we have a dropout probability of 0%, there will be no change concerning the initial model. If we have a dropout probability of 20%, all the layers will have 80% of neurons present during a given training iteration. If we have a dropout probability of 50%, all the layers will have 50% of neurons present during a given training iteration. If we have a dropout probability of 80%, all the layers will have 20% of neurons present during a given training iteration.

The last step was to add stimulated dropout to the layers of the neural network, as we can see in figure 3.2. The stimulated dropout probability will range between four values 20%, 50%, and 80%. If we have a dropout probability of 0%, there will be no change concerning the initial model. If we have a stimulated dropout probability of 20%, there is a 20% chance that we flip a bit in the output value of all the neurons in each layer. If we have

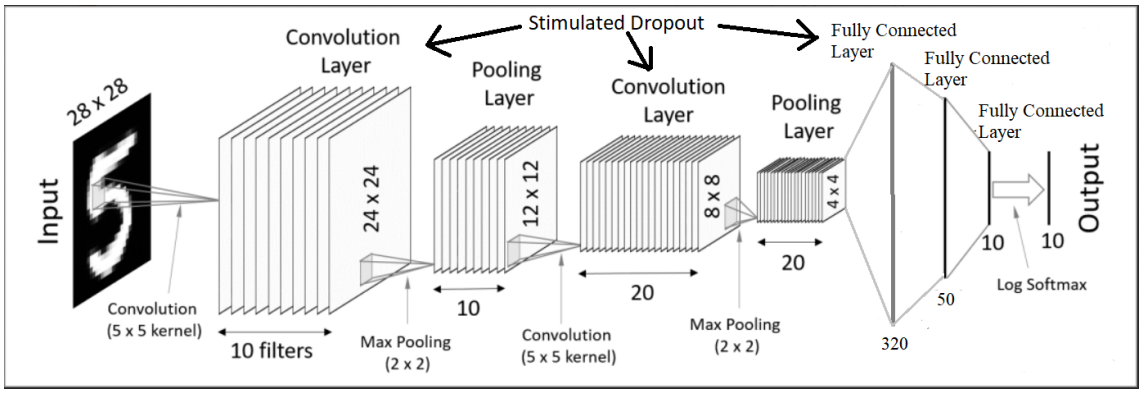

Figure 3.2: Convolutional Neural Network with Dropout Stimulated Dropout[23]

a stimulated dropout probability of 50%, there is a 50% chance that we flip a bit in the output value of all the neurons in each layer. If we have a stimulated dropout probability of 80%, there is a 80% chance that we flip a bit in the output value of all the neurons in each layer.

3.3 Fault Injection

After all the networks were implemented and trained the next step was to test them. While the networks are being tested, the memory registers are being used since it is where the information is stored. Our objective is to change the information that is stored while the networks are running. With this, we pretend to emulate a real scenario, where we have a ANNs running in a computer (e.g., Tesla FSD and NVidia Pegasus), and some malicious entity tries to alter the normal behavior of the ANNs, through HWFI.

Table 3.1: Example of Register Values

Register	**Value**	**Register**	**Value**
RIP	00007f6b1e3727a6	RSP	00007ffdfedba5f0
RBP	00007ffdfedba7b0	RAX	00000000ffffffe
RBX	0000000000000006	RCX	0000000002998100
RDX	0000000000000004	R8	00000000022cs560
R9	00000000022cs500	R10	00000000022b1d48
R11	00007f6b1c5f7050	R12	0000000000000006
R13	00000000007346c0	R14	0000000000000006
R15	00007ffdfedba7d8	RSI	0000000000000006
RDI	0000000000000006	CS	0000000000000033
SS	000000000000002b	DS	0000000000000000
ORIG_AX	ffffffffffff02	FS_BASE	00007f6b2a67d480
GS_BASE	0000000000000000	ES	0000000000000000
FS	0000000000000000	GS	0000000000000000
EFLAGS	0000000000000286		

Since the neural networks process the test sets of both databases quickly, each neural network will be running in a cycle, while we perform HWFI. With it, we will perform a bitflip in each one the 64 bits present in each of the 27 CPU registers. In table 3.1, we have an example of register values during the test of a neural network. As we can see,

we have 27 registers, and the neural network is using some of them. If we flip the bit 0 in the register RIP, the value of this register will be changed to 00007f6b1e3727a7. This small change can impact the final accuracy, thus jeopardizing the integrity of the neural network.

To bring more consistency to the results, each bit is going to be flipped five times. This means that we will perform five bitflips in each of the 64 bits present in each of the 27 CPU registers. In total, we are going to perform 8640 bitflips in each neural network. This will bring more consistency to the results since the final output always depends on the moment on when the fault is injected. Each output is going to be classified as follows:

- No Effect - the fault has zero impact on the output;
- SDC - the application behaves normally but the output is erroneous;
- Crash/Hang - there is an error and the application closes unexpectedly.

If the output is classified as No Effect, the accuracy was the same, before and after the fault injection. If the output is classified as SDC, the accuracy has changed, and if the output is classified as Crash/Hang, the program either detected an error or closed unexpectedly. The problem is that now each bit is going to have potentially five different outputs since the output can be different each time we flip a certain bit. To solve this, we will group the five different outputs into a single output, as follows:

- No Effect - All five outputs were classified as No Effect;
- SDC - At least one of the five outputs were classified as SDC;
- Crash/Hang - At least one of the five outputs were classified as Crash/Hang and none of the outputs were classified as SDC.

If at least one of the five outputs were classified as SDC, this means that at a given moment, a single bitflip can change the output without the system detecting it. It is crucial to identify if the system is vulnerable to SDCs since this will undermine the system's reliability as there can be a failure at any moment, without being detected. That is why it is irrelevant if the other four outputs were classified as No Effects, or Crash/Hang. It is also essential to prevent errors from happening while the program is running. That is why it is also important to identify if any of the outputs were classified as Crash/Hang. With this, we can determine if a single bitflip can question the integrity of our system.

Chapter 4

Results and Discussion

We present and discuss the results of our work in this chapter. To study the impact that dropout and stimulated dropout, had on the resilience/robustness of the ANNs, we will analyze the results using different parameters, namely:

- Number of SDCs - Percentage of outputs classified as SDCs;
- Accuracy - The impact that the SDCs have on the accuracy;
- Training Time - Time it took to train the neural network;

As mentioned before, the outputs were classified into three classes, No Effect, Crash/Hang, and SDCs. Here we will observe the impact that each technique had on the percentage of outputs classified has No Effect, Crash/Hang, and SDCs. Being that we will focus mainly on the number of the SDCs since they represent the most significant problem to the resilience/robustness of a system.

Regarding the accuracy, we will see if a neural network trained with a dropout probability of 50% has a similar accuracy has a neural network trained with a stimulated dropout probability of 50%. In addition to that, we will see the impact that the SDCs had on the accuracy. Has mentioned before, an SDCs will modify the output without the system noticing it. In this case, that means that the accuracy will be changed. We will focus on how big that change was when in comparison to normal accuracy.

Finally, we will analyze how much time it took to train each one of the neural networks. In most cases, there is a trade-off between accuracy and training time. That is, the longer the training time, the higher the accuracy will. In this case, we will see if this trade-off holds and if there is a new trade-off between longer training time and a smaller number of SDCs.

4.1 Dropout

Table 4.1: Summary of the Experimental Setup for Dropout

Dataset	Dropout Probability	Parameter
Mnist	Dropout 0%	Number of SDC's
		Accuracy
		Training Time
	Dropout 20%	Number of SDC's
		Accuracy
		Training Time
	Dropout 50%	Number of SDC's
		Accuracy
		Training Time
	Dropout 80%	Number of SDC's
		Accuracy
		Training Time
Fashion Mnist	Dropout 0%	Number of SDC's
		Accuracy
		Training Time
	Dropout 20%	Number of SDC's
		Accuracy
		Training Time
	Dropout 50%	Number of SDC's
		Accuracy
		Training Time
	Dropout 80%	Number of SDC's
		Accuracy
		Training Time

In table 4.1 is shown a summary of all the experiences done to study the impact that dropout has on the parameters mentioned before. This set of experiences will allow us to see the effect that dropout has on each parameter. More specifically, this will allow us to study the impact that different dropout probabilities have on the number of SDCs. If with an increasing dropout probability, we will get a higher or lower number of SDCs. In addition, this set of experiences will also allow us to study if the impact of SDCs has on the accuracy varies with different dropout probability and if different dropout probabilities have similar training times. With this set of experiences, we can see the impact that dropout as on each parameter, and consequently, we can see whether or not dropout has an impact on the resilience/robustness of ANNs.

