A Countermeasure to Glitch-based Fault Injection Attacks on Deep Neural Networks

Seongwoo Hong , Hyoju Kang , Youngjoo Lee , Gwangyeol Kim , Jaecheol Ha *

Posted Date: 4 September 2024

doi: 10.20944/preprints202409.0373.v1

Keywords: Sensor network; Deep neural network; Image classification; Hardware security; Fault injection attack

Article

A Countermeasure to Glitch-Based Fault Injection Attacks on Deep Neural Networks

Sengwoo Hong [1], Hyoju Kang [1], Youngju Lee [1], Gwangyeol kim [2] and Jaecheol Ha [1,*]

[1] Dept. of Information Security, Hoseo University, Asan-Si, ChungNam-Do, Rep. of Korea
[2] Sinsiway Inc., Songpa-Gu, Seoul-Si, Rep. of Korea
* Correspondence: jcha@hoseo.edu; Tel.: +82-41-540-5991

Abstract: Recently, deep neural networks (DNNs) have been widely used in various fields such as autonomous vehicles and smart homes. Since these DNNs can be directly implemented on edge devices, they offer advantages such as real-time processing in low-power and low-bandwidth environments. However, deployment of DNNs in embedded systems, including edge devices, exposes them to threats such as fault injection attacks. In this paper, we introduce two methods to induce misclassifications by using clock and voltage glitch-based fault attacks in devices where DNN models are executed. As a result of experiments on a microcontroller implemented with a DNN for image classification, we show that glitch injection attacks can lead with high probability to serious misclassifications. Furthermore, we propose a countermeasure to glitch attacks on the softmax function, and confirm that this method is effective in preventing misclassifications.

Keywords: sensor network; deep neural network; Image classification; hardware security; fault injection attack

1. Introduction

Recently, deep learning has been applied to various real-life applications such as autonomous vehicles, smart homes, and IoT applications. However, these deep neural network (DNN) implementations are easily exposed to threats from malicious attackers when operating on systems far from the administrator, such as edge devices. An example of such a threat is when adversaries intentionally alter input data to induce misclassifications in DNNs designed for image recognition [1–3]. Such intentionally caused misclassifications of images can lead to serious accidents in environments like autonomous vehicles. Furthermore, migrating DNN computations from servers or the cloud to edge devices makes threats such as fault injection and side-channel attacks quite possible [4–6].

Many of the fault injection attacks developed so far that involve glitches [7,8], lasers [9], electromagnetic (EM) pulses [10], and other methods to influence device operation have been used to introduce faults into encryption systems. This allows adversaries to analyze faulty output and extract secret information stored inside the devices [11,12]. Recently, various attacks have been proposed to launch fault injection attacks against DNNs to induce misclassifications [13–17].

Liu et al. proposed a white-box attack on DNNs that is based on software simulation [13]. However, it did not provide practical results in real implementation scenarios. Zhao et al. proposed a framework for injecting stealth faults into selected input while maintaining the overall test accuracy [14]. Practical research by Bereir et al. using laser injection techniques on embedded devices to conduct fault injection attacks was conducted to demonstrate the possibility of misclassification by injecting faults into the hidden layers of DNNs [15]. Alam et al. proposed remote fault attacks utilizing vulnerabilities in the RAM of FPGAs, demonstrating the ability to induce timing faults and bit flips [16]. Fukuda et al. evaluated attack performance from faults caused by injecting clock glitches into the softmax function of a DNN output layer, successfully misclassifying input images processed on an 8-bit microcontroller [17].

In this paper, we experimentally confirm that injecting a clock or voltage glitch into a DNN on an edge device can lead with high probability to misclassifications. Since this glitch attack on a softmax function of the DNN was launched in the last layer, it was much more effective than attacks on other activation functions.

 doi:10.20944/preprints202409.0373.v1

In order to detect fault injection attacks and shut down devices in an attack situation, we propose a countermeasure to glitch attacks on the softmax function. The main idea of the countermeasure includes two factors: one is verifying the number of loops during softmax execution, and the other is ensuring that cumulative normalized output values equal 1. As a result of experiments on microcontrollers implemented with image classification DNNs, we confirmed that our countermeasure can completely prevent misclassification from attacks on the softmax function.

This paper is structured as follows. Section 2 describes the DNN structure and the fault injection attacks. In Section 3, we analyze the softmax function that is the target of the attack, and observe possible attack points. Section 4 describes experimental settings on the microcontroller, and the results from fault injection via glitch-based fault attacks on a DNN. Section 5 explains our countermeasure's principles, and presents experimental results when applying it in the targeted board. Section 6 concludes the paper.

2. Materials and Methods

2.1. The DNN Structure

DNNs are artificial neural networks with multiple hidden layers that learn complex patterns in data. They are widely used in areas such as image recognition and natural language processing because they can learn complex and nonlinear data patterns using a network of neurons. The basic form of the DNN, multilayer perceptron (MLP), consists of an input layer, hidden layers, and an output layer. The structure of the MLP model is shown in Figure 1.

