on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2021CIP0015

Publicized:2021/09/22

This article has been accepted and published on J-STAGE in advance of copyediting. Content is final as presented.

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

Fault Injection Attacks Utilizing Waveform Pattern Matching against Neural Networks Processing on Microcontroller

Yuta FUKUDA[†], Kota YOSHIDA[†], *Student Members*, and Takeshi FUJINO^{††}, *Member*

SUMMARY Deep learning applications have often been processed in the cloud or on servers. Still, for applications that require privacy protection and real-time processing, the execution environment is moved to edge devices. Edge devices that implement a neural network (NN) are physically accessible to an attacker. Therefore, physical attacks are a risk. Fault attacks on these devices are capable of misleading classification results and can lead to serious accidents. Therefore, we focus on the softmax function and evaluate a fault attack using a clock glitch against NN implemented in an 8-bit microcontroller. The clock glitch is used for fault injection, and the injection timing is controlled by monitoring the power waveform. The specific waveform is enrolled in advance, and the glitch timing pulse is generated by the sum of absolute difference (SAD) matching algorithm. Misclassification can be achieved by appropriately injecting glitches triggered by pattern detection. We propose a countermeasure against fault injection attacks that utilizes the randomization of power waveforms. The SAD matching is disabled by random number initialization on the summation register of the softmax function.

key words: fault injection, clock glitch, neural network, pattern matching

1. Introduction

Deep learning systems are widely used in practical applications such as image recognition. There is a threat of inducing misclassification by attacking the deep neural networks (DNNs) implemented in these systems. An adversarial Example (AE) is an attack that adds noise to the input image to induce misclassification [1][2][3]. In DNNs that make important decisions (e.g., autonomous car and facial recognition systems), misclassification can have serious consequences. Therefore, countermeasures to AEs are being discussed [4][5][6].

Deep learning is often processed in a cloud or on servers. However, the execution environment for applications that require privacy protection and real-time performance is being moved to edge devices. In terms of security, these devices are physically accessible to attackers. Therefore, physical attacks need to be considered as a new attack method against DNNs.

One kind of physical attack is fault injection. This attack induces a malfunction by injecting faults into devices. Injected faults include clock glitches (illegal clock), power glitches (drop-in power supply voltage), electromagnetic radiation, and laser radiation. Various kinds of fault injection attacks against cryptographic circuits have been reported in

the past. In such a scenario, the attacker's target reveals a secret cryptographic key [7][8].

On the other hand, several attacks have been proposed that induce misclassification by applying fault attacks against DNNs. We summarize these attacks in Table 1. Rakin et al. proposed a method that causes DNN misclassification by searching for vulnerable bits in the DNN weight parameters and flipping the minimum number of bits in row-hammer attacks [9]. Zhao et al. proposed a method that misclassifies a specific input image into an optional target label by changing the DNN parameters [10]. Hong et al. investigated the effect of classification accuracy by changing a single parameter of DNN [11]. These studies assume fault injection attacks that modify DNN parameters by laser radiation or row-hammer attacks.

Bereir et al. proposed a fault attack against MCU on Arduino UNO by laser radiation to misclassify them [12]. This attack misleads the conditional-branch of the activation function. This attack uses a trigger from the target for the attack timing, which is not a realistic scenario. Liu et al. evaluated a fault attack that induces misclassification against the DPU [13][14]. Multiple clock glitches are injected while the DPU processes convolution layers. An adversary can not control the output class.

In this paper, we report an attack that induces misclassification by injecting a single clock glitch into a software implemented neural network (NN). We focus the loop processing on the matrix multiplication at the intermediate layers and the softmax function calculation at the output layer on NN calculation. The loop counts for summation can be reduced by skipping conditional-branch instructions using the clock glitch. The DNN misclassifies most inputs into class 0 under the attack. Also, the timing of the clock glitch injection affects the success rate of the attack. By attacking simple power analysis, it is easier to understand the internal state of the device rather than other applications [15]. Therefore, it is easy to identify the injection timing of the clock glitch. We can identify the injection timing of the clock glitch by applying pattern detection using the sum of absolute difference (SAD) algorithm on the power consumption waveform. This makes the attack scenario more realistic.