4.1.1 Number of SDC's

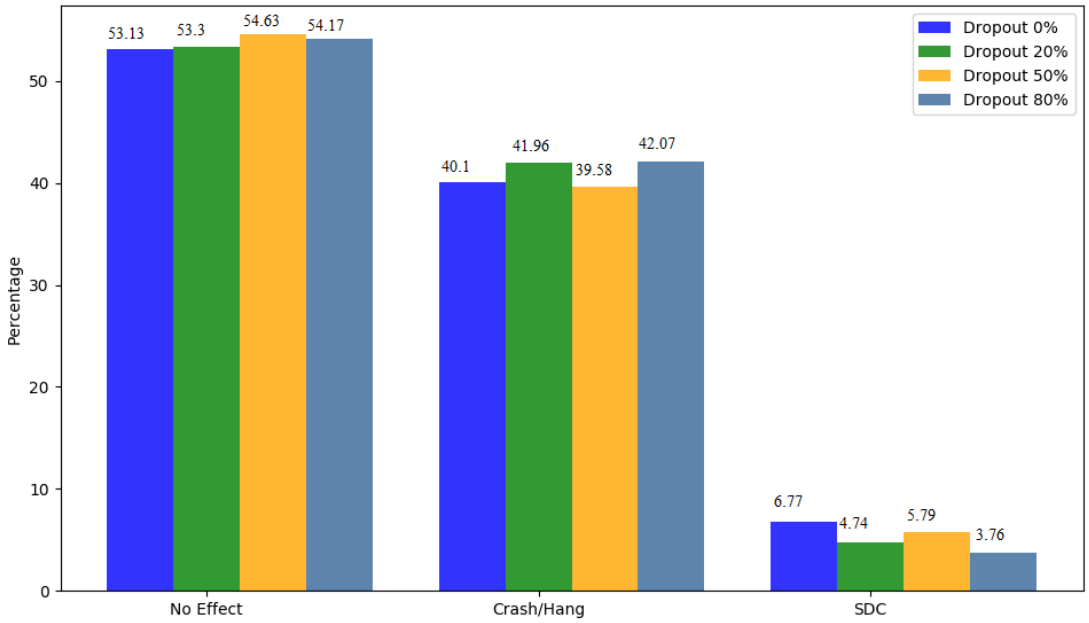

Figure 4.1: Neural Networks trained with Mnist database and different Dropout dropout probabilities

In figure 4.1, we can see the impact that each dropout probability has on the percentage of each output. From the outset, we can see that the number of SDC's is lower in networks trained with dropout than the network trained without dropout. Being that, when we have a dropout probability of 80%, we have a percentage of SDC's of 3.76%, 45% less than when we have a dropout probability of 0%. In the case of a neural network trained with dropout, the highest percentage of SDC's happens when we have a dropout probability of 50%, which is 1% less than the percentage of SDC's obtained in a neural network trained without dropout. With these results, we were able to verify that the dropout technique has an impact on the number of SDC's since the neural networks trained with dropout will always have a smaller number of SDC's, than the network trained without dropout. As for the different dropout probabilities, we can observe that with a dropout probability of 50%, we have a higher percentage of SDC's when compared to the other probabilities, and the lowest percentage of SDC's happens when we have a higher dropout probability.

Regarding the percentage of No Effects, we can see the results vary about 1% between the different dropout probabilities. We have a higher percentage of No Effects with a dropout probability of 50% and a lower percentage of No Effects with a dropout probability of 0%. As for the percentage of Crash/Hang, we can see that the highest percentage of Crash/Hang happens when we have a lower percentage of SDC's when we have a dropout probability of 80 %. The smallest percentage of Crash/Hangs occurs when we have a neural network trained with a dropout probability of 50 %.

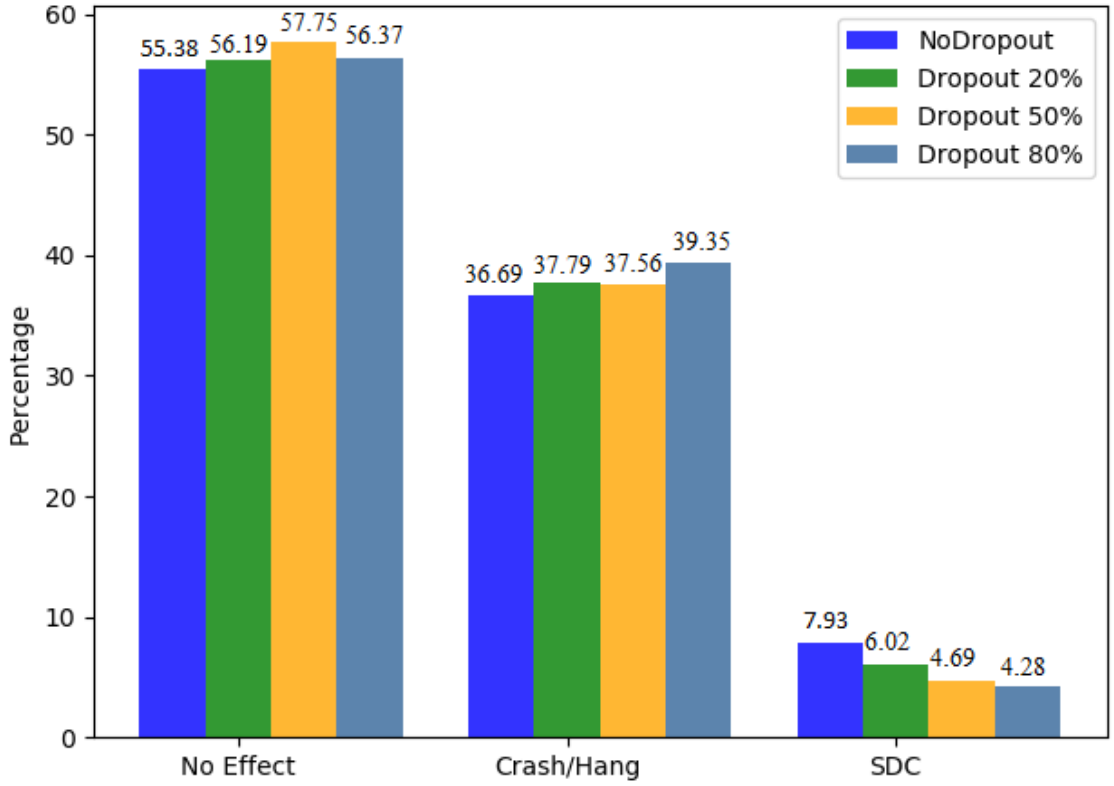

Figure 4.2: Neural Networks trained with Fashion Mnist database and different Dropout dropout probabilities

In figure 4.2, we can see the impact that each dropout probability has on the percentage of each output. As in the case of the Mnist database, here we can also see that the number of SDC's is lower in neural networks trained with dropout than the network trained without dropout. Regarding the number of SDC's, we can see that the results are similar to the ones obtained using the Mnist database. The lowest percentage of SDC's happens when we have a dropout probability of 80%, we have a percentage of 4.8%, about 46% less than when we have a dropout probability of 0%. The highest percentage of SDC's when a network is trained with dropout happens when we have a dropout probability equal to 20%, which is 2% less than when we have a trained network without dropout. The difference between the results obtained in the figure 4.1 and the results obtained in the figure 4.2 is that in this case the percentage of SDC's decreases as we increase the dropout probability.

As for the number of No Effects, we can see the results vary 2% between the different dropout probabilities. We have higher percentage of No Effects with a dropout probability of 50% and a lower percentage of No Effects with a dropout probability of 0%, as happened in the case of the Mnist database. Regarding the number of Crash/Hang, as in figure 4.2, we can see the the highest percentage of Crash/Hang happens when we have a lower percentage of SDC's when we have a dropout probability of 80%. The smallest percentage of Crash/Hang occurs when we have a neural network trained with a dropout probability of 0%.

