Improving the Fault Resilience of Neural Network Applications Through Security Mechanisms

Nikolaos Deligiannis, Riccardo Cantoro, Matteo Sonza Reorda, Marcello Traiola, Emanuele Valea

Improving the Fault Resilience of Neural Network Applications Through Security Mechanisms

Nikolaos I. Deligiannis[1], Riccardo Cantoro[1], Matteo Sonza Reorda[1] Marcello Traiola[2] and Emanuele Valea[3]
[1]Department of Control and Computer Engineering, Politecnico di Torino, Corso Castelfidardo 39, 10129 Torino TO, Italy
[2]University of Rennes, Inria, CNRS, IRISA, Rennes, France. [3]Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France.
(e-mail: nikolaos.deligiannis|riccardo.cantoro|matteo.sonzareorda@polito.it, marcello.traiola@inria.fr, emanuele.valea@cea.fr)

***Abstract*—Numerous electronic systems store valuable intellectual property (IP) information inside non-volatile memories. In order to protect the integrity of such sensitive information from an unauthorized access or modification, encryption mechanisms are employed. From a reliability standpoint, such information can be vital to the system's functionality and thus, dedicated techniques are employed to detect possible reliability threats (e.g., transient faults in the memory content). In this paper we explore the capability of encryption mechanisms to guarantee protection from both unauthorized access and faults, while considering a Convolutional Neural Network application whose weights represent the valuable IP of the system. Experimental results show that it is possible to achieve very high fault detection rates, thus exploiting the benefits of security mechanisms for reliability purposes as well.**

I. Introduction

In the last years several safety-critical domains have been empowered by machine learning (ML) applications, such as autonomous driving, robotics and health [1]. In such ML-empowered systems, some vital information items such as the weights of the network are stored into non-volatile memories (NVMs) in an encrypted manner in order to protect them from unauthorized access and/or modification [2]. In this paper, we experimentally evaluate the positive effects that data encryption may have in terms of reliability enhancements with respect to possible transient faults. Specifically, we use a convolutional neural network (CNN) whose weights, which are the result of the network's training process, represent the IP of the system to be encrypted and stored into a NVM. The CNN we use as a case study is LeNet-5 [3], trained on the MNIST dataset of handwritten digits [4]. The cipher we use for the encryption purposes of the CNN's weights is the Advanced Encryption Standard (AES). AES has been used in various modes that are falling into two categories. The *spreading* category (i.e., if a fault is present then the decryption mechanism amplifies the fault effect by propagating it downstream) and the *non-spreading* category (where this does not happen). In the former, we use AES in counter mode (CTR) and in output feedback chaining (OFB) mode, while in

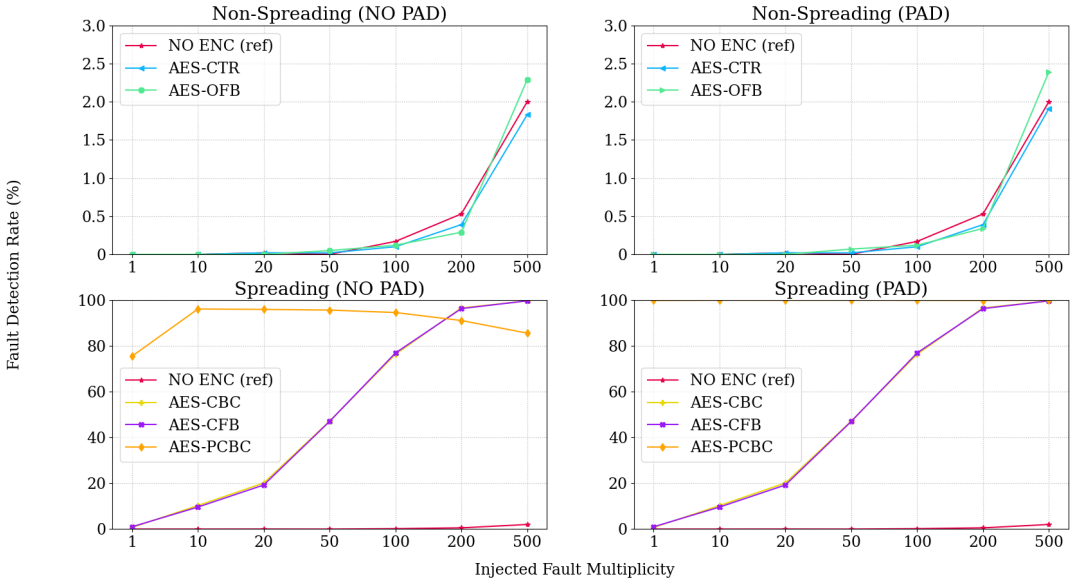

Fig. 1. Accumulative fault injection campaign results for SEU and MBU fault models

the latter we use AES in cipher block chaining (CBC) mode, cipher feedback (CFB) mode and in propagating cipher block chaining (PCBC) mode. Furthermore, we consider on the one hand cases where padding [5] is used during encryption and on the other cases where it is not.

II. Fault Model, Fault Injections and Results

We consider the *Single Event Upset (SEU)* and the *Multiple Bit Upset (MBU)* fault models for the purposes of our experiments. Specifically, for the latter we use fault multiplicities of 10, 20, 50, 100, 200 and 500. A fault injection campaign was performed for each fault model, for each multiplicity and for each AES configuration mentioned above. In order to obtain statistically meaningful results, the number of iterations for every case was calculated according to statistical fault injection metrics [6]. A fault is considered to be detected when (i) the fault affects the program execution by turning a valid floating point number into a NaN value and thus causes a *software exception*; (ii) the fault corrupts the padding bytes of the ciphertext and a padding check action detects the anomaly during the decryption of the weights.

Experimental results are plotted in Fig. 1. Regarding the non-spreading cipher configurations we observe that for both cases including padding there are no major improvements to the overall detection of the faults. When it comes to spreading cipher configurations though, we can clearly see that we have an improvement with respect to the no encryption scenario that we use as a reference. Specifically, for the case of the PCBC cipher and padding, we can see that we achieve significantly high ($\approx 100\%$) fault detection rates no matter the injected fault multiplicity. We present a comprehensive and more in-depth analysis of this work in [7]. Work is currently being done to extend this work by considering a wider variety of CNNs, while also accounting for the attribute of integrity.

References

[1] Y. LeCun *et al.*, "Deep learning," *Nature*, 2015.

[2] Tramèr Florian and others, "Stealing Machine Learning Models via Prediction APIs," in *25th USENIX Conference on Security Symposium*. USENIX Association, 2016.

[3] Y. LeChun, "LeNet-5," http://yann.lecun.com/exdb/lenet/.

[4] "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[5] B. Kaliski, "RFC2315: PKCS #7: Cryptographic Message Syntax Version 1.5," 1998.

[6] R. Leveugle and others, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009.

[7] N. I. Deligiannis *et al.*, "Towards the Integration of Reliability and Security Mechanisms to Enhance the Fault Resilience of Neural Networks," *IEEE Access*, 2021.