The main contributions of this work are summarized as follows:

- We proposed fault injection attacks with clock glitches against software implementation of NNs.
- We studied the fault injection points at the matrix mul-

[†]The author is with the Graduate School of Science and Technology, Ritsumeikan University, Kusatsu-shi, 525-8577, Japan.

^{††}The author is with Department of Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525-8577, Japan.

Table 1 Comparison table between related studies and this paper

	Bereir et al. [12]	Liu et al. [14]	Ours	Others[9][10][10]
Target operation	Activation function (ReLU, sigmoid, tanh)	Convolutional layer	Activation function (Softmax)	DNN parameter
Target Device	Arduino UNO (ATmega328P)	DPU [13]	ATXmega128	-
Fault	Laser	multiple clock glitch	single clock glitch	simulation
Attack timing	triggered by signal from target device	triggered by clock counter	triggered by power waveform matching	-

tiplication of the final hidden layer and the softmax function operation of the output layer. As a result, we found that the success rate of the fault attack is higher in the softmax function operation.

- We utilize the power-waveform matching in order to adjust the fault injection timing. It was found that the waveform obtained at adding zeros by an arbitrary number in the softmax function is a good matching point for high attack success rate.
- To use our method on a practical scenario, we determined the fault injection timing without modifying the Convolutional Neural Network(CNN) software installed on the target device. We used the power consumption waveform to determine the timing of injecting clock glitch. The attack success rate is 98.4% in the experiment.
- We also show the countermeasures that use random number initialization of the summation register, making waveform pattern matching difficult and identification of attack timing impossible.

This paper is organized as follows. Chapter 2 describes NNs, fault injection, and waveform pattern matching. Chapter 3 and 4 explain the threat model and our attack method. Chapter 5 discusses the experiment of fault injection against neural networks. Chapter 6 describes the countermeasures to these attacks. Chapter 7 concludes our work.

2. Preliminaries

2.1 Neural network and softmax function

Deep learning is one kind of machine learning. It trains NNs to perform classification and regression tasks. NNs are composed of neurons that are mathematical models of the nerves of human brains. The structure of a neuron is shown in Fig. 1. Equations (1) and (2) are used to calculate the neurons.

$$u = \sum_{i=0}^{l-1} w_i x_i + b \quad (1)$$

$$y = f(u) \quad (2)$$

$$f(u) = \begin{cases} u & (u > 0) \\ 0 & (u \leq 0) \end{cases}, \quad (3)$$

where $x(x \in x_0, \dots, x_{l-1})$ is the input vector, $w(w \in w_0, \dots, w_{l-1})$ is the weight vector, and b is the bias. The

Fig. 1 Structure of neuron **Fig. 2** Structure of MLP for classification task

weight vector represents the strength of each input's effect on that neuron. The activation function f controls the output of the neuron. In this paper, the ReLU function shown in Eq (3) is used as the activation function of hidden layers.

A multi-layer perceptron (MLP) is one of simple NN structure in which neurons are arranged in layers and connected in the forward direction as shown in Fig. 2. An MLP has one or more hidden layers in addition to the input and output layers. In the classification task, the softmax function is used as the activation function for the output layer. The softmax function is shown in Eq (4).

$$\text{softmax}(y_j) = \frac{\exp(y_j)}{\sum_{k=1}^J \exp(y_k)}, \quad (4)$$

where J is the number of classification. The softmax function normalizes each neuron's output in the output layer so that the sum of the output is 1 without changing their magnitude relationship. Therefore, in the classification task, we use the output value through the neuron's softmax function that corresponds to the probability of classification into that class. In this paper, the softmax function operation in the output layer is attacked to output incorrect probabilities.

Also, convolutional neural networks (CNNs) are the addition of convolution layers next to the input layer of MLP. The convolution layer extracts the features of the image. In this paper, CNN is used to perform the image classification task.

2.2 Fault injection by clock glitch injector

A fault attack induces a malfunction by injecting a fault into device. Injected faults include clock glitches (illegal clock), power glitches (drop-in power supply voltage), electromagnetic radiation, and laser radiation. In this paper, we used an attack method that induces malfunctions by injecting the clock glitch proposed by Fukunaga et al. [16]. Balasch et al. and Korak et al. analyzed the effects of clock glitch injection on processing [17][18].

