# Fault Injection Attacks Using Clock Glitch Against Deep Neural Networks and Their Countermeasures

Seongwoo Hong, Hyojoo Kang, Youngjoo Lee and Jaecheol Ha*

Hoseo University, Asan-si, Chungcheongnam-do, South Korea
{hshsw5660, gywn7597}@gmail.com,
lyj04281@naver.com, jcha@hoseo.edu

**Abstract**

Deep learning models have often been adopted to recognize input images in edge devices. However, edge devices that implement a Deep Neural Network (DNN) are physically accessible to a malicious attacker. In particular, fault injection attacks on these devices can lead to misclassification of an image and can cause serious accidents. In this paper, we introduce a clock glitch, a typical form of fault injection attack, into a device running a DNN model to induce misclassification. As a result of our experiment on a microprocessor implemented with a DNN for image classification, misclassification can be achieved by appropriately injecting clock glitches. We propose two countermeasures specifically against fault injection attacks: One is to check the number of loops in the softmax function, and the other is to accumulate the normalized output values and then check this result.

**Keywords**: Edge Device Security, Fault Injection Attacks, Deep Neural Networks, Misclassification on DNN

## 1 Introduction

Recently, deep learning technologies can be used for image recognition in real life such as self-driving cars, smart homes, IoT applications, and so on. Nevertheless, there are implementation vulnerabilities in Deep Neural Network (DNN) such as Multiple Layer Perceptron (MLP) and Convolutional Neural Network (CNN). A DNN implemented for image recognition can induce

The 7th International Conference on Mobile Internet Security (MobiSec'23), Dec 19-21, 2023, Okinawa, Japan, Article No.26
* Corresponding author: Division of Computer Engineering, Hoseo University, Asan-si, Chungcheongnam-do, 31499, South Korea, Tel: +82-41-540-5991

misclassification of the original input image by fault injection during network processing [1-3]. Such malicious image misclassification may cause serious accidents to vehicle systems and may expose personal information.

In particular, remote edge devices implemented with DNN can be easily accessed by attackers, so physical attacks such as side-channel attacks or fault injection attacks are realistically possible. Fault injection attacks (a type of physical attack) use methods such as clock glitches, voltage glitches, or laser radiation to cause malfunctions while the device is operating. The main purpose of fault injection attacks attempted in the past was to extract the secret key by injecting faults into the cryptographic circuit [4], [5]. On the other hand, several attacks have recently been proposed to induce misclassification by applying fault injection attacks on DNNs [6]-[9].

Although Liu et al. provided a white-box attack on deep neural networks based on software simulations, their attack failed to provide practical results in real implementation environments [6]. Zhao et al. proposed a framework to inject stealthy faults for selected inputs while keeping overall test accuracy performance [7]. Bereir et al. initiated the practical study of leveraging fault injection attacks, by using a laser injection technique on an embedded device. Their results showed that it is possible to achieve a misclassification by injecting faults into the hidden layer of the DNN [8]. Fukuda et al. focus on the softmax function of the output layer and evaluate the fault injection attack performance using clock glitches [9]. As a result of performing a fault injection attack on a DNN implemented on an 8-bit microprocessor, an attacker can achieve misclassification of input images.

In this paper, we experimentally confirmed that injecting a clock glitch into a DNN implemented in an embedded system using software can induce misclassification with a high probability. In particular, fault injection targeting the softmax activation function of the output layer was effective in inducing misclassification. In addition, we present two countermeasures against fault injection attacks. One is to check the number of repetitive loops when executing the algorithm, and the other is to check the results of accumulating normalized output values.

This paper is organized as follows. Chapter 2 provides preliminaries on Deep Neural Networks (DNN) and clock glitch operation. In Chapter 3, we investigate the weak algorithmic points regarding the attack model, discuss the experimental setup, and analyze the results. Chapter 4 introduces countermeasures against such attacks, and Chapter 5 concludes the paper.

# 2 Preliminaries

## 2.1 Deep Neural Network

Deep learning utilizes artificial neural networks with multiple hidden layers to learn complex patterns in data. DNNs have shown remarkable performance in research fields such as image recognition and natural language processing. Among various DNN architectures, Multi-Layer Perceptron (MLP) is becoming the basic type. One MLP consists of an input layer, one or more hidden layers, and an output layer. The basic structure of an MLP model for classification tasks is shown in Fig. 1.

