T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4197 Engineering Project I - Project Specification Document

"ANALYZING DEPENDABILITY OF DEEP NEURAL NETWORKS (DNNs)"

01.12.2022

Group Members

Ozan Yerli - 150118064

Elif Gülay - 150119732

Furkan Erdoğan - 150119668

Supervised by

Asst. Prof. Sanem Arslan Yılmaz

Table of Contents

1. Problem Statement

Due to the great success of deep neural networks (DNNs), its vulnerability has attracted great attention, especially for security-critical applications and autonomous driving [1]. Dependability of DNN is crucial as it is widely utilized in such critical systems. If output is predicted incorrectly, catastrophic failures may occur. In our project, we aim to analyze the dependability of DNNs to detect the most vulnerable layer(s) and also propose fault-tolerant methods to protect the most critical layer(s).

2. Problem Description and Motivation

Deep neural networks (DNN) have been successfully used in solving a wide range of classification problems. DNN algorithms can learn from a given large set of training data to accurately classify a given input pattern. In recent years, DNNs have been proven to provide state of the art results in computer vision with highly accurate prediction rates. Today, DNNs are used in many critical and security sensitive systems such as face recognition, medical diagnosis, and autonomous driving. Since it is frequently used in such critical systems, dependability of a DNN is crucial.

Fault-tolerance is prevention of failure in a system in case of occurrence of faults. Error detection, error containment, and error recovery are some of the methods to improve fault-tolerance of systems.

The output of a DNN depends on the input data and the data in the hidden layers. Integrity of this data is crucial for the prediction accuracy of a DNN. Corruption of data in these layers may result in a misclassification by the DNN in the output layer. The corruption of data may be caused by alpha particles, cosmic rays, thermal neutrons, or other environmental causes. This type of corruption is called a soft error. Such soft errors can cause application failures, and they can result in violations of safety and reliability specifications [2].

Fault injection has become a very classical method to determine the dependability of an integrated system with respect to soft errors [3]. Fault injection methods quantify the

reliability level of a system by inserting faults into the various system components at different times and evaluating the outcome.

A misclassification caused by soft errors may result in major problems in the security-critical systems which leverage DNN algorithms. A misclassification of an object by a DNN used in autonomous driving systems may lead to dangerous accidents. A DNN used in medical diagnosis may predict a misdiagnosis, leading to serious health and safety implications. Therefore, dependability of a DNN is of great concern.

The motivation behind this project is to improve the dependability of DNNs. The project aims to analyze the criticality of layers of the neural network and propose fault-tolerant methods to protect the most critical layer(s).

3. Main Goal and Objectives

The main goal of the project is to identify the most critical layer(s) in the created neural network, and also identify the methods to protect this layer to make it more fault tolerant. The necessary objectives to achieve this goal can be listed as:

(i) To create a neural network where we can detect and protect the most critical layer(s),

- Training the neural network we created with a large data set,
- Testing the neural network using test data sets to ensure the training process was successful,

(ii) To implement a fault-injector to corrupt the data in each layer of the neural network by sending a large number of errors,

(iii) To detect the most critical layer(s) in the neural network by assessing the results of fault injections experiments,

(iv) To provide methods to make the most critical layer(s) more fault-tolerant to avoid failures occurring in those layer(s).

4. Related Work

In the work of Li et al [2], authors performed a large-scale fault injection study using a simulator for faults that occur in the datapath of accelerators. Authors categorized error propagation characteristics based on the neural network's structure, data types, layer placements, and layer types. Authors examined two cost-effective error protection strategies to reduce Silent Data Corruptions (SDCs, which are data errors that are undetected by the system), or inaccurate results, and the reliability implications of constructing DNN accelerators and applications. Software is used to implement the first strategy, symptom-based detectors, while hardware is used to achieve the second technique, selective latch hardening. Authors analyze these methods based on their overhead and fault coverage. Authors have discovered that the network architecture, the type of data being used, and the bit location of the defect all affect how sensitive a given DNN is to SDCs. Li et al [2], found that only high-order bits are vulnerable to SDCs, and also susceptibility increases with the dynamic value range that the data type may represent. The overall FIT (Failure in Time) rate of the DNN system can be decreased by more than an order of magnitude. This is consistent with prior work that suggested the usage of such data types for energy efficiency. Authors also stated that, with the help of the suggested selective latch hardening technique, they may selectively safeguard the vulnerable bits by taking advantage of the asymmetry in the SDC sensitivity of bits. Their analysis reveals that with a 20% area overhead, the equivalent Failure in Time (FIT) rate of the datapath can be decreased by 100 times.