Fig. 3 explains the basic principle of the clock glitch-based fault attack. In general, computation devices use synchronous circuits composed of combinational logics and registers. The result of operations by the combinational circuits is written to the registers at the clock's rising edge. In a fault attack that uses clock glitching, the glitch is injected before the combinational circuit's arithmetic operations are completed. A malfunction is induced by writing the result of an operation in progress to the register at an incorrect rising edge.

Fig. 3 Schematic diagram of synchronous circuit on microcontroller

Branch instructions are often the target in attacks on microcontrollers. The principle of such an attack is explained in Fig. 4. In normal operation, the address value $N + 1$ incremented by the value N of program counter output (PC_{out}) becomes the next input of the program counter (PC_{in}). The

Fig. 4 Program counter (PC) transition during branch instruction execution

address $N + 1$ is written to the PC register at the next rising edge of the clock. In branch instructions, first, the program counter is incremented to the address value $N + 1$, which becomes the input signal PC_{in} . Next, a combinational circuit is used to calculate whether the branching conditions are true or not. If the branching condition is true, PC_{in} is changed to the branch destination address I , which is written to the register at the next rising edge of the clock. In fault attacks as shown in Fig. 4 (b), the clock's rising edge is received before the computation of whether the branching conditions are true or not. Therefore, the incremented address value $N + 1$ is stored as a next PC value. As a result, the loop process can be exited in the middle.

In this paper, we used the Atmel AVR ATxmega128 as the target device. We injected a clock glitch into BRNE (Branch If Not Equal), which is assembly language instruction.

2.3 Waveform pattern matching

To achieve a reproducible fault attack using glitching, the attacker needs to know the internal operating state. In this work, we observed the power consumption waveform and performed pattern matching to accurately inject glitches. There are various algorithms for pattern matching [19], but we used the SAD algorithm in this paper. The SAD algorithm is explained in Fig. 5. First, the attacker enrolls the waveform pattern that he wants to match in advance. Next, the value is calculated as the total absolute value of the difference between the enrolled waveform pattern and the input waveform. Finally, if the value is less than or equal to the threshold, the waveform pattern and the input waveform are considered a match. This technique is also used to attack de-

Fig. 5 SAD algorithm for pattern matching

vices with misalignment countermeasures, such as random delay[20][21] and clock jitter in side-channel attacks.

The sampling rate is an important factor when performing waveform pattern matching. Trigger devices that perform pattern matching often have an upper limit on the number of points that can be kept. A detailed waveform can be handled at a higher sampling rate. However, the sample time becomes shorter. Conversely, the sample time can be long at a lower sampling rate. However, there is concern about fluctuations in the output trigger. The sampling rate needs to be chosen appropriately because it affects trigger generation accuracy and the fluctuation of the output trigger.

3. Threat Model

3.1 Adversary's capability

In this paper, we assume that the attacker has the following capabilities:

- The attacker can inject clock glitches into the target device. This assumption is applicable in the case that targeted clock is supplied from the external pin.
- The attacker can observe the power waveform of the target device.
- The attacker possesses a freely controllable device (profiling device) equivalent to the target that allows them to obtain the power waveform for matching.

3.2 Attack scenario with waveform pattern matching

In the practical fault attack, side-channel information is often used to adjust the clock glitch's injection timing. Van Woudenberg et al. performed power-signal-guided fault injection using a triggering mechanism based on real-time pattern matching [22]. In this paper, we used that method as well. In this profiling method, the analysis phase, in which the specific power waveform is enrolled, is required before the attack phase. The profiling phase and the attack phase are executed in that order as shown in block diagrams of each are shown in Fig. 6.

The pattern needed for pattern matching is selected in

Fig. 6 Attack sequence of power-signal-guided fault injection

the profiling phase. Using the specific same device as the target, the power waveform is observed by intentionally outputting a trigger near the attack timing. The observed waveforms are analyzed, and the part that shows the same power waveform regardless of the input value is selected as the pattern for matching.

In the attack phase, a fault is actually injected into the target device. The target device's power waveform is observed in real-time, and waveform matching is conducted using the patterns obtained in the analysis phase. The trigger generated based on the SAD algorithm is output to the glitch generator. The glitch generator injects a clock glitch into the target device when a trigger is activated.