If we compare the results obtained in both databases, Mnist and Fashion Mnist, we can see that there are some similarities but there are also two major differences. The first one is that in the case of the Fashion Mnist database, the percentage of SDC's decreases as the probability of dropout increases while in the case of the Mnist database we had a higher

percentage of SDC's when we had a 50% dropout probability than when we had an 20% dropout probability. The second one is that the number of SDC's increases when we used the Fashion Mnist. Being that, from these results we can conclude that with the increase of the complexity of the database, there was an increase in the number of SDC's.

The results presented in figures 4.1 and 4.2 demonstrate that a network is trained with dropout, regardless of the probability of dropout, will have a lower number of SDC's than a network trained without dropout. With these results, we can say that the dropout is not only an overfitting prevention technique, but it also can be used as a way to increase the resilience/robustness of the ANNs.

4.1.2 Accuracy

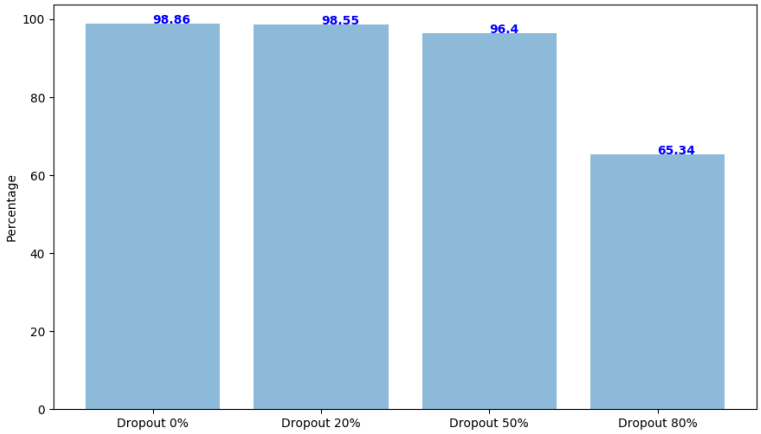

Figure 4.3: Accuracy of Neural Networks trained with Mnist database and different Dropout dropout probabilities

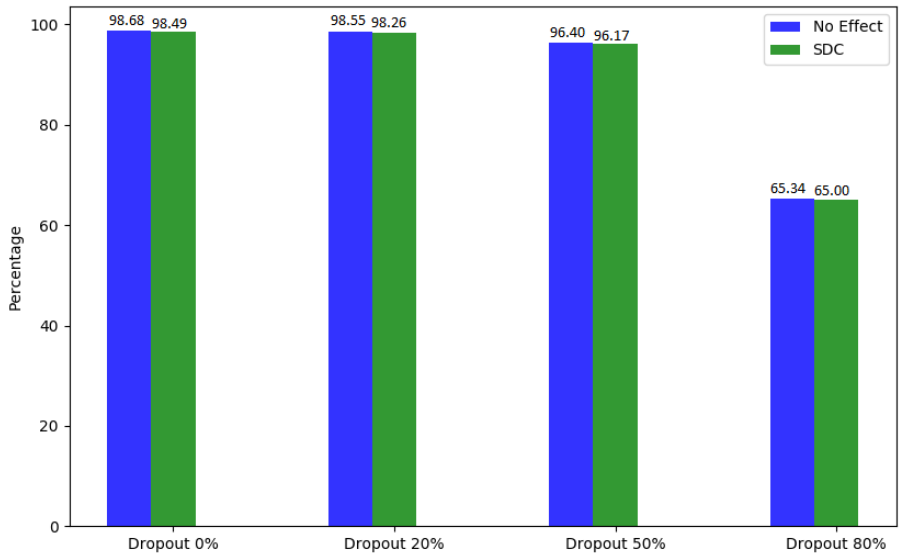

Figure 4.4: Effect of SDC's on the accuracy of Neural Networks trained with Mnist database and different Dropout dropout probabilities

In figure 4.3, we can see that the accuracy decreases as we increase the dropout probability. Being that, the accuracy drops abruptly when we have a dropout probability of 80%.

As for the impact that the SDC's will have on the accuracy, the accuracy will decrease between 0.19% and 0.34% in all dropout probabilities. At first, it doesn't seem to be a very significant difference, but in high performance systems where we need to have a high accuracy, a small change can cause major damage to the system.

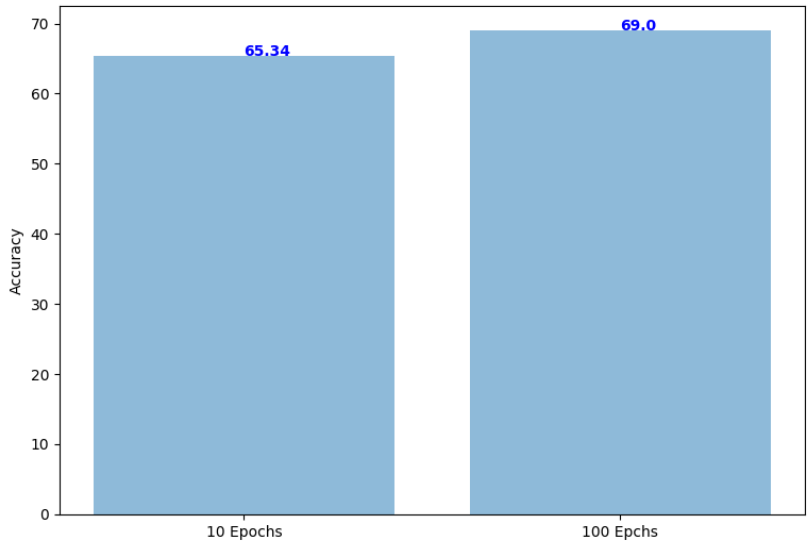

Figure 4.5: Accuracy of Neural Network trained with different number of epochs and Dropout probability of 80%

Regarding the number of SDC's, as shown before in both databases, Mnist and Fashion Mnist, the lowest number of SDC's happens when we have a dropout probability of 80%. But as we can see in 4.3, when we have a dropout probability, the accuracy decreases abruptly. This means that there is a tradeoff between the number of SDC's and the accuracy, that is, we have to compromise the accuracy to have a lower number of SDC's. As a solution to this problem, we increased the number of epochs during training, and with this, we increased the accuracy by 3.66%. Although this does not increase the accuracy by a lot, it just shows that instead of having a tradeoff between the number of SDC's and the accuracy, we can instead increase the training time to have high accuracy with a lower number of SDC's.

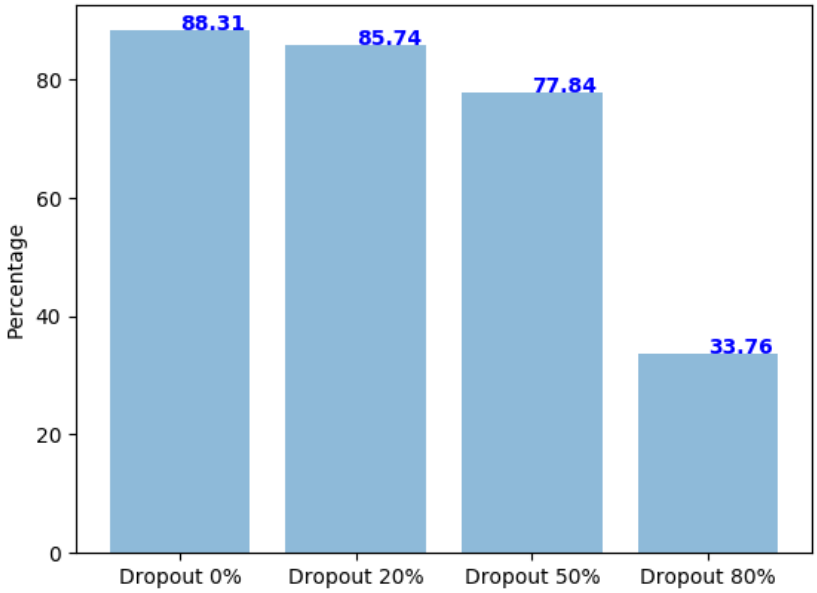

Figure 4.6: Accuracy of Neural Networks trained with Fashion Mnist database and different Dropout dropout probabilities