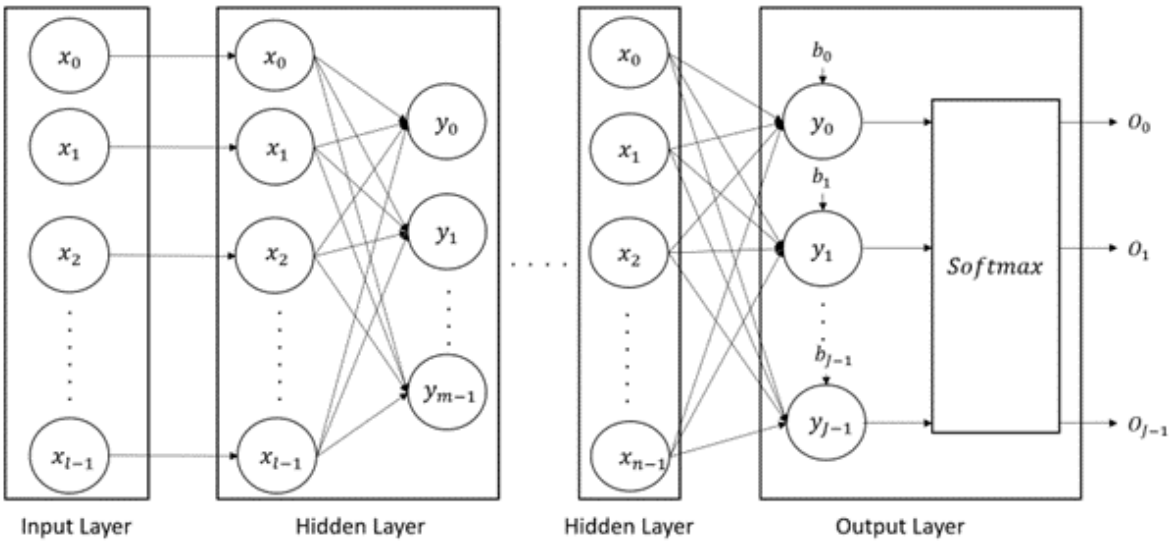


Figure 1. Structure of the MLP Model

Activation functions used in DNNs are functions that nonlinearly transform the sum of input signals at each neuron in the neural network. These activation functions are essential in order for the network to learn and classify complex data patterns. The representative activation functions include sigmoid, tanh, ReLU, and softmax. Softmax is commonly used in the output layer of DNN models that process multiple classes by transforming output values into probabilities for each class. That is, each output represents the probability of belonging to a certain class.

The formula for calculating input value y_i of the $i-th$ neuron in the softmax function is shown in Equation (1). Here, J represents the number of classes, indicating the number of neurons in the output layer.

$$O_i = Softmax(y_i) = \frac{exp(y_i)}{\Sigma_{k=0}^{J-1} exp(y_k)} \tag{1}$$

 doi:10.20944/preprints202409.0373.v1

Since the softmax function normalizes the classification probability of the image, it is characterized by the sum of each output being 1. Based on these special properties, classification models can interpret softmax output as the predicted probability for each class. This study focuses on the softmax function commonly used in the output layer of most classification models. Meaningful injection of malicious faults into the final output nodes of an image classification DNN can lead to misclassification.

2.2. Fault Injections

In a fault injection attack, a fault to induce a malfunction is inserted into a device. These attacks can be used to compromise the stability of a system or to extract secret data. Fault injection attacks can be conducted through various methods, including glitches in the clock or the voltage of the target device. In this study, we conduct glitch-based fault injection attacks using clock or device voltage disruption, both of which are relatively low-cost but highly effective.

In general, the clock controls the timing of operations in a computing device. If abnormal behavior occurs in this clock signal, instructions may not work properly. Furthermore, these faults may lead to abnormal termination or incomplete use of memory. A malicious attacker can induce abnormal operations by injecting clock glitches while the target device is running. Figure 2 illustrates the effect of a clock glitch on pipeline instructions.