4. Fault Injection on Neural Networks

4.1 Attack on matrix multiplication

The attacker skips the calculation of matrix multiplication of the n -th neuron in the output layer of the N -class classification problem. As a result, the probability of the n -th class can be reduced. By expanding Eq. (1), we get Eq. (5). We assume that the computation of the first to the i -th terms is implemented in a loop process. The bias b is added after matrix multiplication. If BRNE instruction for the loop process is skipped, only the first term and the bias can be used for operations, in Eq. (6). This is equal to the fact that the weights for the second and subsequent neurons in the previous layer are zero, as shown in Fig. 7.

$$u = w_0x_0 + w_1x_1 \cdots w_{l-1}x_{l-1} + b \quad (5)$$

$$u = w_0x_0 + b \quad (6)$$

The attacker switches Eq. (5) to Eq. (6) by injecting a clock glitch at the timing when the n -th neuron in the output layer is being processed. This can reduce the output probability, but it is difficult to intentionally misclassify to a particular class because the attacker cannot know the sign of the w and x values. Therefore, we change the attack point in the following sections.

4.2 Attack on softmax function

This paper focuses on the softmax function calculated in the

Fig. 7 Attack on matrix multiplication

Algorithm 1: Calculation of softmax function

Input: y ($y \in \{y_1, \dots, y_N\}$)
Output: O ($O \in \{O_1, \dots, O_J\}$)

- 1: $sum = 0$
- 2: for $k = 1$ to J do ← Attack Point
- 3: $O_k = \exp(y_k)$
- 4: $sum = sum + O_k$
- 5: end for
- 6: for $k = 1$ to J do
- 7: $O_k = O_k / sum$
- 8: end for

output layer as the timing for injecting clock glitches. The softmax function is the most commonly used in the output layer for multiclass classification of NNs. It uses Eq. (4) and outputs the probabilities for each class.

The softmax function is implemented in the microcontrollers using Algorithm 1. We assume that the array O is initialized with all zeros. If the attacker exits the loop process in lines 2–4 at the timing when $k = 1$, O_n ($n > 1$) becomes 0. As a result, the probabilities of the first and subsequent classes are all zero. For example, considering the case of $\exp(y_1) = 0.01$ and $\exp(y_2) = 0.09$, $\text{Softmax}(y_1) = 0.1$ and $\text{Softmax}(y_2) = 0.9$, and it is classified as the second class. However, when skipping the first operation in lines 2–4, $\text{Softmax}(y_1) = 1$ and $\text{Softmax}(y_2) = 0$, and it is classified as the first class. This attack can be used to prevent classification into the first and subsequent classes regardless of the value of the input.

We discuss the possibility of an attack on lines 6–8 in Algorithm 1. This is calculated by dividing the value of the exponential operation for each class by the variable sum. Then, a loop is processed to calculate the probability of each class. If a clock glitch is injected during this process, the division is not processed after the clock glitch is injected. For classes in which division is not processed, the exponential function is used as the output probability. We consider that

this attack cannot reproducibly induce the misclassification intended by the attacker, so the above method adding lines 2–4 is demonstrated in the following experiment.

5. Experimental Results

5.1 Experimental setup

ChipWhisperer-Lite (CW1173), a board developed by New AE Technology, was used as a control FPGA. The target device for the fault attack was an ATXmega128, which implemented part of the CNN processing. It ran the program with a clock frequency of 7.38 MHz. To generate the accurate glitch shape, the FPGA board DE0-CV developed by Terasic is used as the clock glitch generator. This generator is injected into the clock line of the target device as same as the experimental setup of [14][16]. To achieve waveform pattern matching, we used the icWaves, a trigger device developed by Riscure. This device can also be used to perform waveform analysis. We used the DSOX3104T oscilloscope developed by Keysight Technologies to observe the target device's power waveform, output trigger, and clock glitches. The block diagram and the photograph of the experimental setting are shown in Fig. 8 and Fig. 9.