The computation in each neuron of DNN for classification is depicted in the following equation. Here, x represents the input, w stands for the weight, b is the bias, f denotes the activation function, and y is the output value.

$$\boldsymbol{y_k} = f\left(\sum_{i=0}^{n-1}\left(w_{i,k} \cdot x_i\right) + b_k\right) \qquad (1)$$

The activation function transforms the summed input signals into an output signal. Among the various activation functions, the softmax is commonly used as the activation function of the output layer of a classification model. The softmax function performed at each node is shown in Eq. (2).

$$\boldsymbol{O_k} = \boldsymbol{s}(\boldsymbol{y_k}) = \frac{\boldsymbol{e^{x_i}}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \boldsymbol{e^{x_j}}} \quad (2)$$

Where x represents the input of the softmax function, and n is the number of classifications. The softmax function normalizes the output for each neuron, ensuring that the sum of the output values equals 1. Based on this characteristic, the classification model interprets the softmax output as the predicted probability for each class. In this study, we focus on the softmax function, specifically its output. Some glitches on the final output node can be meaningfully impacted, allowing attackers to conduct attacks that lead to misclassifications.

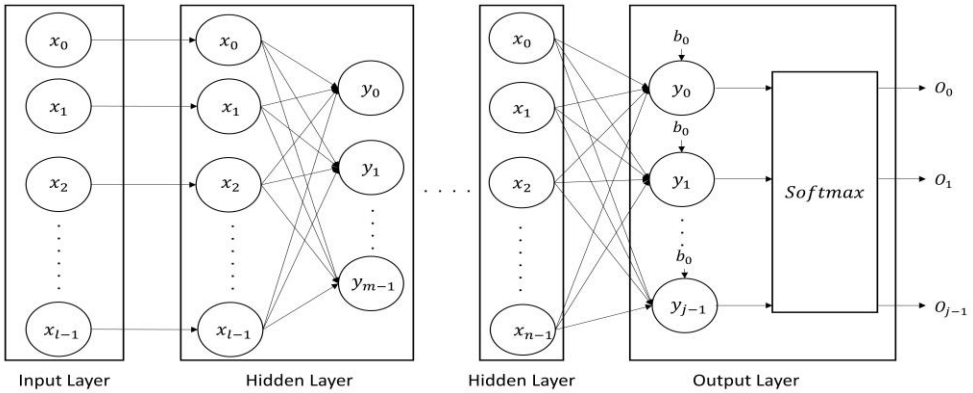



**Figure 1: Structure of MLP**

## 2.2 Clock Glitch

A fault injection attack maliciously introduces faults into a device to induce specific malfunctions or anomalies. These attacks can be classified into various categories based on the types, effects, and durations of the faults. The types of faults include clock glitches, voltage glitches, laser insertions, and others. In this paper, we adopt an attack method that induces misclassification by injecting clock glitches proposed by Fukuda et al. [9]

A clock glitch is an attack that disrupts the normal operation of a digital system, most notably a computing device such as a microcontroller or FPGA, by injecting a temporary abnormal clock into the timing of a clock signal [10]. Fig. 2 explains the basic principle of the fault injection attack using a clock glitch.

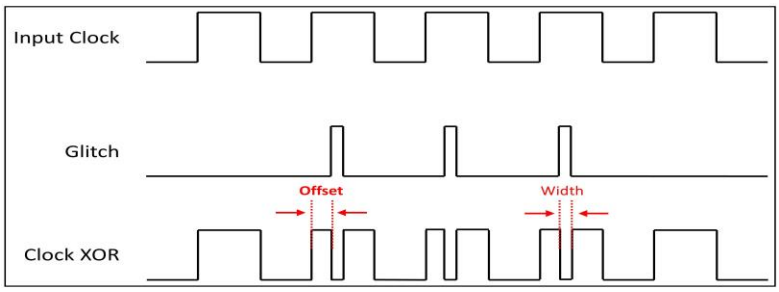



**Figure 2: Fault injection attack using clock glitch**

The system clock is the key signal that allows the various components in a digital circuit to operate in synchronization. The operational result of the combinational circuits is recorded in registers at the rising edge of the clock. If a clock glitch is injected before the arithmetic operations of combinational circuits are completed, then a malfunction is induced by storing the incorrect result of an operation in the register.

# 3 Fault injection attacks using clock glitch

## 3.1 Attack Model on Softmax Function

We assume that an attacker can physically access the target device to inject clock glitches. The structure of the target DNN model is already known to the attacker. The primary objective of the attacker is to induce misclassification in the DNN model by applying clock glitches to the DNN accelerator.

A strategic point for the insertion of these glitches is the softmax function, which serves as the activation function of the output layer. The softmax function is the most commonly used in the output layer for multiclass classification for input datasets such as MNIST digit images.