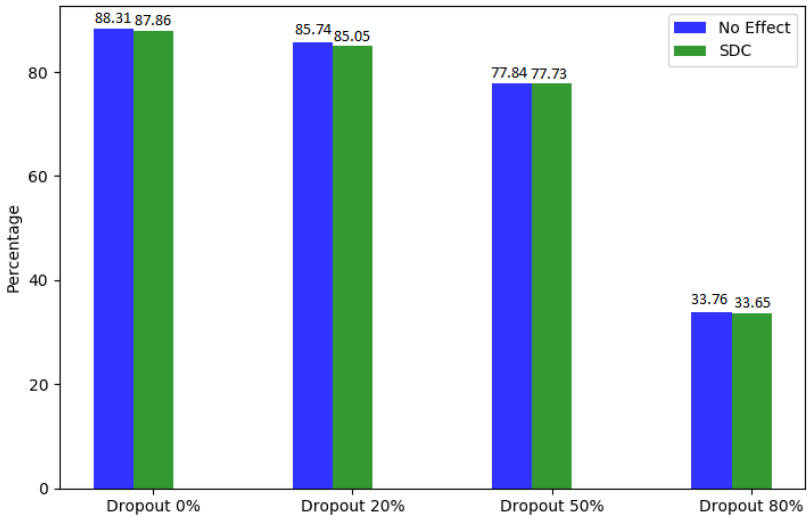

Figure 4.7: Effect of SDC's on the accuracy of Neural Networks trained with Fashion Mnist database and different Dropout dropout probabilities

In figure 4.6, we can see that the accuracy decreases as we increase the dropout probability. Being that as observed in figure 4.1, the accuracy drops abruptly when we have a dropout probability of 80%.

As for the impact that the SDC's will have on the accuracy, the accuracy will decrease between 0.11% and 0.95% in all dropout probabilities. In comparison to the results obtained using the Mnist database, in this case, we will have a more significant decrease in the accuracy when we have a dropout probability of 0% and 20%. When we have a dropout

probability of 50% and 80%, the accuracy will not decrease as much as in the results presented in figure 4.1. The impact that the SDC's have on the accuracy on average is higher in the results obtained using the Fashion Mnist database, meaning this, with a database with higher complexity, the SDC's will have a higher impact on the accuracy.

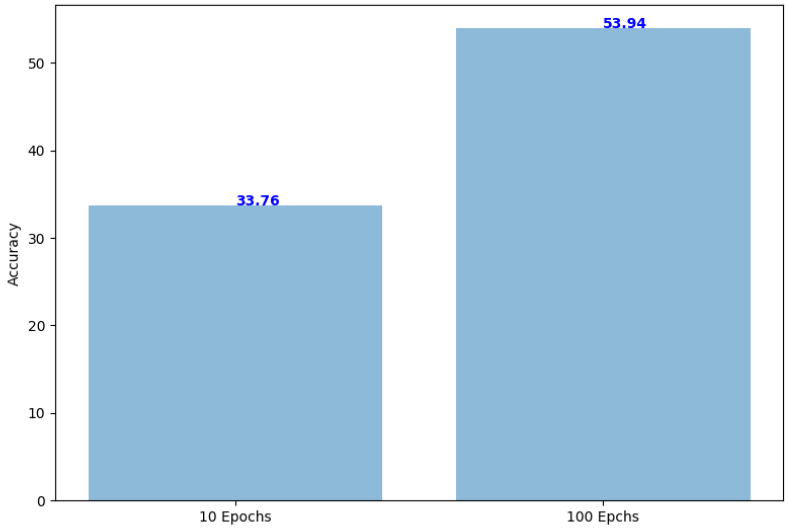

Figure 4.8: Accuracy of Neural Network trained with different number of epochs and Dropout probability of 80%

As mentioned before, the lowest number of SDC's happens when we have a dropout probability of 80%. As we observed for the results obtained using the Mnist database, in this case, there is also a tradeoff between the number of SDC's and the accuracy, that is, we have to compromise the accuracy to have a lower number of SDC's. Contrary to what happened before, here the fact that we increased the number of epochs had a big impact on the accuracy since the accuracy increased by 20.18%. In both cases, at first, there was a tradeoff between the number of SDC's and the accuracy, but we showed that instead, we could increase the training time to have high accuracy with a lower number of SDC's. In the next section, we will the impact that this change has on the training time.

4.1.3 Training Time

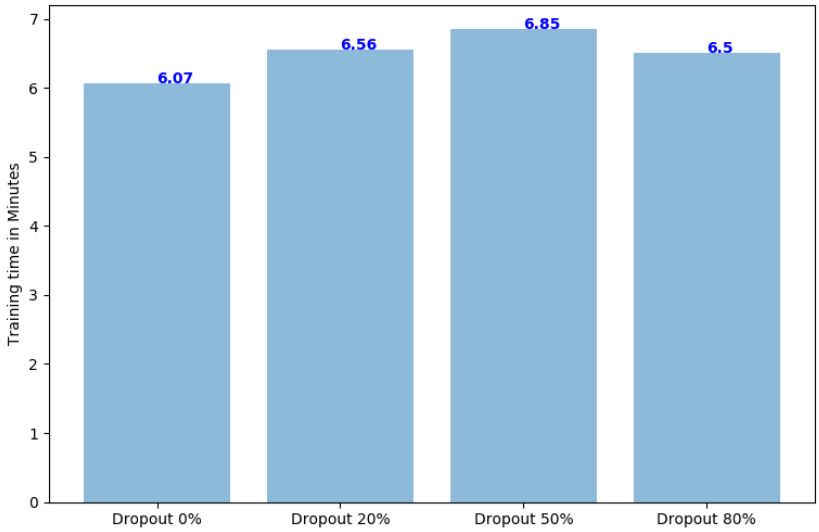

Figure 4.9: Training time of Neural Networks trained with Mnist database and different Dropout dropout probabilities

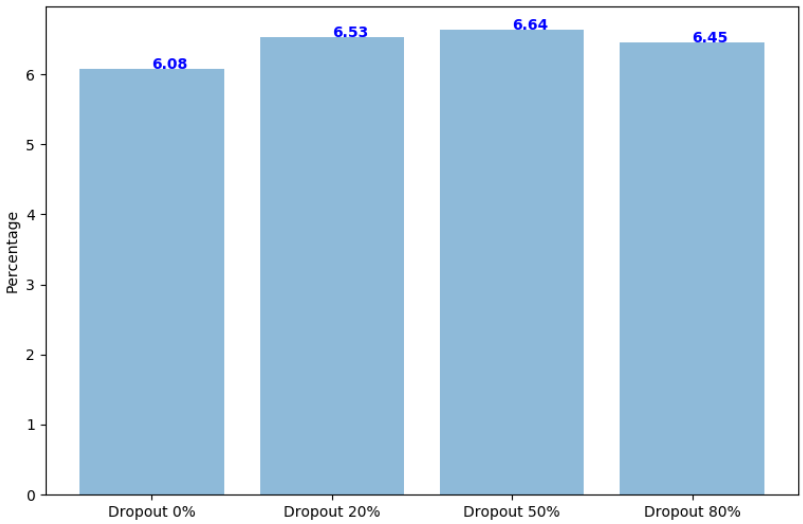

Figure 4.10: Training time of Neural Networks trained with Fashion Mnist database and different Dropout dropout probabilities

In figures 4.9 and 4.10 we can see the training time for every dropout probability, that is the time it took to train each neural network. We can see that in both figure the difference between the training time of each neural network is less than 1 minute, being that the difference is slightly lower in the neural networks trained with the Fashion Mnist database in comparison to the neural networks trained with Mnist database. Hereupon, as expected the training time is similar, since the architecture of the neural networks is similar.