Fig. 8 Block diagram of experimental equipment

Fig. 9 Photograph of experimental equipment

This attack's target was a CNN that performs MNIST handwritten character classification. The classification takes

Table 2 Structure of CNN for MNIST

Layer Type (Data Shape)
Input(28,28)
Conv2D + Relu (26,26,8)
Conv2D + Relu (24,24,4)
Flatten (2304)
Fully connected + Relu (32)
Fully connected + Softmax (10)

Table 3 The specification of target device

Target device	ATXmega128
Program memory size	128KB
Clock frequency	7.38MHz
Compiler	avr-gcc 5.4.0
Optimizer option	-Os(size optimization)

input images of handwritten digits from 0 to 9 and predicts the number. There are ten classes to be classified. The structure of the CNN is shown in Table 2. The network consists of two convolutional layers and two full-connecting layers. Due to the microcontroller's lack of memory space, the first three layers were calculated on another PC using Python. The calculated intermediate values were sent to the microcontroller. Then, the fourth layer of operations, including the softmax function (the attack target of this paper), was executed. The device, compiler, and optimization options for implementing the NN into the targeted device are shown in Table 3. If the targeted device has large program memory size, all calculation can be done in the device, however, the layers except the output layer were calculated on the external PC and the calculated intermediate values were sent to the targeted device, because the device has small program memory. The results of the fault injection attack are the same on both cases because the attack point is on the output layer. A source code of the output layer is written in C according to Algorithm 1. The C source and compiled assembly list are shown in Fig. 10 and Fig. 11. The fault clock is issued at the BRNE instruction, then the loop execution is skipped. We used 60,000 samples for the training dataset and 10,000 samples for the test dataset. The confusion matrix of the CNN for the test dataset is shown in Fig. 12. The accuracy of the model is 98.6%.

5.2 Clock glitch generation for skipping branch instruction

The shape of the clock glitch affects the success rate of the fault attack. The parameters of the clock glitch are width and offset, as shown in Fig. 13. To find the appropriate parameters, we measure success rate in skipping the BRNE instruction when each of these parameters is varied.

First, we measured the success rate of the attacks by varying the offset while the width was fixed at 3.6 ns. The success rate was calculated from the number of successful attacks per ten measurements at each measurement point. The offset was tested in 35 conditions from the range of 0 ns to 15 ns. The number of successful attacks per ten measurements is shown in Fig. 14. Fig. 14 shows that 100% of the attacks were successful in the offset time range of 9.1

```

void softmax(double* digit, int digit_len, double* out)
{
    double sum = 0.0;
    int i;
    for(i=0; i<digit_len; i++)
    {
        out[i] = exp(digit[i]);
        sum += out[i];
    }
    for(i=0; i<digit_len; i++)
    {
        out[i] /= sum;
    }
    return;
}

```

← Attack Point

Fig. 10 Source code for softmax function written in C language

```

438:  f8 01  movw  r30, r16
43a:  61 91  ld    r22, Z+
43c:  71 91  ld    r23, Z+

474:  8e 06  cp     r8,  r30
476:  9f 06  cpc   r9,  r31
478:  f9 f6  brne  .-66
47a:  20 e0  ldi   r18, 0x00
47c:  30 e0  ldi   r19, 0x00
47e:  a9 01  movw  r20, r18

```

Fig. 11 Assembly list of softmax **Fig. 12** Confusion matrix on MNIST classification in the normal operation

ns to 13 ns.

Next, we measured the success rate of the attacks by varying the width with the offset fixed at 11 ns. As in the offset experiment, the measurements were conducted ten

Fig. 13 Parameters of clock glitch

Fig. 14 Offset time dependence (glitch width fixed at 3.6 ns)

Fig. 15 Glitch width dependence (offset time fixed at 11 ns)

times at each measurement point, and the success rate was calculated. The width was tested in 19 conditions in the range of 0 ns to 6 ns. The number of successful attacks per ten measurements is shown in Fig. 15. Fig. 15 shows that 100% of the attacks were successful in the width time range of 2.3 ns to 5 ns

Based on these results, we experimented using a clock glitch with a glitch of 4.5 ns and an offset of 11.3 ns.

5.3 Profiling phase for enrollment of triggering waveforms

In the profiling phase, the pattern for timing identification was selected. The icWaves trigger device had an A/D converter and the memory for storing waveforms. The device was used to analyze the waveform from the profiling device as shown in Fig. 6 (a). The sampling rate of the device was set to 12.5 MHz. The waveform of the whole softmax function with a certain input sample is shown in Fig. 16 (a). The waveform of the class 0 exponential process (Algorithm 1, line 3) and the additive process (Algorithm 1, line 4) are shown in Fig. 16 (b). The waveform of the additive process is only shown in Fig. 16 (c). The same trend can be observed in the power waveform of the additive process. Therefore, the average of these patterns is used as the matching pattern. This waveform is shown in Fig. 16 (d). This pattern was enrolled in the icWaves.