In the work of Su et al [4], it was highlighted how Artificial Neural Networks (ANNs) have a great computational capacity to mimic human reasoning in order to identify appropriate answers for a wide range of problems. They focused on the capability of defining a fault-tolerant system based on Artificial Neural Networks properties. A unique approach was introduced by Zhang [4], that automatically analyzes log files to identify any failure warning signals. The authors evaluated the proposed solution by contrasting it with machine learning methods based on logical traces. The results showed that the suggested approach has a strong ability to predict breakdowns in complex systems.

Sampaio and Barbosa [5], focused on how the growth of data centers has led to an increase in failure rates. The authors examined the availability and energy costs of two state-of-the-art fault-tolerant mechanism approaches. The results demonstrated that proactive fault-

tolerant systems outperform conventional techniques in terms of performance. The authors draw attention to the fact that the correctness of the proactive failure tolerance mechanism is still largely reliant on the specs of the stated model.

In conclusion, studies show that the data types, values, data reuse, and layer types in the design all determine how dependable a DNN system is. In our project, we examine the dependability of DNNs to identify and protect the most vulnerable layer(s). Also, fault detection and layer protection is going to be done at the application level compared to the related works.

5. Scope

Neural network architectures are specific arrangements of the hidden layers of a neural network. These layers are responsible for convolution, pooling, and normalization processes. Over the years, several DNN architectures became prominent in classification problems. AlexNet and ConvNet are one of these architectures.

The project is based on existing implementations of AlexNet and ConvNet neural network architectures. The neural network architectures utilized in this project are used to classify images. Image classification is the task of assigning a label or class to an entire image. Images are expected to have only one class for each image in our project.

The results obtained in this project may not be applicable to other neural network applications which use another set of classes or that do not use the neural network architectures selected for this project.

Hardware and memory constraints have to be taken into account in our project, as training of neural networks requires a lot of memory and processing power. It is relatively harder to train a neural network on a typical computer we use in our daily life. A lack of computational resources may result in a crash of the neural network or it may prolong the execution time to unfeasible amounts. It is required to find a machine with a high amount of GPU processing power and memory bandwidth to avoid these issues**.**

In this project, a series of fault injection experiments will be repeated for a large amount, resulting in manipulation of thousands of bytes in the layers of the neural network. The neural network models will be tested with the corrupted data. The execution of these tests may also require a high amount of time and computational resources and may be affected by hardware and memory constraints.

6. Methodology and Technical Approach

Our approach to determine the fault tolerance of a DNN will be based on quantitative methods. To run our tests and experiments, we will train and test neural network models. Using an application-based fault injector, we will corrupt data in the layers of the neural network in the testing phase. After experimenting with a high number of fault-injection tests, the most critical layer(s) will be detected using the data obtained in our tests. Then we will provide methods to make the most critical layer detected more fault tolerant.

DNNs are neural networks that consist of layers of neurons. Each neuron takes an input, processes it with mathematical operations, then passes the result as input to other neurons that are connected to it. The first layer of neurons in a DNN is called the input layer. The last layer of neurons is the output layer. The output layer consists of the given number of classification classes for the classification problem. The layers between the input and the output layers are called the hidden layers. In Figure 1, a deep neural network used for image classification and its layers are visualized.

