Practical Electromagnetic Fault Injection on Intel Neural Compute Stick 2

Shivam Bhasin, Dirmanto Jap, Prasanna Ravi
National Integrated Centre for Evaluation (NiCE)
Nanyang Technological University, Singapore
{sbhasin,djap,prasanna.ravi}@ntu.edu.sg

Marina Krček, Stjepan Picek
Radboud University
Nijmegen, The Netherlands
{marina.krcek,stjepan.picek}@ru.nl

***Abstract*—Machine learning (ML) has been widely deployed in various applications, with many applications being in critical infrastructures. One recent paradigm is edge ML, an implementation of ML on embedded devices for Internet-of-Things (IoT) applications. In this work, we have conducted a practical experiment on Intel Neural Compute Stick (NCS) 2, an edge ML device, with regard to fault injection (FI) attacks. More precisely, we have employed electromagnetic fault injection (EMFI) on NCS 2 to evaluate the practicality of the attack on a real target device. We have investigated multiple fault parameters with a low-cost pulse generator, aiming to achieve misclassification at the output of the inference. Our experimental results demonstrated the possibility of achieving practical and repeatable misclassifications.**

***Index Terms*—EMFI, Edge ML, Model Evasion**

I. Introduction

With the growth of IoT, there is also a growing demand to perform efficient ML computation on edge devices, focusing on smaller devices, like Apple neural engine and Qualcomm Hexagon DSP. However, edge deployment introduces novel threat vectors. Breier *et al.* [1] demonstrated the first practical fault attacks targeting the activation function of a deep neural network. These experiments were performed on a simple 8-bit microcontroller and restricted to very small neural networks. A recent work [2] used methods like Rowhammer to inject faults in neural network inference running on cloud servers. The authors exploited bit flips introduced by Rowhammer to degrade the accuracy of neural networks. Even Edge Ml devices like NCS2 have been subjected to side-channel [3] and cold boot attacks [4]. To our knowledge, none of the works explore the fault vulnerability of commercial edge ML devices. *This work presents a practical EMFI study on NCS 2, as well as empirical evidence of misclassification during inference, as a proof-of-concept to show that these attacks can also be applied practically on commercial devices.*

A. Target Device

NCS 2 is a compact, plug-and-play development kit designed to accelerate neural network inference. It is based on the Intel Movidius Myriad X architecture that provides high performance and low power consumption for ML applications. OpenVINO is a Python toolkit to optimize deep learning models on powerful systems like the cloud for deployment on edge devices[1]. The trained model will be converted to OpenVINO format and loaded to the device.

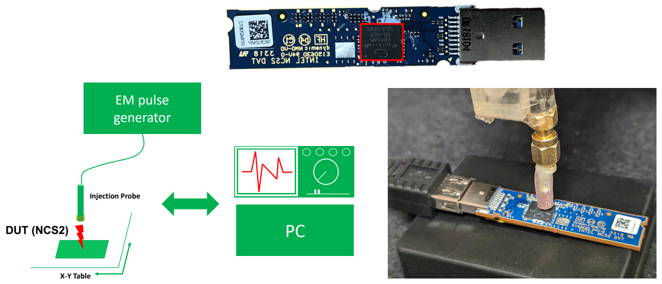

Fig. 1: ChipShouter setup: (top) Intel NCS 2, (left) Scheme for EMFI, (right) EM probe on NCS 2

B. EM Fault Injection (EMFI)

EMFI injects a very high and short voltage pulse directly onto the surface of the chip using a near-field EM probe. Since the EM probe can be placed to target different spatial locations on the chip, EMFI could be used to precisely target different sub-blocks within the target. EMFI does not require chip decapsulation, only removing a plastic case from the NCS 2, for closer proximity to the probe.

For some applications, it is desirable to inject multiple consecutive EM pulses in a burst. For instance, it was shown in [2] that to achieve successful misclassification in ResNet20 when trained on CIFAR-10, 21 bit flips are required. Thus, we also consider *burst mode* in our experiments, where multiple EM pulses are injected sequentially on the target device.

II. Experimental Setup and Observations

We aim to induce misclassification by making the model predict an incorrect class (untargeted). EMFI is performed in a black-box setup with no injection optimizations based on the model implementation. We have trained a custom Convolutional Neural Network (CNN) model on the MNIST dataset, based on the example from OpenVINO, which covers the standard layers used in CNN and is small enough to fit into the device. For this experiment, MNIST is chosen due to its simplicity for analysis purposes. The dataset and the network

[1]https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html

Fault Type		Value at Output Layer									
Fault I	Correct	0.00	0.00	0.00	0.01	0.00	**0.98**	0.00	0.00	0.00	0.00
	Faulty	0	0	**1**	0	0	0	0	0	0	0
Fault II	Correct	0.00	0.00	0.00	0.01	0.00	**0.98**	0.00	0.00	0.00	0.00
	Faulty	0.09	0.02	0.06	**0.34**	0.00	0.24	0.01	0.04	0.03	0.15

TABLE I: The changes observed at the *SoftMax* output

are chosen as a proof-of-concept to explore how practical EMFI is in the context of a practical real target.