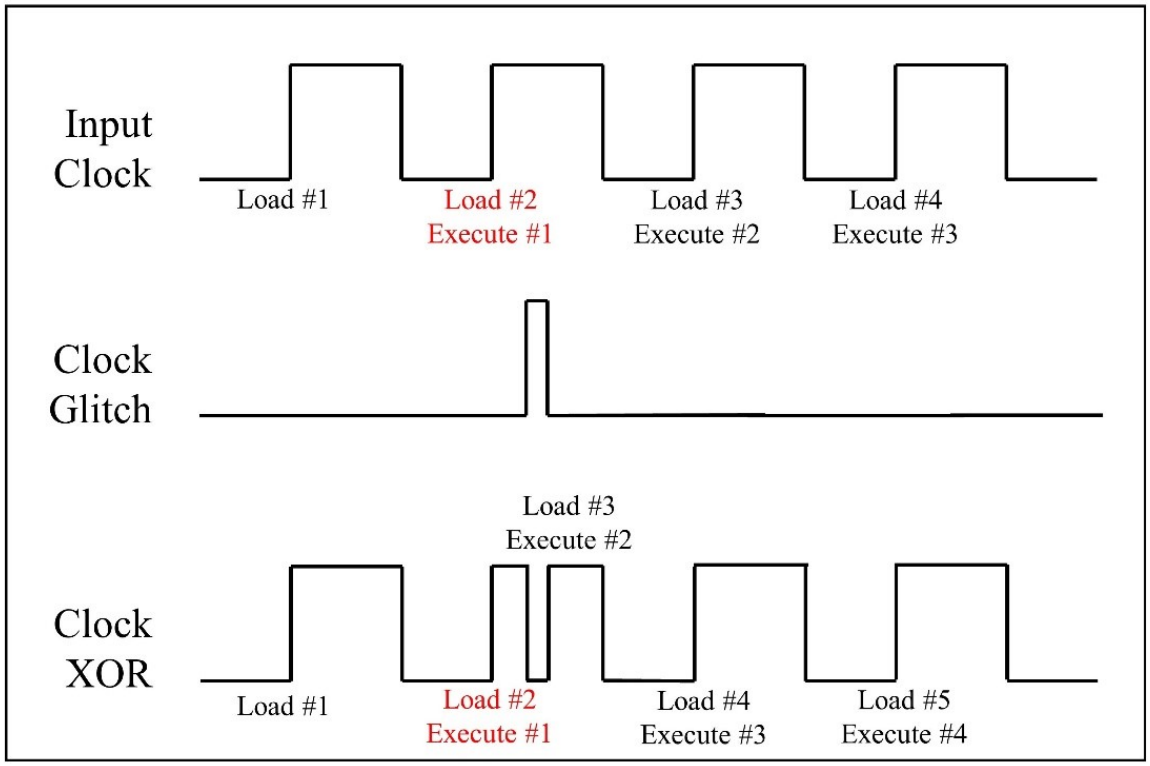


Figure 2. The Effect of a Clock Glitch on Pipeline Instructions

We can also use a voltage glitch that injects a fault causing high or low voltage on the power line of a device. Several parameters for a voltage glitch attack are shown in Figure 3. Like the clock glitch attack, this can lead to a malfunction in the instructions and incomplete use of memory. However, compared to a clock glitch, the voltage-based glitch attack is more sensitive to the attack environment, and requires additional work to accurately select the attack point.

 doi:10.20944/preprints202409.0373.v1

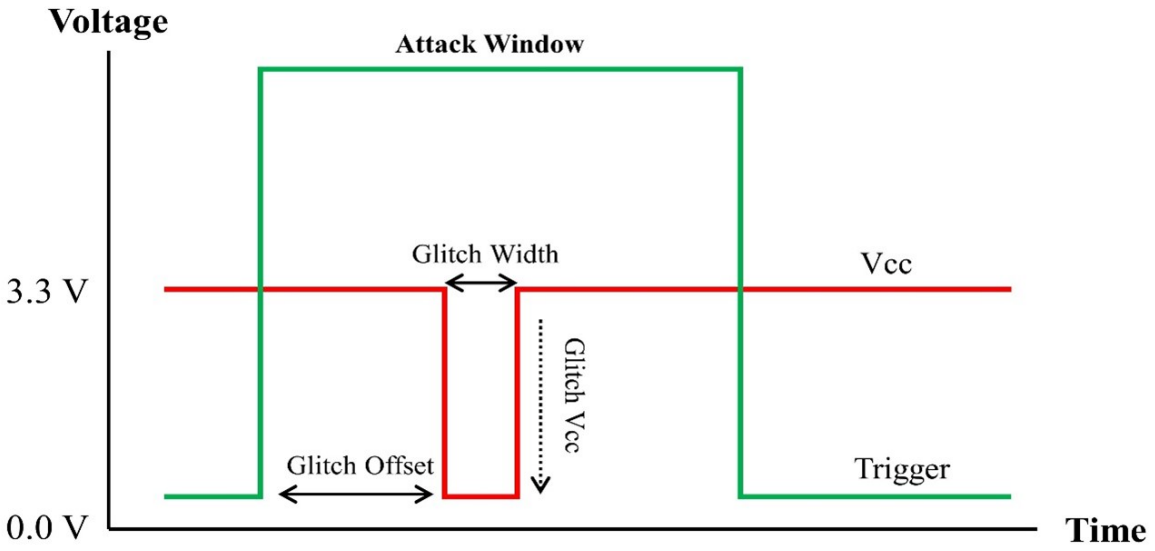


Figure 3. Voltage Glitch Parameters

3. The Fault Injection Model

In our assumption for model design, the attacker inserts a clock or voltage glitch into the softmax function at the output layer of the image classification DNN. The objective of the attack is to cause incorrect classifications for the given input images. Here, we assume that an attacker has the ability to inject glitches at desired points during execution of the softmax function. Algorithm 1 which is based on Equation (1) shows the operational flow of the softmax function implemented in a microcontroller.

In this algorithm, output array O is first initialized to 0. In the following For loop, an exponential operation is performed on input value y_k stored in array O at index k. Then, the values in O_k are summed and stored in the variable sum. In the second For loop, O_k is divided by sum. Consequently, output array O has the values converted to a probability distribution from the value of each neuron in the y array.