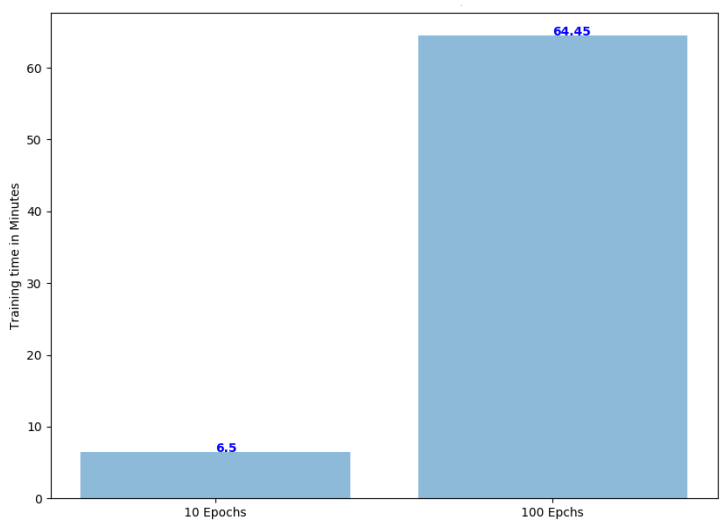

Figure 4.11: Training time of Neural Networks trained with different number of epochs and Dropout probability of 80%

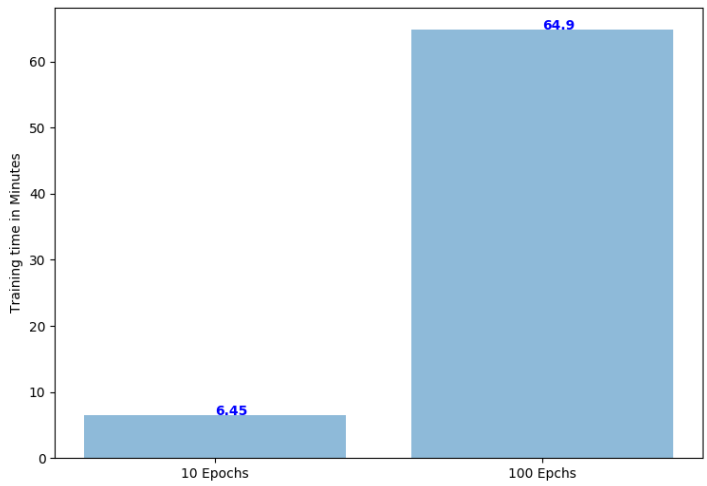

Figure 4.12: Training time of Neural Networks trained with different number of epochs and Dropout probability of 80%

As seen in the previous section, for the Mnist database and the Fashion Mnist database, the lowest number of SDC's happen when we have a dropout probability of 80%. But there was a problem with the fact that the accuracy decreases abruptly when we have a dropout probability of 80%. This means that there was a tradeoff between the number of SDC's and accuracy. As a solution, we increased the number of training epochs, but this means that the training time will increase. In the case of the Mnist database, by increasing the number of training epochs ten times, the accuracy was increased by 3.66%, but the training time was also about ten times bigger, as we can see in figure 4.11. In the case of the Fashion Mnist database by increasing the number of training epochs ten times, the accuracy was increased by 20.18%, but the training time was also about ten times bigger, as we can see in figure 4.12.

4.2 Stimulated Dropout

Table 4.2: Summary of the Experimental Setup for Stimulated Dropout

Dataset	Dropout Probability	Parameter
Mnist	Stimulated Dropout 0%	Number of SDCs
		Accuracy
		Training Time
	Stimulated Dropout 20%	Number of SDCs
		Accuracy
		Training Time
	Stimulated Dropout 50%	Number of SDCs
		Accuracy
		Training Time
	Stimulated Dropout 80%	Number of SDCs
		Accuracy
		Training Time
Fashion Mnist	Stimulated Dropout 0%	Number of SDCs
		Accuracy
		Training Time
	Stimulated Dropout 20%	Number of SDCs
		Accuracy
		Training Time
	Stimulated Dropout 50%	Number of SDCs
		Accuracy
		Training Time
	Stimulated Dropout 80%	Number of SDCs
		Accuracy
		Training Time

In table 4.2 is shown a summary of all the experiences done to study the impact that stimulated dropout has on the number of SDC's, the accuracy and the training time. This will allow us to study the impact that different stimulated dropout probabilities have on the number of SDC's, the impact that those SDC's have on the accuracy and to see if different stimulated dropout probabilities have similar training times. With this set of experiences, we can see the impact that stimulated dropout as on each parameter, and consequently, we can see whether or not stimulated dropout has an impact on the resilience/robustness of ANNs.

4.2.1 Number of SDC's

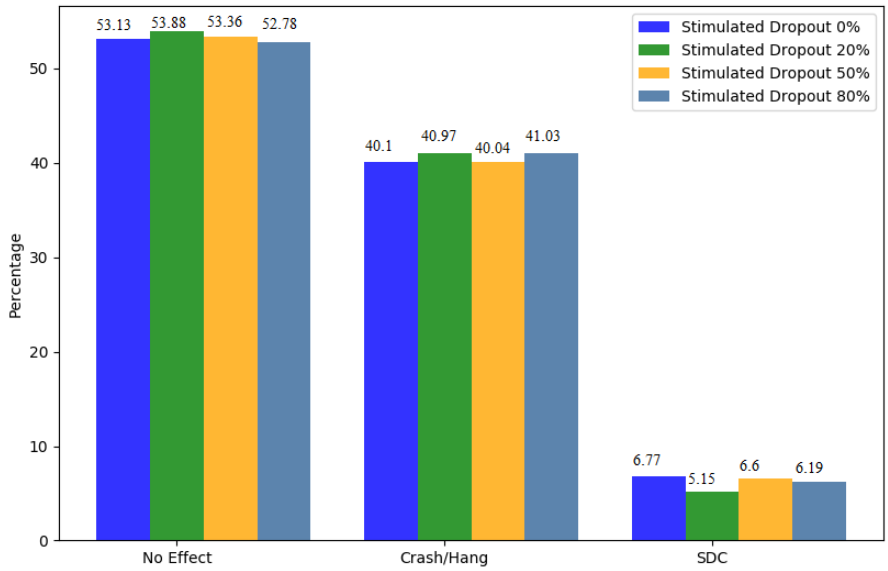

Figure 4.13: Neural Networks trained with Mnist database and different Stimulated Dropout dropoutprobabilities

In figure 4.13, we can see the impact that each stimulated dropout probability has on the percentage of each output. We can see that the number of SDC's is lower in networks trained with stimulated dropout than the network trained without stimulated Dropout. When we have a stimulated dropout probability of 20% we have a percentage of SDC's of 5.15%, 24% less than when we have no stimulated dropout. In the case of a neuronal network trained with stimulated dropout, the highest percentage of SDC's happens when we have a stimulated dropout of 20%, which is 0.17% less than the percentage of SDC's obtained in a neural network trained without stimulated dropout. With these results, we were able to verify that the stimulated dropout technique has an impact on the number of SDC's since the neural networks trained with stimulated dropout will always have smaller number of SDC's. As for the different stimulated dropout probabilities, we can observe that with a dropout probability of 50%, we have a higher percentage of SDC's when compared to the other probabilities. The lowest percentage of SDC's happens when we have a lower stimulated dropout probability.

Regarding the percentage of No Effects, we can see the results vary about 1% between the different stimulated dropout probabilities. We have a higher percentage of No Effects with a stimulated dropout probability of 20% and a lower percentage of No Effects with a dropout probability of 0%.As for the percentage of Crash/Hang, we can see that the highest percentage of Crash/Hang happens when we have stimulated dropout probability of 80%, and the smallest percentage of Crash/Hangs occurs when we have a neural network trained with a stimulated dropout probability of 50%.