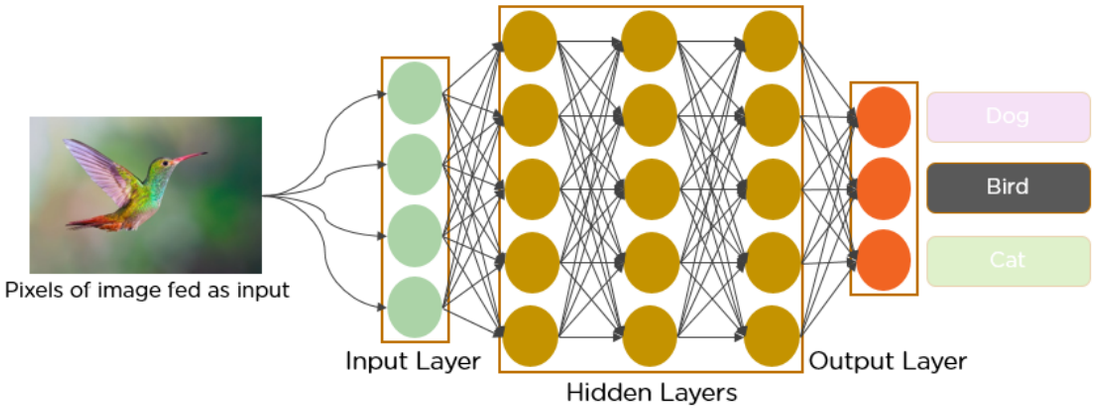

Figure 1: A deep neural network used in image classification [6].

The AlexNet and ConvNet neural network architectures will be used as our neural network models, as these models are the state of the art algorithms in image classification.

These neural network models will be implemented using the Python programming language. TensorFlow and Keras open-source Python libraries will be used for constructing the neural network models and getting access to large image datasets.

CIFAR-10 and ImageNet datasets will be used in training of the neural network models, as these are large image datasets that are commonly used in training of neural networks for image classification purposes.

Neural network models are constructed in 3 phases with separate datasets. These are training, validation, and testing phases. In the training phase, the neural network uses the data to learn and train the model. In the validation phase, the model is evaluated. After the model is completely trained using the training and validation datasets, the model is evaluated in the testing phase against competing models. The testing phase is where we will perform fault injection testing and corrupt data in the layers of the neural network.

We will implement a software-based fault injector in Python for this purpose. The fault injector will be used to corrupt data in each layer of the neural network in the testing phase. The fault injection process will be repeated using different parameters such as the layer of injection, type and timing of the injection. In Figure 2, a simple neural network is visualized, in which data in one of the hidden layers of the neural network is corrupted by a fault injector, resulting in a misclassification of an image.