Algorithm 1 The softmax function

Require: $y(y \in \{y_0, \ldots, y_{J-1}\})$
Ensure: $O(O \in \{O_0, \ldots, O_{J-1}\})$
1: **for** $(k = 0; k < J; k++)$ **do**
2: $\quad O_k = 0$
3: **end for**
4: $sum = 0$
5: **for** $(k = 0; k < J; k++)$ **do** // Attack point of loop counter
6: $\quad O_k = exp(y_k)$
7: $\quad sum = sum + O_k$
8: **end for**
9: **for** $(k = 0; k < J; k++)$ **do**
10: $\quad O_k = O_k / sum$
11: **end for**

In this study, we assume that glitches are inserted into the first "For" loop: lines 5 to 7 of the softmax algorithm. The precise timing is at the point after the operation $k = 0$ in the loop has completed and before k becomes 1 during the first "For" loop. At this point, O_0 contains the correct values, but all subsequent values in the O array remain at zero. Additionally, only the value of O_0 is stored in the variable sum. Therefore, when the second "For" loop is executed, O_0 is 1, while the remaining classes have a value of 0.

 doi:10.20944/preprints202409.0373.v1

The assembly list compiled from lines 5 to 7 of Algorithm 1 is shown in Figure 4. The label L3 represents the section performing the operations within the loop. The label L2 corresponds to the portion verifying the conditional branch; if k is less than J then branching to L3 occurs. The injected glitch causes Line 50 of the assembly list to be skipped, and it prematurely terminates the first "For" loop.

```
L3:
16   ldr    r3, [fp, #-24]    Load Loop Counter(fp-24)
17   lsl    r3, r3, #3        3 Shift Left (scale for memory access)
18   ldr    r2, [fp, #-32]    Load argument (y)
19   add    r3, r2, r3        Calculate address of current element in y
20   ldmia  r3, {r2-r3}       Load multiple elements form arr (r2 to r3)
                .
                .
                .
L2:
47   ldr    r2, [fp, #-24]    Load Loop Counter (fp-24)
48   ldr    r3, [fp, #-36]    Load argument (J)
49   cmp    r2, r3            Compare R2 and R3
50   blt    .L3               Branch to L3 if condition is met
```

Figure 4. Assembly List of the Softmax Function

4. Glitch-Based Fault Attacks on a DNN

4.1. Experiment Setup

The target DNN models for fault injection attacks are MLP, AlexNet, and ResNet, which can classify digit images from the MNIST database [18]. This classification is the process of predicting input digit images ranging from 0 to 9 into 10 classes. The MNIST dataset used in our experiment consisted of 60,000 samples as the training dataset and 10,000 samples as the test dataset. Table 1 shows the MLP model structure for classifying the MNIST datasets used in our experiments. The input layer uses 784 neurons to receive images, and the output layer consists of 10 neurons. The output layer produces the final result through the softmax function.

Table 1. Structure of the MLP Model for Classification of the MNIST Datasets

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

Figure 5 shows classification of MNIST digits by a normal MLP model that was not subjected to a fault injection attack. This result is classifications from using the dataset of 10,000 MNIST images. As shown in the figure, we can confirm that classification was done with 97.74% accuracy.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 September 2024 doi:10.20944/preprints202409.0373.v1

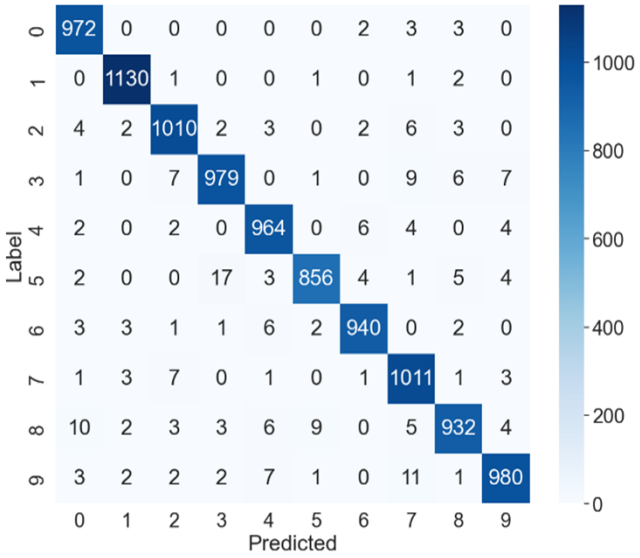


Figure 5. MNIST Dataset Classifications under Normal MLP Operation

To evaluate the performance of our fault injection attacks, practical experiments were conducted using the ChipWhisperer platform from NewAE, which includes CW-Lite (CW1173). Additionally, settings for the glitch-based fault injection attacks included the CW308 UFO Board equipped with an ARM Cortex-M4 STM32F303 microcontroller. The target DNN models were implemented in C and uploaded to the device under test using the ARM-GCC compiler. The setup for the experiment and the interface connection between devices is in Figure 6.