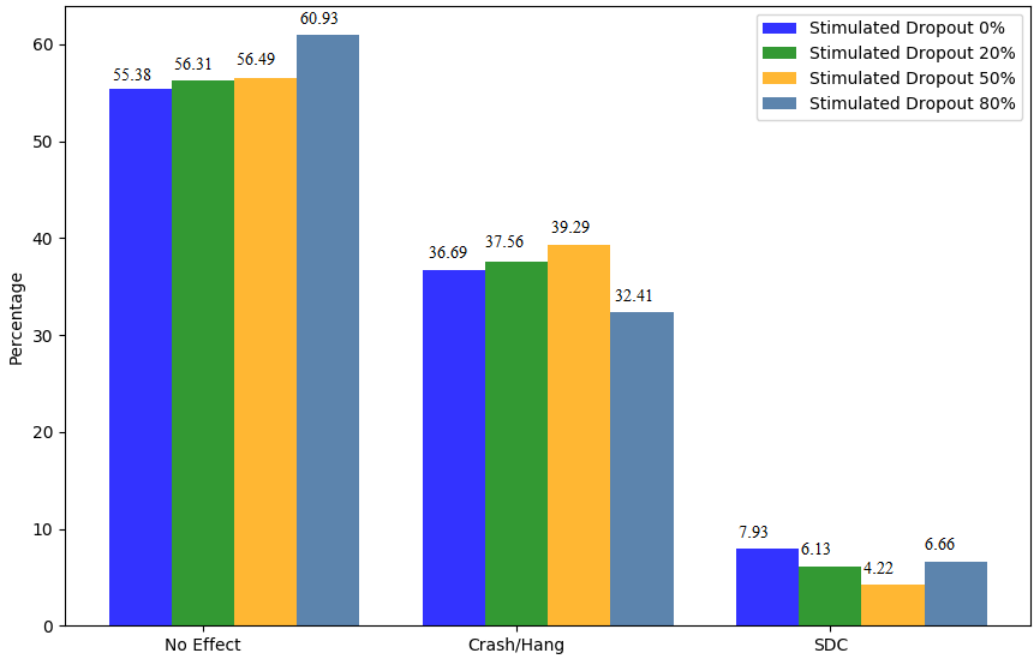

Figure 4.14: Neural Networks trained with Fashion Mnist database and different Stimulated Dropout dropout probabilities

In figure 4.14, we can see the impact that each stimulated dropout probability has on the percentage of each output. As in the case of the Mnist database, here we can also see that the number of SDC's is lower in neural networks trained with stimulated dropout than the network trained without dropout. Regarding the number of SDC's, we can see that the results are similar to theones obtained using the Mnist database. The lowest percentage of SDC's happens when we have a stimulated dropout probability of 50%, we have a percentage of 4.22%, about 47% less than when we have a stimulated dropout probability of 0%. The highest percentage of SDC's when a network is trained with stimulated dropout happens when we have a stimulated dropout probability equal to 80%, which is 1.27% less than when we have a trained network without dropout.

As for the number of No Effects, we can see that there is an increase when we have a stimulated dropout probability of 80%. For the other probabilities the results vary about 2% between them. We have higher percentage of No Effects with a dropout probability of 80% and a lower percentage of No Effects with a dropout probability of 0%. Regarding the number of Crash/Hang, we can see the the highest percentage of Crash/Hang happens when we have a stimulated dropout probability of 50%, and the smallest percentage of Crash/Hang occurs when we have a neural network trained with a stimulated dropout probability of 80%.

If we compare the results obtained in both databases, Mnist and Fashion Mnist, there are some differences between them. The major ones are the fact in the Fashion Mnist there is a bigger decrease in the percentage of SDC's when we compare the results of the neural networks trained with stimulated dropout in comparison with the neural network trained without stimulated dropout. The second one is that the number of SDC's increases when we used the Fashion Mnist. Being that, from these results we can conclude that with the increase of the complexity of the database, there was an increase in the number of SDC's.

The results presented in figures 4.13 and 4.14 demonstrate that a network trained with stimulated dropout, regardless of the stimulated dropout probability, will have a lower number of SDC's than a network trained without stimulated dropout. With these results, we can say that the timultead dropout can be used to increase the resilience/robustness of the ANNs.

4.2.2 Accuracy

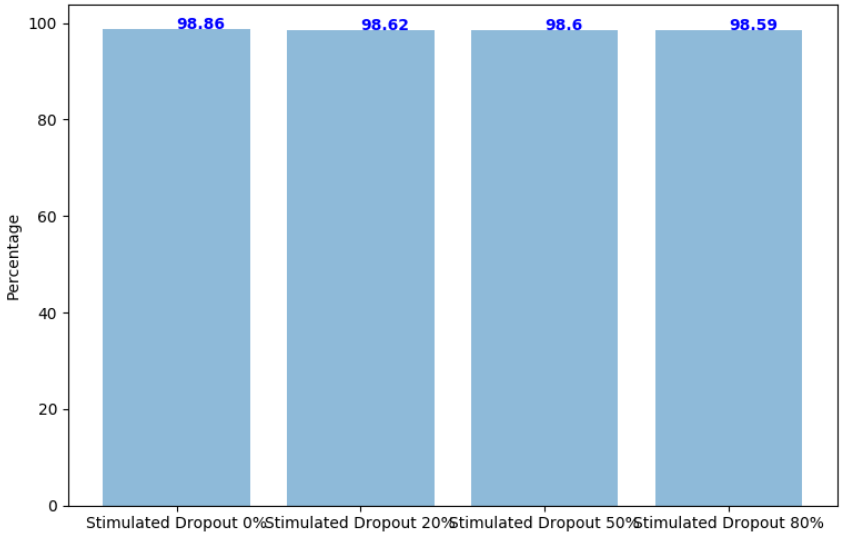

Figure 4.15: Accuracy of Neural Networks trained with Mnist database and different Stimulated Dropout dropout probabilities

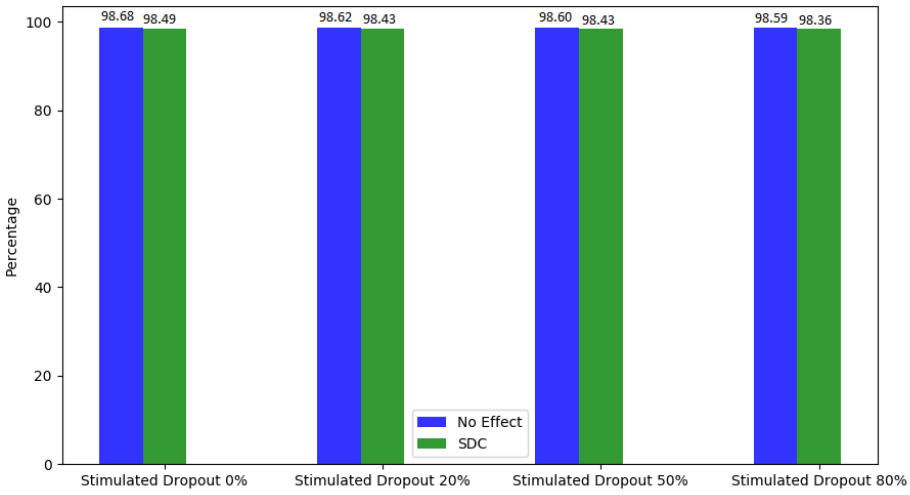

Figure 4.16: Effect of SDC's on the accuracy of Neural Networks trained with Mnist database and different Stimulated Dropout dropout probabilities

In figure 4.15, we can see that the accuracy decreases as we increase the stimulated dropout probability.Being that, the accuracy just slightly changes between all the networks. As for the impact that the SDC's will have on the accuracy, the accuracy will decrease between 0.17% and 0.23% in all dropout probabilities. As mentioned before, although at first, it doesn't seem to be a very significant difference, but in high performance systems where we need to have a high accuracy, a small change can cause major damage to the system.

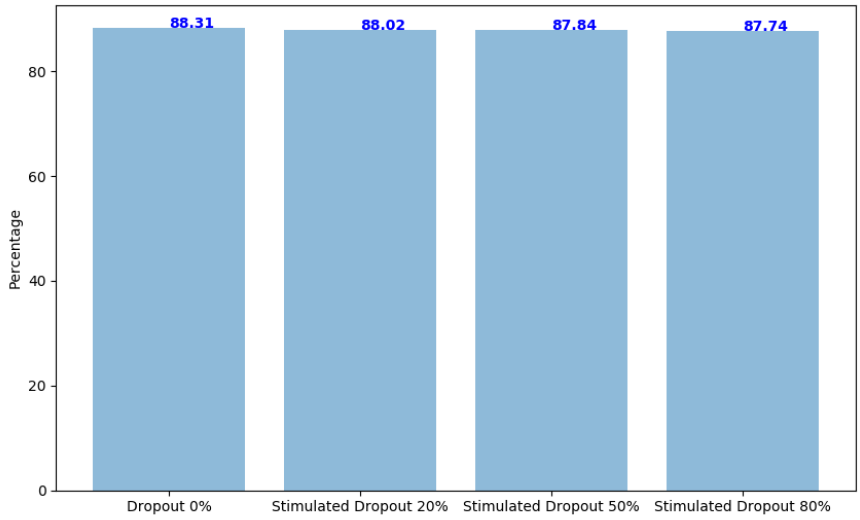

Figure 4.17: Accuracy of Neural Networks trained with Fashion Mnist database and different Stimulated Dropout dropout probabilities