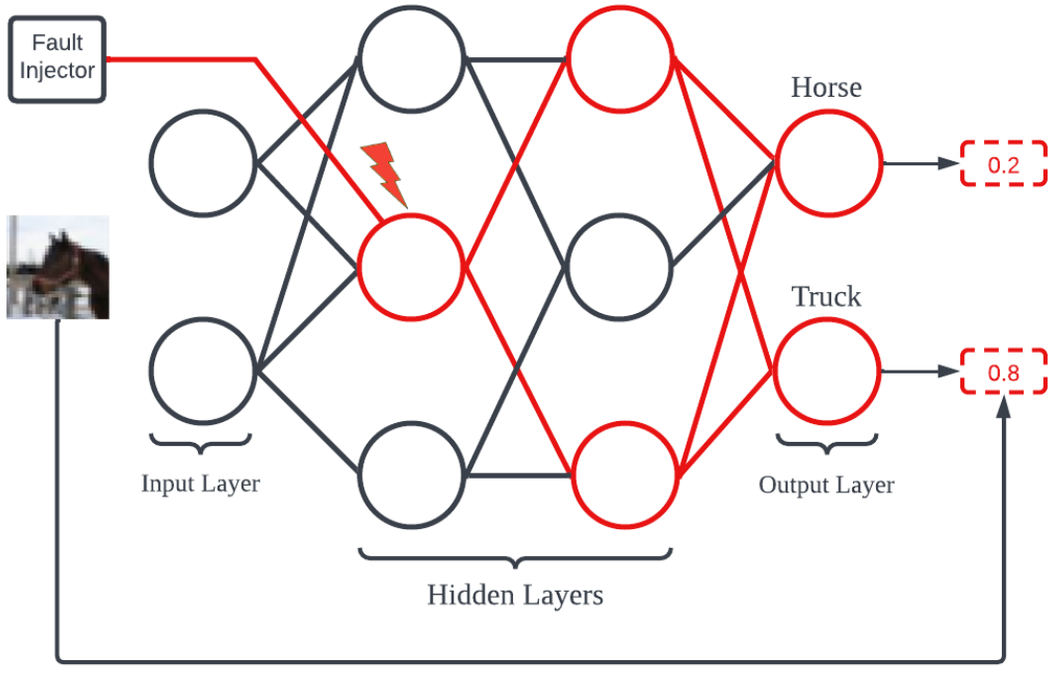

Figure 2: A misclassification by a fault-injected neural network

We will be conducting a large amount of fault injection testing in the neural network to understand how it is behaving to such attacks and measure the fault tolerance of the layers in the neural network by examining the accuracy of the outputs after fault injection experiments. Using the data obtained in our tests, we will determine the most vulnerable layer in the neural network.

Training and testing of neural networks require a lot of memory and processing power. In order to not face any hardware or memory constraints, we will be using Google's Colab service. Colab will allow us to run neural network models implemented in Python in our browsers and provide us with the necessary computing resources including GPUs and memory.

After the most vulnerable layer in a neural network is determined, we will be investigating error recovery techniques to increase resilience of the neural network. By conducting another round of testing, we will measure and compare the efficiency of techniques of error recovery. The error recovery will include techniques such as duplication/triplication of weights in the most critical layers to provide redundancy or executing them multiple times on different cores.

The results we obtained in testing of fault-injected neural networks with error recovery techniques will be used to evaluate the success of our project. The prediction accuracy rates in our testing will be compared to the accuracy rates of the same neural network models without injecting faults.

Finally, we will present our findings on the fault tolerance of the deep neural networks, visualize the data we obtained in our tests, and compare the efficiency of the error recovery techniques presented.

7. Professional Considerations

7.1 Methodological considerations/engineering standards

We are going to use GitHub for source code control during development of the project. GitHub provides coordination and consistency between team members. We will use Python programming language for the project since it has large library support for the image processing and neural network.

We are going to use AlexNet for the neural network architecture for the project. AlexNet has architecture for object detection tasks and large applications in machine learning problems.

ConvNet will be the other neural network architecture for our project. ConvNet is more frequently used for classification and computer vision tasks. It provides a more scalable approach to image classification.

Cifar-10 is going to be the first dataset of the neural network which contains 50000 training images and 10000 testing images. The Cifar-10 dataset is generally used in machine learning algorithms.

Imagenet is going to be the second dataset of the neural network with more than 1 million training images and 10000 testing images. The Imagenet dataset provides an improved method for computer vision.

Training and testing of the neural network will be on Google Colab due to their resources they provide (GPU and other computing resources). Google Colab is a service from Google Research that allows users to write and execute their Python code on the browser. It is mainly used and well suited to machine learning and data analysis.

7.2 Realistic Constraints

7.2.1 Economical

As we stated in Section 2, Deep neural networks (DNNs) have been successfully used in solving a wide range of classification problems. In systems that use DNN algorithms, an incorrect output prediction could result in serious issues. Minimizing the error rate will follow an economically positive path which will be a result for avoiding unnecessary expenditure to building a new neural network. In this project, the most important economical

constraint is reducing expected maintenance cost since we don't expect any profit from the project.

7.2.2 Environmental

In this project, there will be no potential effect on environmental pollution however detecting the critical layer will result in more energy usage. Protecting only the detected critical layer will result in less energy usage rather than protecting all layers of the network.

7.2.3 Sustainability

Deep Neural Networks (DNNs) can be used for real-life issues that we encounter every day, such medical image analysis, video conferencing, and autonomous driving, when combined with the correct image data. If Deep Neural Networks (DNNs) continue with state-of-the-art results and as it continues to be applied in crucial areas for safety, our project will be sustainable. If Deep Neural Networks (DNNs) replace themselves with a new technology, then our project will be unsurvivable and irrelevant.