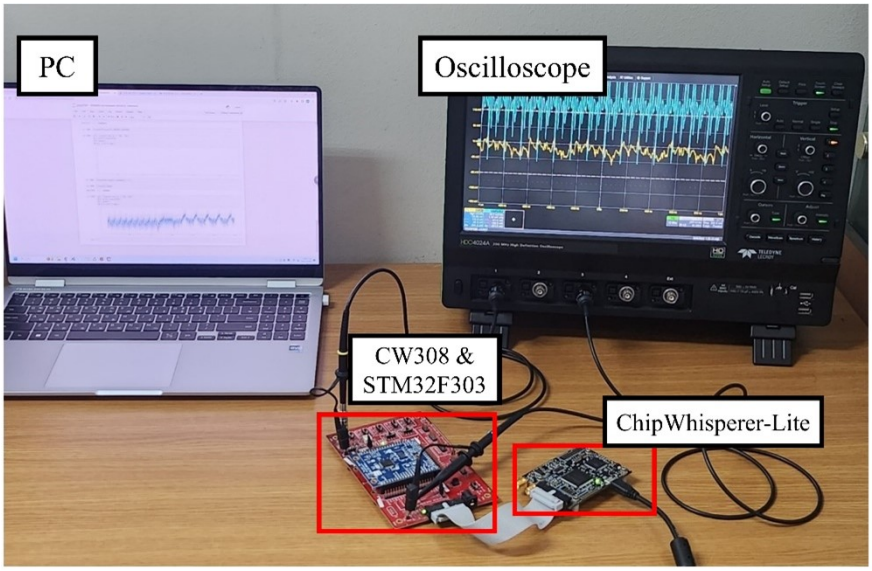


Figure 6. Setup and Interface between Experimental Devices

4.2. Glitch Attack Results

The middle column of Table 2 shows the prediction probability of classifying the input classes from the MNIST dataset. Since this is a normal situation, we can see that 1 was predicted as Class 1. On the other hand, on the right side of the table, you can see that image 1 was classified as 0 due to the glitch-based fault injection attack.

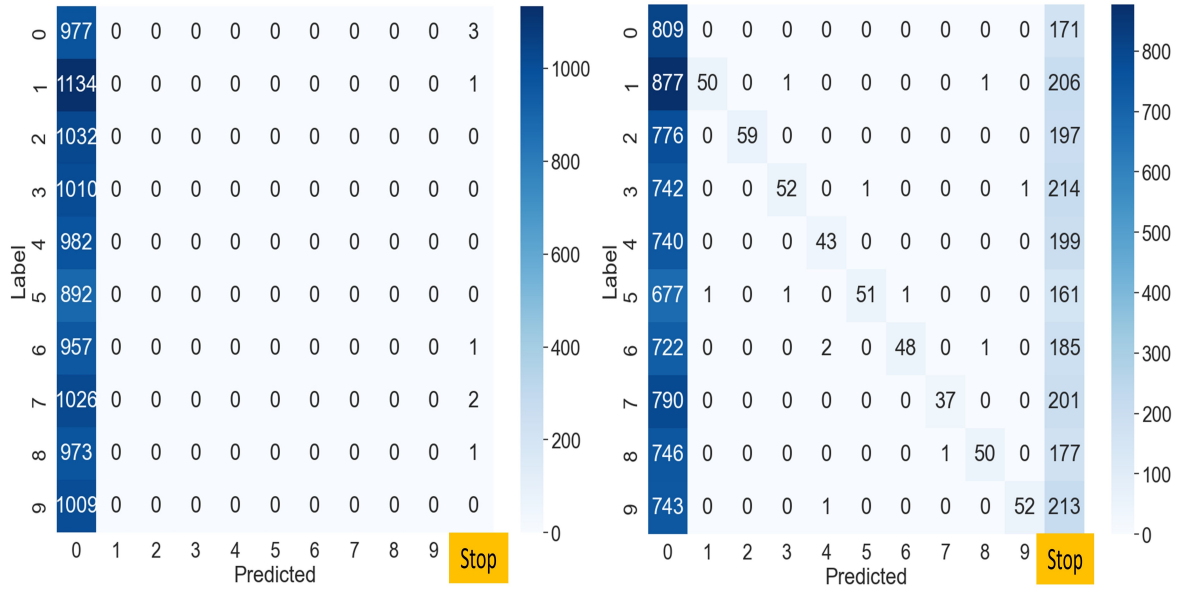
 doi:10.20944/preprints202409.0373.v1

Table 2. Comparison of Prediction Probabilities

Class	Normal Model	Attacked Model
0	1.0725232e-10	1.0
1	0.9999546	0
2	1.5233452e-07	0
3	5.8926429e-10	0
4	2.6625094e-07	0
5	1.3121079e-06	0
6	7.8415707e-07	0
7	3.3144510e-05	0
8	9.7056473e-06	0
9	1.4593203e-10	0

Figure 7a presents the results of the clock glitch fault injection attack on the MLP classification model using the MNIST test dataset of 10,000 samples. In this attack, most outputs are misclassified as 0 regardless of the input. We analyze that the reason is that the loop was skipped at the attack point in Algorithm 1. That is, the attacker can omit the branch statement at line 50 in Figure 4. This shows that the attacker can intentionally make the output zero through fault injection in the first iteration in Algorithm 1. While the accuracy of the normal MLP model was 97.74%, accuracy of the attacked model decreased to 9.77%. The values in the rightmost column of the confusion matrix indicate where the device terminated irregularly due to the clock glitch. The success rate of this fault attack was 99.9%, excluding data in Class 0.

In the voltage glitch fault attack, similar to the clock glitch, the loop process terminated at the same location. Figure 7b depicts the confusion matrix for the results obtained from the voltage glitch injection. This attack classified digit images with 12.51% accuracy. However, compared to the clock glitch attack on the MLP model, it caused a lot of response failures from the target device. This was not the desired outcome of the experiment, because it was not the result the attacker intended.