The softmax function finally outputs the probabilities for each class. Since the softmax function is closest to the final output, it is especially susceptible to even a small number of glitches, which significantly affects the DNN performance. The softmax function can be implemented in the microcontrollers based on Algorithm 1.

**Algorithm 1: *Softmax* function**

Input: $y(y \in \{y_0, \dots, y_{J-1}\})$

Output: $O(O \in \{O_0, \dots, O_{J-1}\})$

```
1:  for (k = 0; k < J; k++) {
2:      O_k ← 0
3:  }
4:  sum ← 0
5:  for (k = 0; k < J; k++) {          ← Attack Point
6:      S_k ← exp(y_k)
7:      sum ← sum + S_k
8:  }
9:  for (k = 0; k < J; k++){
10:     O_k ← S_k / sum
11: }
```

During the first iteration of the softmax function, when $k = 0$, the loop proceeds normally. However, as index k becomes 1, a clock glitch is injected in line 5 of algorithm 1, causing the iteration to be skipped. That is, the attacker skips the loop process of lines 6-7 at the timing when $k = 1, 2, 3, \dots J -$

1. Then, all array values except $S_0$ will be 0. In an attack like this, only the final $O_0$ value becomes 1 and it is always classified as the first class regardless of the values of the input.

Secondly, we assume that the fault injects into the process of 6-7 lines in Algorithm 1. This is calculated by accumulating the values of the exponential operation for each class. This fault injection attack can calculate inaccurate intermediate values $S_k$ or incorrect accumulated value, which can lead to misclassification.

Thirdly, we discuss the possibility of a fault injection attack at 9 lines in Algorithm 1. These processes calculate the final classification possibility by dividing the value of the exponential operation for each class by the accumulated value. In this case, the divisional operation is not processed properly after the faulty clock glitch is injected. Ultimately, this attack can also lead to misclassification.

We consider that attacks changing the intermediates values such as $S_k$, variable sum and $O_k$ cannot induce the misclassification intended by the attacker. Therefore, we will demonstrate the fault injection experiment at the loop skip method at line 5 of Algorithm 1. In fact, this attack experiment can lead to misclassification with a high probability.

## 3.2 Experiment Setup

The target DNN model for our fault injection attack is an MLP that performs MNIST handwritten digit classification. Since this classification takes input digit images from 0 to 9 and predicts the number, then the number of classes is 10. The target MLP consists of an input layer, three hidden layers, and an output layer. After the training process using the MNIST dataset, we implement the MLP for test image classification on a microprocessor. The detailed structure of the MLP used in our experiment is shown in Table 1.

| Layer | Number of Neurons | Activation function |
|---|---|---|
| Input Layer | 784 | - |
| Hidden Layer – 1 | 128 | ReLu |
| Hidden Layer – 2 | 64 | ReLu |
| Hidden Layer – 3 | 32 | ReLu |
| Output Layer | 10 | Softmax |

**Table 1: Structure of experimental MLP model for MNIST dataset**

To evaluate the possibility of fault injection attacks, we conducted our experiments on the Chipwhisperer platform developed by New AE technology, which includes the CW-Lite (cw-1173). Furthermore, our experiments are targeting the CW308 UFO Target Board, which adopts the ARM Cortex-M4 STM32F303 microcontroller. The softmax algorithm in the MLP model is implemented on this target microcontroller. The photograph of the experimental setting is shown in Fig. 3.