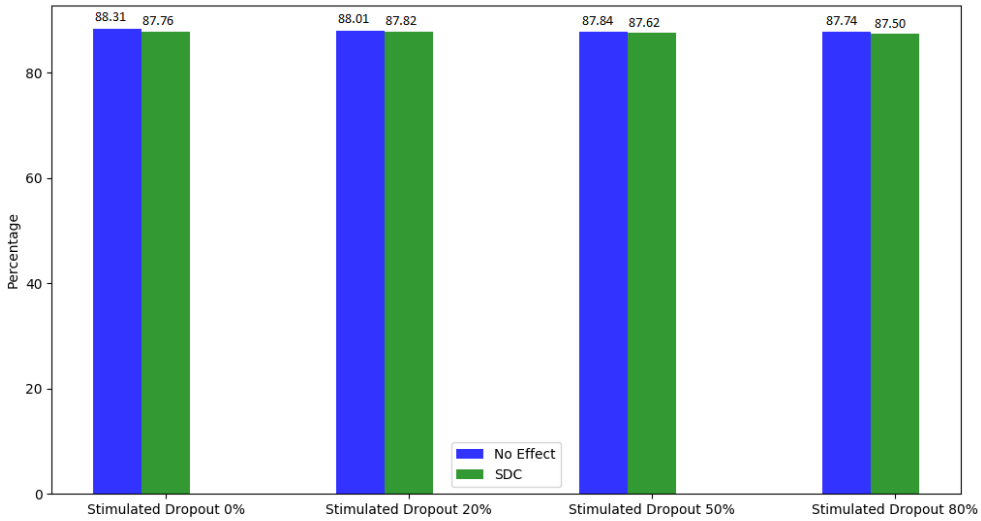

Figure 4.18: Effect of SDC's on the accuracy of Neural Networks trained with Fashion Mnist database and different Stimulated Dropout dropout probabilities

In figure 4.17, we can see that the accuracy decreases as we increase the stimulated dropout probability.Being that as observed in figure 4.15, the accuracy just slightly changes between all the networks. As for the impact that the SDC's will have on the accuracy, the accuracy will decrease between 0.19% and 0.55% in all stimulated dropout probabilities. . The impact that the SDC's have on the accuracy on average is higher in the results obtained

using the Fashion Mnist database, meaning this, with a database with higher complexity, the SDC's will have a higher impact on the accuracy.

4.2.3 Training Time

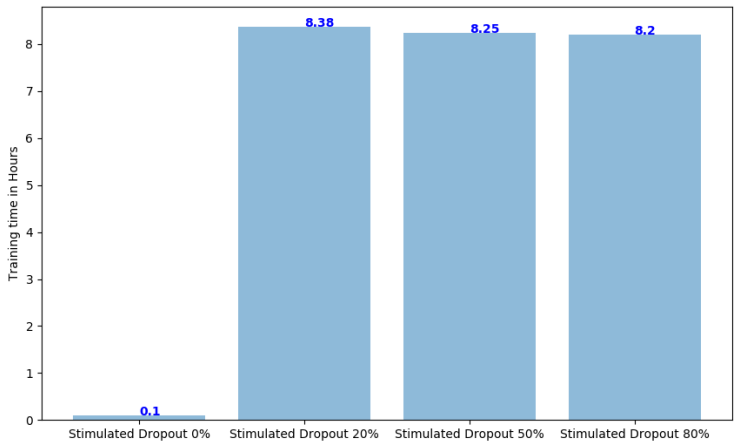

Figure 4.19: Training time of Neural Networks trained with Mnist database and different Stimulated Dropout dropout probabilities

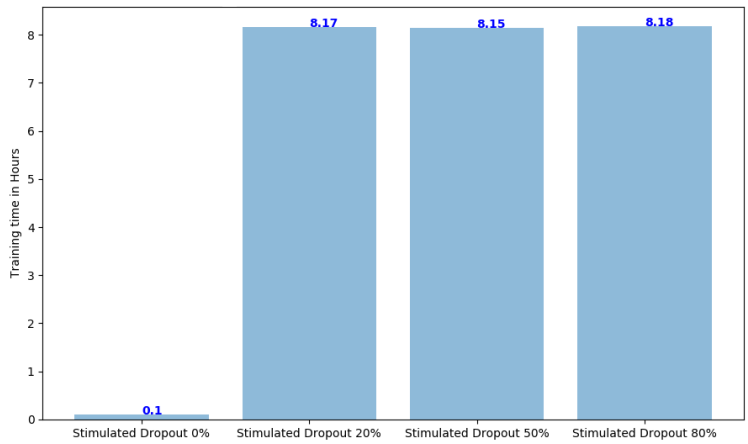

Figure 4.20: Training time of Neural Networks trained with Fashion Mnist database and different Stimulated Dropout dropout probabilities

In figures 4.19 and 4.20 we can see the training time for every stimulated dropout probability, that is the time it took to train each neural network. We can see that in both figure the difference between the training time of each neural network is less than 2 minutes, being that the difference is slightly lower in the neural networks trained with the Fashion Mnist database in comparison to the neural networks trained with Mnist database. Hereupon, as expected the training time is similar, since the architecture of the neural networks is similar.

4.3 Discussion

The results show that adding dropout and stimulated dropout to the layers of the neural networks will decrease the number of SDCs, which means that both techniques have a positive impact on the resilience/robustness of ANNs. Regarding the results, we find some differences between them.

If we compare both techniques based on the probabilities in which we obtained a lower number of SDCs for the Mnist database, we can see that when we have a network trained without any of the techniques, we have a percentage of SDCs of 6.77%, being that with a dropout probability of 80% the number of SDCs will be 3.76%, about 45% less. While in the case of the stimulated dropout, the lowest percentage of SDCs occurs when we have a probability of 20%, where we have a percentage of SDCs of 5.15%, about 24% less. For the Mnist database, Stimulated Dropout has a smaller impact on the number of SDCs than Dropout. The same is not valid for the Fashion Mnist database, where the network trained without any of the techniques has a percentage of SDCs of 7.93%, and with a dropout probability of 80%, the number of SDCs will be 4.28%, about 46% less. While in the case of the stimulated dropout, the lowest percentage of SDCs happens when we have a 50% probability, where we have a percentage of SDCs of 4.22%, about 47% less. It is also possible to observe that the smallest percentage of SDCs in the case of the dropout always happened when we had a probability of 80%, while in the case of the stimulated dropout, this probability varied being 20% for the Mnist database and 50% for the Fashion Mnist database.

In both databases, the accuracy decreases abruptly when we have a dropout probability of 80%, while in the case of the stimulated dropout, the accuracy just slightly changes in all probabilities. As mentioned in Chapter 2, if the probability of stimulated dropout is 50%, there is a 50% chance that we flip a bit in the value, but this does not mean that this value has changed since if the bits are of the mantissa, it usually has no impact on the value. This makes that the probability of stimulated dropout does not have as much impact on accuracy as even though there is a greater probability that the value will be changed, this does not mean that it will be changed since it depends on the bit. That said, there are two paths that can be explored in the future so that the probability of stimulated dropout has more impact on both accuracy and the number of SDCs. The first option would be that the probability of stimulated dropout was related to the number of values that would be changed. As mentioned in Chapter 2, it was not possible to change each one of the values in each layer since the training time would be too high. That said, we flip one-third of the values in each layer. The new probability of stimulated dropout would change this. That is, if we have a 50% probability, this will mean that 50% of values in each layer can be flipped. We believe that this will decrease the number of SDCs, but on the other hand, it would decrease the accuracy as the probability increased, and it would increase the training time. As for the accuracy, we already showed that it could be increased by increasing the training time. With this solution, we would end up with the same problem. The second option would be that the probability of stimulated dropout was related to the flipped bit. As mentioned before, if we flipped a bit of the mantissa, the probability that the value is changed is lower than if you flip a bit of the exponent. That said, in this case, if we have a 50% probability of stimulated dropout, this will mean that there is a 50% chance that the bit that we are going to flip is a bit of the exponent. Regarding this second option, we do not know how it would impact the number of SDCs since, during the tests, the bit can be both of the mantissa and the exponent.