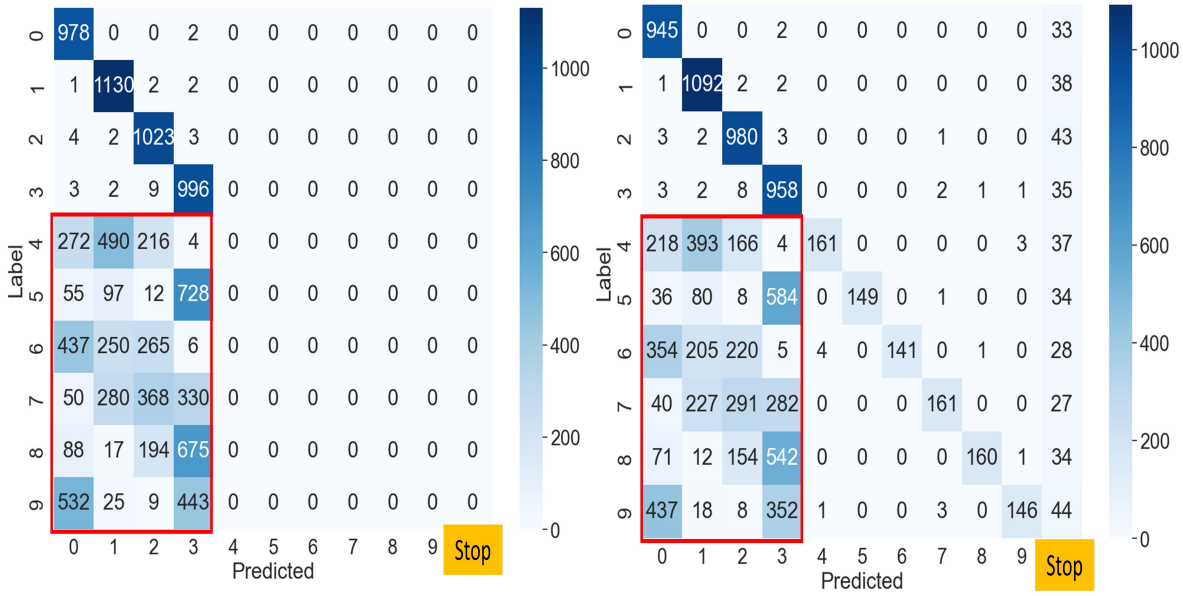
(a) Clock Glitch Attack **(b)** Voltage Glitch Attack

Figure 7. MLP Classifications under Glitch Fault Injection Attacks

We conducted experiments to determine if an attacker can intentionally classify MNIST images other than class 0. Figure 8a shows the misclassification results when injecting a fault into the fourth

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 September 2024 doi:10.20944/preprints202409.0373.v1

loop in the clock glitch attack. As shown in the figure, we found that the success rate of the attack is 100%, except for the results that predicted 0, 1, 2, and 3, which means that all inputs with a label of 4 or higher are misclassified. Figure 8b shows that when a voltage fault was injected during the fourth loop, the attack success rate could be 81.52%. Since the attack result depends on the order of the labels, the attacker cannot fully control the misclassification into a particular class. Nevertheless, the location of the flaw injection can limit the scope of misclassification in some cases.



(a) Clock Glitch Attack **(b)** Voltage Glitch Attack

Figure 8. Fault Attacks for Intentional Classification

Because the attack method in this approach targets only the softmax function in the output layer, it is theoretically applicable regardless of the DNN model or dataset. Table 3 presents results from attacking MLP and three CNN-based DNN models, popularly used in image recognition. Observe that not only the MLP model but also the other three models misclassified images with high probability.

Table 3. Fault Injection Attacks on MLP and other CNN-based DNN Models

	Normal Accuracy	**Glitch Model Accuracy**	
		Clock	**Voltage**
MLP	97.74%	9.77%	12.51%
AlexNet	98.99%	9.50%	9.35%
InceptionNet	99.06%	9.61%	10.64%
ResNet	98.98%	9.33%	10.39%

A comparison with other existing studies is shown in Table 4. The practical fault attack using laser injection resulted in random misclassifications of more than 50%[15]. Khoshavi et al. found that glitches targeting clock signals can reduce accuracy by 20% to 80%[19]. Liu et al. showed that an attacker can induce more than 98% misclassification rate in 8 out of 9 models using clock glitches[20]. A clock glitch-based fault injection into the softmax function of a DNN can lead to 98.6% misclassification with class 0 only[17]. We also performed misclassification attacks using clock glitches and voltage glitches against softmax functions and experimentally confirmed that we can misclassify them as 0 or other classes more than 99%.

 doi:10.20944/preprints202409.0373.v1

Table 4. Comparison of fault injection attacks

	Target	Method	Goal	Effect
Bereir et al.[15]	activation functions	Laser injection	random misclassification	$\geq$ 50% misclassification
Khoshavi et al.[19]	clock signal	clock glitch	accuracy degradation	20% $\sim$ 80% accuracy degradation
Liu et al. [20]	clock signal	clock glitch	misclassification	$\geq$ 98% misclassification
Fukuda et al.[17]	softmax function	clock glitch	misclassification to class 0	98.6% misclassification
Ours	softmax function	clock glitch voltage glitch	misclassification to class 0 or others	$\geq$ 99% misclassification

5. Countermeasure

5.1. Proposed Softmax Algorithm

The core of the fault injection attack on the softmax function lies in artificially omitting the loop process during execution. The main ideas of the countermeasure are based on two factors. One is that we should verify the number of loops during softmax execution. To address this, the softmax algorithm is designed to check whether the loop process has been executed completely or not. If you have not executed enough "For" loops for the number of output classes, you stop the process.

Even if a "For" loop executes normally, an injected fault can change the value of a variable in memory. Therefore, the other idea in the countermeasure is to check that the sum of output normalized probability values is always 1. The idea in the second method is to check the memory value to prove that the operation inside the loop was performed properly.