We use Raspberry Pi (RPi) 3B+ as the main controller to load the CNN model and communicate with NCS 2. We set up UART communication to synchronize the fault parameters and errors. We set the GPIO pin in RPi as the trigger to indicate the start of inference. Due to noise from surrounding operations and operating system that causes desynchronization, the inference processes are not in constant time.

We use ChipShouter pulse generator, a low-cost EMFI tool. To control the pulses, we connect it to a ChipWhisperer (CW) to generate the pulse pattern. The pulse width can be expressed as a fraction of the CW clock cycle (around 2-3 μs), and for pulse delay, it can be expressed up to 2^{32} clock. Even with non-constant timing, up to $25,000$ cycles are sufficient to cover the whole inference. From preliminary profiling of the device, we found no error obtained beyond this parameter.

Based on the size of the chip and the EM probe, we scan the whole chip with a 7×7 grid to find vulnerable positions. Our ideal target operation is at the last layer of inference, mainly before or during the *SoftMax* function. Since our trigger points come from the host RPi, we do not have precise control over the fault timing. The black box nature of NCS 2 also makes it difficult to pinpoint the exact timing of *SoftMax*. Thus, we scan the whole inference phase with a burst of $50\times$ as a way to cover the execution of this operation.

We performed prior profiling to identify parameters that result in faults. The minimum required pulse voltage is $500V$, with a fixed pulse width of 45 clocks and delay between $5,000$ and $25,000$ clocks, with a step size of $1,000$ clocks. A total of 10 million injections were performed. Most faults are failure errors, such as segmentation fault, failure to load model to device, *etc*. We have observed errors at *SoftMax* outputs, which only result in minor changes in decimal points. However, we also received some successful misclassifications. We record the fault parameters and test the repeatability of these faults. We performed statistical analysis done over 100 repetitions. Overall, 47% of the injections do not lead to a successful fault, 32% lead to a system error, and 21% of faults result in a successful misclassification.

A. Fault Analysis

We observed that the successful faults fall under two distinct categories, *Fault I* and *Fault II*. *Fault I* results in integer outputs, and *Fault II* leads to random values, both resulting in misclassification. We verified that the fault is repeatable for a fixed position of the probe. Examples are shown in Table I.

- For *Fault I*, all the changes are in the first fully connected layer, for example some values are changed from -1.5, -2.1, and -2.9 into 2046, 2044, and 2047, respectively. These changes propagate to subsequent layers and eventually change the *SoftMax* input, giving a very high value for one of the classes, resulting in (untargeted) misclassification.
- For *Fault II*, most changes are in the output of *ReLU* from the second convolution layer. The average change of these faults is 1.06, and the maximum is 6.09, meaning significant numbers of the values are changed, propagating them to the *SoftMax* layer.
- There are also faults that cause minor changes to the mantissa, with no misclassification. The changes occur at the very first convolution layer, with around 300 entries affected. However, the changes are small, and the neural network is able to correct them during the propagation.

III. Discussion and Conclusion

One of the main challenge is that the architecture implemented in NCS 2 is done in black box manner, where the attacker might have the knowledge regarding the model, but not on how the model is implemented on the device itself. The presence of desynchronization also makes it harder for the attacker to precisely inject the fault in the desired timing, achieving lower repeatability. Nevertheless, we have shown that we can achieve misclassification for the prediction if the fault(s) can be injected at the desired location with precise timing, posing a real-world risk. Thus, we have demonstrated the practicality of a fault injection attack targeting a commercial target as a proof-of-concept, which helps bridge the gap between practical and theoretical fault injection attacks. For future work, we would like to investigate the scalability of EMFI on different target models on more complex platforms.

Acknowledgments

This work was (in part) supported by Dutch Research Council (NWO) through the Challenges in Cyber Security (CiCS) project of the research programme Gravitation under the grant 024.006.037.

References

[1] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2204–2206.

[2] F. Yao, A. S. Rakin, and D. Fan, "{DeepHammer}: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1463–1480.

[3] Y.-S. Won, S. Chatterjee, D. Jap, S. Bhasin, and A. Basu, "Time to leak: Cross-device timing attack on edge deep learning accelerator," in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, pp. 1–4.

[4] Y.-S. Won, S. Chatterjee, D. Jap, A. Basu, and S. Bhasin, "Deepfreeze: Cold boot attacks and high fidelity model recovery on commercial edgeml device," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.