As for the training time, this is the biggest problem of the stimulated dropout when

compared to dropout. In order to understand why the training time is higher than the training time of dropout, let's review some concepts introduced in chapter 2. Neurons of different layers exchange information with each other, the output of a neuron is the input of other neurons, this information is a real value. In the case of dropout, there is a probability that this value will be replaced by 0. In the case of stimulated dropout instead of replacing this value with 0, we will do a bit-flip of this value. These values are in a matrix, the matrix size depends on the type of the layer. For example, if we have a fully connected layer we will have a square matrix, and if we have a convolution layer we have a quadratic matrix. That said, what happens in the case of the dropout is that this matrix is multiplied with a matrix of 0's and 1's, causing some values of that matrix to become 0 and others to maintain its value. While in the case of the dropout we only do a simple multiplication of matrices in the case of the stimulated dropout we will have to change each value individually which will increase the complexity and consequently increase the training time. This is the biggest obstacle to the stimulated dropout, and if we find a solution to this problem, we can flip more bits during training and consequently reduce the number of SDCs.

Chapter 5

Conclusion and Future Work

We are surrounded by systems that rely on ANNs in order to function correctly. Some of these systems are considered SCSs since there are human lives that depend on them. In this work, we address the fact that there is a need to ensure resilience/robustness in ANNs. There are several techniques already studied to increase resilience in ANNs. In this work we studied the impact that the Dropout technique has on the resilience of an ANNs. To the best of our knowledge this is the first study evaluate Dropout for such aim. We will also introduce a new technique that aims to improve the resilience/robustness of ANNs.

To accomplish the proposed objectives, we trained multiple neural networks with different dropout and stimulated dropout probabilities. In addition to that, all the networks were trained and tested using two different datasets, the Mnist dataset, and the Fashion Mnist dataset. During the tests, will be injected into the hardware, specifically memory bit-flips. In order to understand the impact of both techniques in the resilience of ANNs, the results were analyzed using three parameters, the number of SDC's, accuracy, and training time. With this, we were able to make a complete analysis of the results and better understand the impact that each technique had on the resilience/robustness of ANNs.

Regarding the results, in both datasets, the largest number of SDCs occurs when we have network trained without any techniques. When we add dropout and stimulated dropout to the neural networks, the number of SDCs will decrease, regardless of the probability associated with each technique. Being that, for the Mnist dataset, the number of SDCs will decrease by about 45% when we have a dropout probability of 80%. Whereas for the Fashion Mnist dataset, the number of SDCs happens will reduce by approximately 46% when we have a stimulated dropout probability of 20%. With these results, we can see that both dropout and stimulated dropout positively impact the resilience/robustness of ANNs. In the case of dropout, the lowest number of SDCs in both datasets happens when we have a dropout probability of 80%. Being that when we have a dropout probability of 80%, it is also possible to observe an accentuated decrease in accuracy. With this, it is possible to observe a tradeoff between the accuracy and the number of SDCs. Regarding the Stimulated dropout, there is a slight decrease in accuracy as the probability increases. In this case, we can observe a tradeoff between the number of SDCs and the training time.

In this work, we show a new aspect of dropout, dropout as a technique to increase the resilience/robustness of ANNs. Besides, we developed a new technique called stimulated dropout. As for the stimulated dropout case, the results also demonstrate that this technique can be used to increase the resilience/robustness of the ANNs. Being that in the case of the stimulated dropout, as it is a recently developed technique, there is still room for progress, and we think it is possible to increase the impact that this technique has on

the resilience/robustness of the ANs.

5.1 Future Work

Although in this work, the networks were trained with different dropout and stimulated dropout probabilities, all of them had the same architecture. That said, in the near future, it would be interesting to see the impact that a different architecture would have on the results. In addition, as it was possible to observe, that there was an increase in the number of SDCs when we compare the results obtained for the Mnist dataset with the results obtained for the fashion Mnist dataset. It would also be interesting to see if, with a dataset of greater complexity, the number of SDCs would increase even more. With this, we could affirm that with the increase of the dataset's complexity, the number of SDCs would also increase. This would reinforce the need to ensure the resilience/robustness in ANNs.

Regarding stimulated dropout as this was a newly developed technique, there are many paths that can be explored in the future. Bearing in mind that the biggest obstacle to this technique's progress is the training time, it would be ideal from an early stage to solve this problem to understand if we could further increase the impact that this technique has on the resilience/robustness of ANNs. Even if we do not find an immediate solution to this problem, there are other possibilities that can be explored. The first option is related to the probability of stimulated dropout. In this case, the probability would be related to the number of values that would be changed at each training iteration. As mentioned in Chapter 2, we flip one-third of the values, with the new probability if we have a 50% probability, this will mean that 50% of values can be flipped. We believe that the number of SDCs will decrease as the probability increases. But on the other hand, the accuracy will also decrease as the probability increases. The second option is to relate the probability of stimulated dropout with the flipped bit. As mentioned before, if we flipped a bit of the mantissa, the probability that the value is changed is lower than if you flip a bit of the exponent. In this case, if we have a 50% probability of stimulated dropout, this will mean that there is a 50% chance that the bit that we are going to flip is a bit of the exponent.

References

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 01(1):11–33, 2004.

[2] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical Fault Attack on Deep Neural Networks. pages 2204–2206, 2018.

[3] Frederico Cerveira. ucxception. [Online; accessed 1-October-2019], <https://ucxception.dei.uc.pt>.

[4] Frederico Cerveira and Faculdade De Ciências. Soft error sensitivity and vulnerability of languages and their implementations. x(x):1–31.

[5] Frederico Cerveira, Imre Kocsis, Raul Barbosa, Henrique Madeira, and András Pataricza. Exploratory data analysis of fault injection campaigns. *Proceedings - 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security, QRS 2018*, (August):191–202, 2018.

[6] Chih Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10482 LNCS:251–268, 2017.

[7] Joseph Clements and Yingjie Lao. Hardware trojan design on neural networks. *Proceedings - IEEE International Symposium on Circuits and Systems*, 2019-May, 2019.

[8] James Elliott, Frank Mueller, Miroslav Stoyanov, and Clayton Webster. Quantifying the Impact of Single Bit Flips on Floating Point Arithmetic. *Preprint*, 2013.

[9] Aurelien Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* 2017.

[10] github. github. [Online; accessed 30-November-2019],

[11] Mei Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer. Fault injection techniques and tools. *Computer*, 30(4):75–82, 1997.

[12] John C. Knight. Safety critical systems: Challenges and directions. *Proceedings - International Conference on Software Engineering*, pages 547–550, 2002.

[13] Jc Laprie. Dependability and resilience of computing systems. pages 0–21, 2010.

[14] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. *Deep learning*, volume 521. 2015.

[15] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[16] Nvidia. Nvidia pegasus available at: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/. [Online; accessed 05-September-2020],

[17] Tutorials Point. Single precision ieee 754 floating-point standard [image] available at: https://tutorialspoint.dev/computer-science/computer-organization-and-architecture/ieee-standard-754-floating-point-numbers. [Online; accessed 23-July-2020],

[18] PyTorch. Pytorch. [Online; accessed 25-November-2019],

[19] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:1211–1220, 2019.

[20] SAE. Sae levels of driving automation available at: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic. [Online; accessed 05-September-2020],

[21] Behrooz Sangchoolie, Karthik Pattabiraman, and Johan Karlsson. One Bit is (Not) Enough: An Empirical Study of the Impact of Single and Multiple Bit-Flip Errors. *Proceedings - 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017*, pages 97–108, 2017.

[22] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a Formal Model of Safe and Scalable Self-driving Cars. pages 1–37, 2017.

[23] Vishal Sharma. Shift-invariant convolution neural network (cnn) [image] available at: https://www.quora.com/what-is-shift-invariance-in-a-convolutional-neural-network-cnnoswoo.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[25] Tesla. Tesla fsd available at: https://www.tesla.com/support/full-self-driving-computer. [Online; accessed 05-September-2020],

[26] TheGuardian. Uber self-driving crash. [Online; accessed 28-December-2019],

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[28] YannLeCun. Mnist. [Online; accessed 17-October-2019],