The countermeasure at the algorithmic level to resist fault attacks is outlined in Algorithm 2. In the proposed countermeasure, we verify that the loop process executed correctly through three conditional statements in the algorithm. The conditional statement in Line 9 checks if the first loop executed fully, and Line 16 verifies the second loop. Finally, Line 19 confirms that the sum of the output values equals 1. Here, we emphasize that Line 10 of Algorithm 1 and Line 13 of Algorithm 2 must be computed differently. The reason is that if Line 13 of Algorithm 2 is calculated as $O_k = O_k/sum$, the fault injection attack attempted at Line 6 cannot be detected.

 doi:10.20944/preprints202409.0373.v1

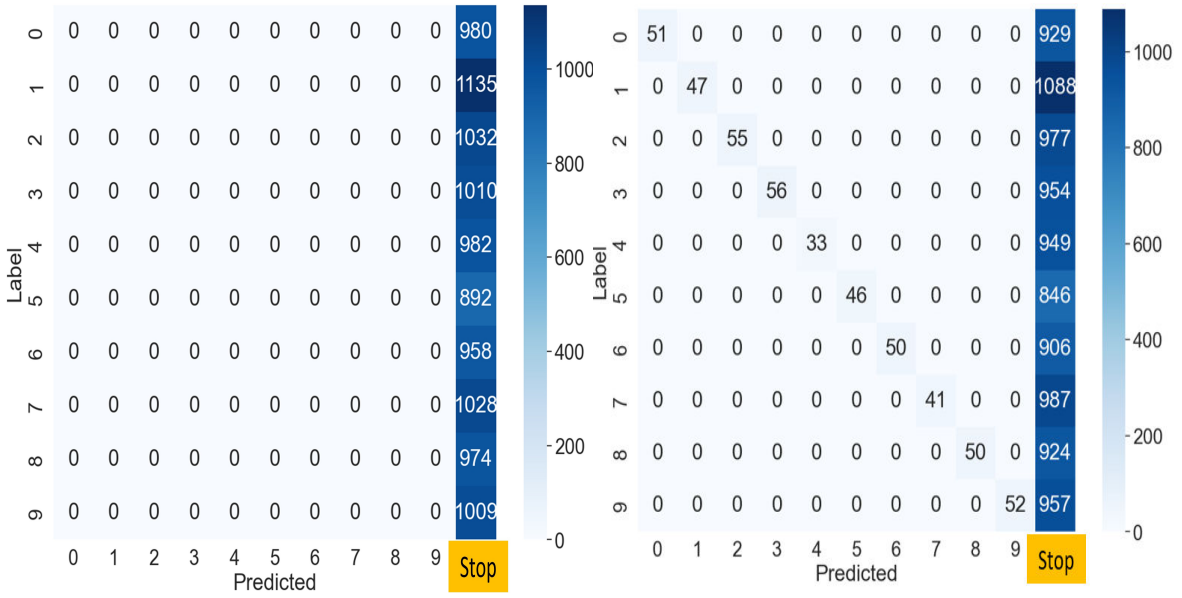
Algorithm 2 The proposed softmax function

Require: $y(y \in \{y_0, \ldots, y_{J-1}\})$
Ensure: $O(O \in \{O_0, \ldots, O_{J-1}\})$
for $(k = 0; k < J; k++)$ **do**
    $O_k = 0$
end for
$sum = 0, check = 0$
for $(k = 0; k < J; k++)$ **do**
    $O_k = exp(y_k)$
    $sum = sum + O_k$
end for
if $(k! = J)$ **then**
    exit(1);
end if
for $(k = 0; k < J; k++)$ **do**
    $O_k = exp(y_k)/sum$
    $check = check + O_k$
end for
if $(k! = J)$ **then**
    exit(1);
end if
if $(check! = 1)$ **then**
    exit(1);
end if

5.2. Fault Attacks on the Proposed Algorithm

To validate the proposed countermeasure algorithm, we conducted experiments on four DNN models using the same test dataset. In all cases where fault injection attacks were not launched on the softmax algorithm with countermeasures, image classification had the same accuracy as normal. Next, a glitch fault injection attack was launched against the algorithm adopting the countermeasure, and the result confirmed that faults were detected with 100% probability in the target device, and device operation stopped. Figure 9a shows the result from injecting a clock fault into the countermeasure softmax algorithm, and Figure 9b shows the result from injecting a voltage fault. From the experiments on the microcontrollers implemented with the proposed countermeasure, we confirmed that it can completely prevent misclassification on MNIST digit images.

 doi:10.20944/preprints202409.0373.v1



(a) Clock Glitch Attack **(b)** Voltage Glitch Attack

Figure 9. Fault Attacks on MLP with the Countermeasure

6. Conclusions

Deep learning, which has recently been attracting attention, is widely used for facial recognition and image detection in IoT-edge devices and smart homes. However, remote edge devices that run a DNN are exposed to threats such as fault injection and side-channel attacks. In particular, glitch fault injection attacks can easily cause misclassified images by utilizing simple tools.