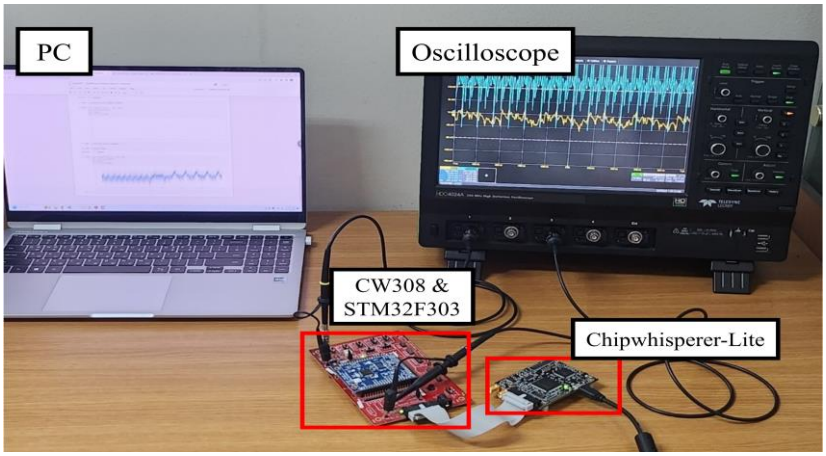



**Figure 3: Photograph of the experimental setting**

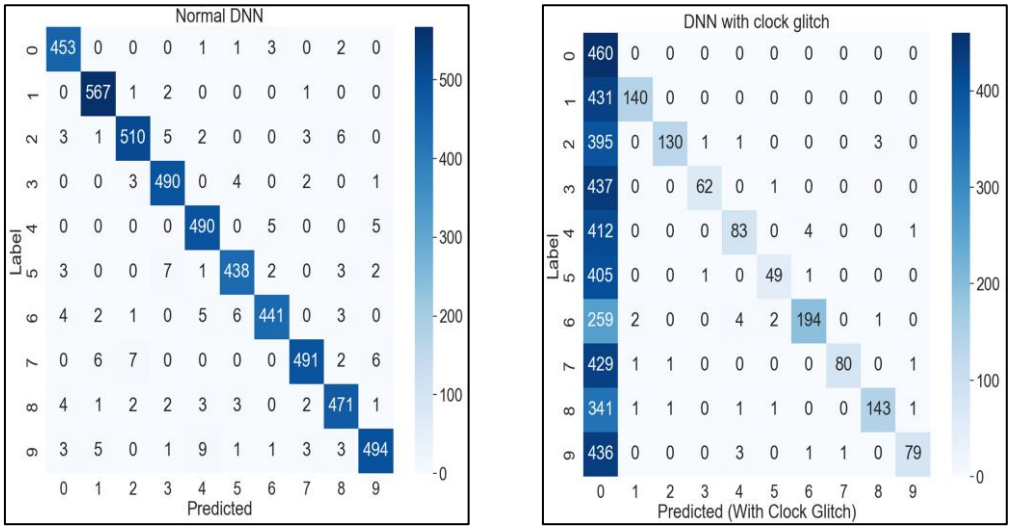
The target device runs the MLP program with a clock frequency of 7.38MHz. In the practical clock glitch attack, we set the 'Repeat' value to 5, allowing for glitch insertion to be attempted five times. Additionally, the 'offset' and 'width' were set to -38 and -10, respectively. The detailed shape of the clock glitch used in our experiment is described in Table 2.

| Clock frequency | 7.38 MHZ |
|---|---|
| Glitch Repeat | 5 |
| Glitch Offset | - 38 |
| Glitch Width | - 10 |

**Table 2: Parameter setting of experimental clock glitch**

## 3.3 Experimental Results

The results of the MNIST classification using the microcontroller are shown in Fig. 4. While the accuracy of the test images in the normal operation was 96.9% shown in Fig. 4. a), the accuracy under the clock glitch attack phase drops to about 28.4% shown as in Fig. 4. b), Both classifications were conducted on the 5,000 MNIST digit images as a test dataset. The goal we attempted in the fault injection attack was to perform only the first loop at the point of attack (line 5 of Algorithm 1) and skip the remaining iterations. Since the 80.1%(=4,005/5,000*100%) of all test images are classified into number 0. Since the purpose of the attack was to induce the DNN to classify any input as '0', any naturally occurring '0' labels in the test dataset make it challenging to ascertain the true success of the attack. Excluding those naturally occurring '0' values, the success rate of our fault attack is 78.1%(=3,545/4,540*100%).




**a) DNN classification in normal operation** **b) DNN classification under fault attack**

**Figure 4: Classification results for the MNIST dataset**

# 4 Countermeasures Against Fault Injection Attacks

The primary objective of fault injection attacks on the softmax activation function in the output layer is to induce misclassification or malfunction within the DNN model. To accomplish this attack goal, an attacker injects intentional faults into the counter values in looped calculation or variables to store intermediate calculation values such as $S_k$, and $O_k$. Considering these attack points, this paper proposes a countermeasure that integrates two check conditions to ensure the correct operation of softmax function.

## 4.1 Checking Loop Counter

The first countermeasure is to check whether the correct loop is performed. As shown in Algorithm 1, the number of loop iterations should correspond to the number of output nodes. So, if we have J output nodes, the final loop counter we need to perform must have a value equal to J. If this condition is satisfied, then the next processes are continued. Otherwise, the DNN model will stop working to prevent potential malfunctions. Given that fault injection attacks on the loop statement can occur at two distinct points during the execution of the softmax function, it is essential to check the number of iterative operations after each of these loops is completed.