7.2.4 Ethical

In this project, there is no violation of any security or privacy of the users. We are going to use open-source resources and there is no implicit use of patent protected design and concepts.

7.2.5 Health and Safety

In this project, there are no potential effects on the health of users and the public. We don't threaten users and there are no special safety considerations for the usage of infants and children since it is created by codes.

7.2.6 Social

As we mentioned in Section 2, project that we are developing is not related with any discrimination of genders, races and it is not destructive for people.

7.3 Legal considerations

Programs that will be used in this project are open source and we do not need a license for developing the project.

8. Management Plan

8.1 Description of Task Phases

8.1.1 - Literature Survey of Deep Neural Networks and Fault tolerance Methods

8.1.2 - Creating a neural network - training and testing of the neural networks

8.1.3 - Implementation of an application-level fault injector

8.1.4 - Testing of the fault injector for different data types, fault locations and timings

8.1.5 - Experimenting with a high number of fault injection tests to determine the most critical layer(s)

8.1.6 - Providing methods to make the most critical layer detected more fault-tolerant

8.1.7 - Experimenting with fault injection test to check that the most critical layers are protected correctly

8.1.8 - Testing of the whole system and preparing final project documents

8.2 Division of responsibilities and duties among team members

Tasks 1.1, 1.3 1.4, 1.5, 1.6, 1.7, and 1.9 will be shared equally between all the team members.

For task 1.2, we are going to implement 2 different neural networks which are ConvNet and AlexNet. Elif will be responsible for implementing ConvNet. Ozan and Furkan are going to implement AlexNet neural network mutual.

8.3 Timeline

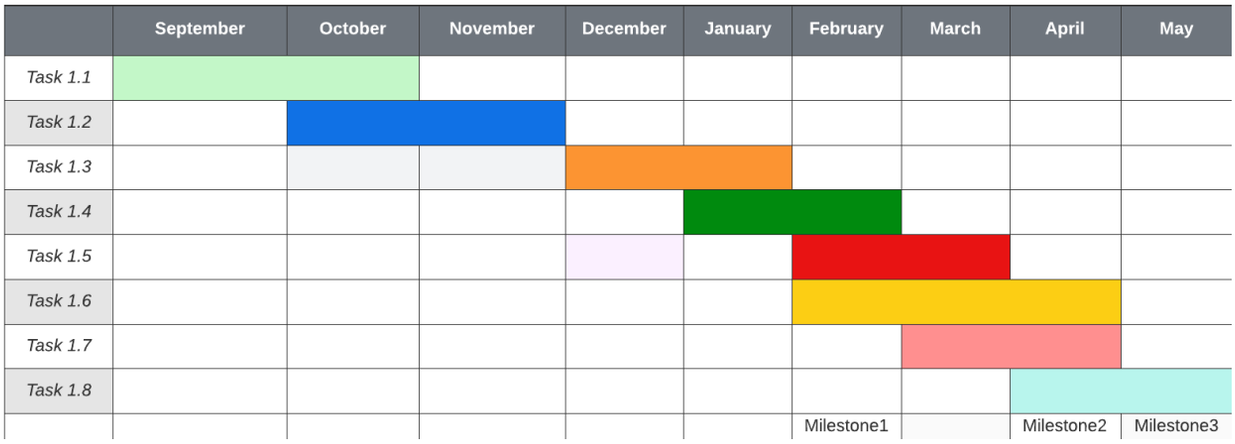

Figure 3: Gantt Chart with Milestones

9. Success Factors and Risk Management

A. Measurability/Measuring Success

The success factors for the objectives indicated in Section 3 are given below:

(i) <u>Success Factor for Objective 1:</u> When we test the neural network we created after completing the training process, if we can obtain a prediction rate of around 80% or greater than 80%, it can be said that this objective has been successful.

(ii) <u>Success Factor for Objective 2:</u> Our goal of making a fault injector is to corrupt the data by sending a large number of errors to each layer, and thus to identify the layer(s) that has the most impact on accuracy. For this reason, the success criteria of this objective is whether the fault injector we created can corrupt the data in the layers or not.