This paper demonstrated that clock or voltage glitch injection attacks can effectively induce errors in image classification DNN models. The target for fault injection attacks is the softmax function at the output layer. To verify the applicability of fault injection attacks, we used datasets from the MNIST database for image classification and implemented DNN models on a 32-bit STM32F303 microcontroller. By terminating the first loop of the softmax function using glitches, misclassification of Class 0 was easily induced. Experimental results showed the classification accuracy from MLP decreased from 97.74% to 9.77% from a clock glitch attack and decreased to 12.51% from a voltage glitch. Furthermore, we proposed a countermeasure to glitch attacks on the softmax function, which is based on checking the loop counter and validating computational correctness. Consequently, we confirmed that our proposed softmax algorithm can sufficiently defeat glitch-based fault injection attacks.

Author Contributions: Conceptualization, methodology, software, S.H. and J.H. ; validation, Y.L., H. K. and J.H.; writing—original draft preparation, S.H. and H.K.; writing—review and editing, G.K. and J.H.; All authors have read and agreed to the published version of the manuscript.

Funding: 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)

Institutional Review Board Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

 doi:10.20944/preprints202409.0373.v1

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolution Neural Network
DNN	Deep Neural Network
EM	Electromagnetic
FPGA	Field-programmable gate array
IoT	Internet of Things
MLP	multi-layer perceptron

References

1. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* **2013**. doi:10.48550/arXiv.1312.6199.
2. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* **2014**. doi:https://doi.org/10.48550/arXiv.1412.6572.
3. Akhtar, N.; Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access* **2018**, *6*, 14410–14430. doi:https://doi.org/10.1109/ACCESS.2018.2807385.
4. Liu, Y.; Dachman-Soled, D.; Srivastava, A. Mitigating reverse engineering attacks on deep neural networks. 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2019, pp. 657–662. doi:https://doi.org/10.1109/ISVLSI.2019.00122.
5. Fang, Q.; Lin, L.; Zhang, H.; Wang, T.; Alioto, M. Voltage scaling-agnostic counteraction of side-channel neural net reverse engineering via machine learning compensation and multi-level shuffling. 2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits). IEEE, 2023, pp. 1–2. doi:https://doi.org/10.23919/VLSITechnologyandCir57934.2023.10185228.
6. Athanasiou, K.; Wahl, T.; Ding, A.A.; Fei, Y. Masking feedforward neural networks against power analysis attacks. *proceedings on privacy enhancing technologies* **2022**. doi:https://doi.org/10.2478/popets-2022-0025.
7. Agoyan, M.; Dutertre, J.M.; Naccache, D.; Robisson, B.; Tria, A. When clocks fail: On critical paths and clock faults. International conference on smart card research and advanced applications. Springer, 2010, pp. 182–193. doi:https://doi.org/10.1007/978-3-642-12510-2_13.
8. Barenghi, A.; Bertoni, G.; Parrinello, E.; Pelosi, G. Low voltage fault attacks on the RSA cryptosystem. 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). IEEE, 2009, pp. 23–31. doi:https://doi.rg/10.1109/FDTC.2009.30.
9. Breier, J.; Jap, D.; Chen, C.N. Laser profiling for the back-side fault attacks: with a practical laser skip instruction attack on AES. Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, 2015, pp. 99–103. doi:https://doi.org/10.1145/2732198.2732206.
10. Dehbaoui, A.; Dutertre, J.M.; Robisson, B.; Orsatelli, P.; Maurine, P.; Tria, A. Injection of transient faults using electromagnetic pulses Practical results on a cryptographic system. *ACR Cryptology ePrint Archive (2012)* **2012**.
11. Boneh, D.; DeMillo, R.A.; Lipton, R.J. On the importance of checking cryptographic protocols for faults. International conference on the theory and applications of cryptographic techniques. Springer, 1997, pp. 37–51. doi:https://doi.org/10.1007/3-540-69053-0_4.
12. Biham, E.; Shamir, A. Differential fault analysis of secret key cryptosystems. Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17. Springer, 1997, pp. 513–525. doi:https://doi.org/10.1007/bfb0052259.
13. Liu, Y.; Wei, L.; Luo, B.; Xu, Q. Fault injection attack on deep neural network. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2017, pp. 131–138. doi:https://doi.org/10.1109/iccad.2017.8203770.
14. Zhao, P.; Wang, S.; Gongye, C.; Wang, Y.; Fei, Y.; Lin, X. Fault sneaking attack: A stealthy framework for misleading deep neural networks. Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–6. doi:https://doi.org/10.1145/3316781.3317825.
15. Breier, J.; Hou, X.; Jap, D.; Ma, L.; Bhasin, S.; Liu, Y. Practical fault attack on deep neural networks. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 2204–2206. doi:https://doi.org/10.1145/3243734.3278519.

 doi:10.20944/preprints202409.0373.v1

16. Alam, M.M.; Tajik, S.; Ganji, F.; Tehranipoor, M.; Forte, D. RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions. 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). IEEE, 2019, pp. 48–55. doi:https://doi.org/10.1109/FDTC.2019.00015.
17. Fukuda, Y.; Yoshida, K.; Fujino, T. Fault injection attacks utilizing waveform pattern matching against neural networks processing on microcontroller. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **2022**, *105*, 300–310. doi:https://doi.org/10.1587/transfun.2021CIP0015.
18. LeCun, Y.; Cortes, C.; Burges, C. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* **2010**, *2*.
19. Rakin, A.S.; He, Z.; Fan, D. Bit-flip attack: Crushing neural network with progressive bit search. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1211–1220.
20. Liu, W.; Chang, C.H.; Zhang, F.; Lou, X. Imperceptible misclassification attack on deep learning accelerator by glitch injection. 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020, pp. 1–6.