## 4.2 Summation of Classification Possibilities

Nevertheless, an attacker could still induce faults into variables to store intermediate values. This attack can also lead to misclassification of images. In this case, even if we add the classification probabilities of all output nodes, the value will not become 1. We propose a function to check the accumulated value based on the fact that the summation of the normalized probability values of the final output layer becomes 1. If this condition is not satisfied, then the next processing is stopped. Otherwise, the DNN model will return the classification possibilities as the final output.

The following Algorithm 2 shows the secure softmax algorithm that complies with both check conditions. Consequently, we confirm that the proposed softmax algorithm can counteract against fault injection attacks.

**Algorithm 2: Countermeasure of $Softmax$ function**

Input: $y (y \in \{y_0, y_1, \dots, y_{J-1}\})$
Output: $O (O \in \{O_0, O_1, \dots, O_{J-1}\})$

```
1:  for (k = 0; k < J; k++) {
2:      O_k ← 0
3:  }
4:  sum ← 0, check ← 0
5:  for (k = 0; k < J; k++) {
6:      S_k ← exp(y_k)
7:      sum ← sum + S_k
8:  }
9:  if (k != J) {print "ERROR-1", return 0}
10: for (k = 0; k < J; k++) {
11:     O_k ← exp(y_k)/ sum
12: }
13: if (k != J) {print "ERROR-2", return 0}
14: for (k = 0; k < J; k++) {
15:     check ← check + O_k
16: }
17: if (k != J) {print "ERROR-3", return 0}
18: if (check != 1) {print "ERROR-4", return 0}
```

# 5 Conclusion

Recently, deep learning technology has been widely used in the fields of facial recognition and image detection in smart homes and IoT-based edge devices. Nevertheless, when a DNN is executed on edge devices, some physical attacks such as side-channel analysis or fault injection should be considered.

In this paper, we demonstrated that a clock glitch attack, a variant of fault injection attacks, can be effectively utilized to achieve misclassification of MNIST digit images. Our target algorithm for fault injection attack is the softmax function of the output layer. The DNN model for the classification of input digit images is implemented on the STM32F303 microcontroller. By injecting a clock fault after running the first loop process, 78.1% of the test data was misclassified as an adversarial class rather than a true class. Additionally, we proposed two novel algorithmic countermeasures designed for the secure execution of the softmax function under fault injection attacks. One is to check that the number of loop iterations is equal to the number of output nodes. The other is to verify that the accumulated value of classification possibilities is 1. As a result, the combination of these countermeasures can ensure the security of devices performing DNN from fault injection attacks.

# Acknowledgment

This work was supported by a project for Smart Manufacturing Innovation R&D funded Korea Ministry of SMEs and Startups in 2022. (Project No. RS-2022-00140535)

# References

[1] Y. Liu, L. Wei, B. Luo and Q. Xu, “Fault injection attack on deep neural network”, IEEE/ACM International Conference on Computer-Aided Design, pp. 131-138, 2017.
[2] A.S. Rakin, Z. He, and D. Fan, “Bit-flip attack: Crushing neural network with progressive bit search”, IEEE/CVF International Conference on Computer Vision, pp.2322-2330, 2019
[3] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş, “Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks”, USENIX Conference on Security Symposium, pp.497-514, 2019
[4] D. Boneh, R.A. DeMillo, and R.J. Lipton, “On the importance of checking cryptographic protocols for faults”, Advances in Cryptology EUROCRYPT, pp.37–51, 1997.
[5] E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems”, Annual international cryptology conference, pp.513–525, 1997
[6] W. Liu, C.-H. Chang, F. Zhang and X. Lou, “Imperceptible misclassification attack on deep learning accelerator by glitch injection”, ACM/IEEE Design Automation Conference, pp. 1-6, 2020.
[7] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, “Fault sneaking attack: a stealthy framework for misleading deep neural networks”, ACM/IEEE Design Automation Conference, pp.1-6, 2019
[8] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin and Y. Liu, “Practical fault attack on deep neural networks”, ACM Conf. Comput. Commun. Security, pp. 2204-2206, 2018.
[9] Y. Fukuda, K.Yoshida and T. Fujino, “Fault Injection Attacks Utilizing Waveform Pattern Matching against Neural Networks Processing on Microcontroller”, IEICE Transactions, vol. 105, no. 3, pp. 300-310, 2022
[10] C. Kim and J. Quisquater, “Faults injection methods, and fault attacks”, IEEE Design & Test of Computers, vol. 24, no. 6, pp.544-545, 2007.