(iii) <u>Success Factor for Objective 3:</u> At this stage, after corrupting the data of each layer with the fault injector, we have to find the most critical layer(s) by evaluating the results of fault injections experiments. As a result of these experiments, if we can identify the layer(s) that affects the accuracy rate of our neural network the most, we can say that this objective has been successfully completed.

(iv) <u>Success Factor for Objective 4:</u> For this stage, our aim is to propose method(s) to protect the most critical layer(s) in our neural network. The proposed method(s) should reduce the error rate obtained in the previous step and present a similar accuracy rate to the original version of the neural network where the faults are not injected to the system. If the accuracy rate returns to its original state after applying this proposed method(s), it means that our method(s) has been successful.

B. Risk Management

Risk 1: Not being able to implement the fault injector we plan to use in order to corrupt the data in the layers.

- **Resolution:** If we have any problem implementing the fault injector, we can use one of the fault injectors publicly available. However, we aim to implement and use an application level fault injector. If we cannot find a suitable application level fault injector for our project, the scope of how the data in the layers will be corrupted may change.

Risk 2: While we aim to detect the most critical layer(s) after corrupting the data in the layers by injecting errors into them, we may not find any critical layer.

- **Resolution:** If we cannot find a critical layer in line with the results we obtained after our experiments, we can use another neural network architecture instead of the AlexNet and ConvNet. Then, we can try to find the most critical layer in this new neural network architecture by doing the same operations.

Risk 3: We may not be able to protect the most critical layer(s) that we have detected as a result of our experiments.

- **Resolution:** If we cannot do the protection at one level, we can try to do it at another level such as OS, hardware, assembly.

10. Benefits and Impact of the Project

Benefits/Implications:

The benefits of our project can be listed as follows:

- Our project will contribute to reducing the error rate of neural networks and obtaining more reliable results, as we will identify the layer that has the most impact on accuracy and propose methods to protect it.
- In safety critical systems, such as self-driving cars, a possible failure can cause catastrophic disasters. Our project will reduce the error rate in neural networks and ensure that such critical systems are more secure, and thus will help prevent catastrophic failures.

Academic researchers, companies that develop autonomous driving algorithms for cars, applications using the safety-critical DNN algorithms can benefit from our project.

The four type of impacts of our project are given below:

i. Scientific Impact:

We can say that our project will have a scientific impact. Since our project is mostly an academic study, we can send our findings to scientific conferences.

ii. Economic/Commercial/Social Impact: Since neural networks are used in many fields to help people with complex real-life problems, if we can increase the dependability of these networks, more reliable systems can be offered to people. For this reason, we can say that our project will indirectly increase the quality of life.

iii. Potential Impact on New Projects: Our project may have positive effects on new projects. If the most critical layer protection method(s) we proposed is used in the new projects, the error rate can be minimized with different methods.

iv. Impact on National Security: It can be said that our project has an indirect effect on security systems. For example, since neural networks are used in intrusion detection and prevention systems, the method we proposed to protect the most critical layer(s) can reduce the error rate and make such systems more reliable.

References

[1] Jiawang Bai, et al. Targeted Attack Against Deep Neural Networks Via Flipping Limited Weight Bits, Tsinghua Shenzhen International Graduate School, Tsinghua University, 2021.

[2] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, S. W. Keckler, Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications. SC17, November 12–17, 2017, Denver, CO, USA

[3] R. Leveugle, A. Calvez, P. Maistri and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," 2009 Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 502-506, doi: 10.1109/DATE.2009.5090716.

[4] F. Su, P. Yuan, Y. Wang, C. Zhang, The superior fault tolerance of artificial neural network training with a fault/noise injection-based genetic algorithm, Protein & Cell. 7 (2016) 735–748. Computing: Informatics and Systems. (2017).

[5] A.M. Sampaio, J.G. Barbosa, A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud, Sustainable, Computing: Informatics and Systems. (2017).

[6] Mandal, M. (2021, July 23). *CNN for deep learning: Convolutional Neural Networks*. Analytics Vidhya. Retrieved November 30, 2022, from https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/