Fig. 16 Waveform analysis of softmax function

(a) Overall view

(b) Enlarged view

Fig. 17 Trigger timing signal and clock glitch

5.4 Attack phase with clock glitch

The clock of the target microcontroller is supplied from the clock glitch generator, DE0-CV, and the triggering signal is generated from icWaves as shown in Fig. 6 (b). The trigger was output using the SAD algorithm. The SAD algorithm threshold was set to 600, and the trigger was set to output 3,860 ns later when the calculation result was lower than the threshold. Fig. 17 (a) shows the clock of the target microcontroller, the timing of the pattern matching, and the output trigger. Fig. 17 (b) shows an enlarged view of the target microcontroller's clock and the output trigger.

The classified results of MNIST at microcontroller under attack phase is shown in Fig. 18. While the accuracy was 98.6% in the normal operation as shown in Fig. 12, the accuracy was decreased to 11.2%. The 98.4% of the test images were classified into class 0, so we were calculated that the attack success rate is 98.4%.

5.5 Discussion

The experimental results show that the fault attack using the pattern matching of the power waveform was effective in the attack where all data is classified class 0. We considered an attack against another class (class 1–class 9). For example,

Fig. 18 Confusion matrix in fault attack **Fig. 19** Waveform of additive process in class 4

the addition process' waveform in class 4 is shown in Fig. 19. For the addition process in class 0, the processing time was matched for various input images, as shown in Fig. 16 (c). On the other hand, for the addition process in class 4, the processing time differed for each input image. Therefore, it is difficult to identify the specific pattern for matching.

In floating-point addition, the process is basically as shown in Fig. 20. First, the exponents of the two values are compared. The smaller number is shifted and adjusted to the exponential part of the larger number matches (1). Next, the mantissa part is added (2). Finally, the sum is normalized (3), and the mantissa part is rounded to the appropriate number of bits (4). If the result is not normalized, the sum repeats the process of normalizing (3)(4) until it is normalized.

In the class 0 additive process, the addend depends on the input image, but the augend is fixed at 0. When the addend is added to 0, the sum is the equivalent of the addend, so the sum does not need to be normalized. Therefore, the processing time is constant. In a class 4 additive process, the total up to class 3 is different depending on the input image, and the values of the augend depend on the input image. As a result, a normalization loop is more likely to be performed, and the processing time is different depending on the value of the number of augends. Therefore, we confirmed that fault attacks that use pattern matching are effective for class 0 targets, but it is difficult to attack other classes.

Fig. 20 Flowchart of floating-point addition

6. Countermeasures

6.1 Loop unrolling processing

A simple countermeasure is implementing a looped calculation without using “for” or “while” statements. Typical fault attacks using clock glitches target the conditional jump instructions during loop processing. Therefore, not implementing those instructions is a countermeasure to the attack. However, the number of code lines used for implementation increases if there are many classification classes.

6.2 Random number initialization of summation register

Random initialization is an effective countermeasure in the fault attacks that use power waveform pattern matching. In this section, we describe a countermeasure method that uses random number initialization on the summation register. The algorithm for the softmax function with random number initialization is shown in Algorithm 2. Section 4.4 discussed how the processing time for adding uncertain values to uncertain values is variable. Using this phenomenon, we use a random number m instead of 0 as the initial value of sum . This makes the attack difficult because the processing time varies in class 0.

6.3 Attack against countermeasured neural network

In this section, we describe our experiments on attacks against CNNs with countermeasures for random number initialization explained in Section 6.2. An analysis phase was executed to determine the power consumption trend. The

Algorithm 2: Calculation of *softmax* function

Input: y ($y \in \{y_1, \dots, y_N\}$)

Output: O ($O \in \{O_1, \dots, O_N\}$)

Random Initial: m

```

1:  $sum = m$ 
2: for  $k = 1$  to  $J$  do  $\leftarrow$  Attack Point
3:    $O_k = \exp(y_k)$ 
4:    $sum = sum + O_k$ 
5: end for
6:  $sum = sum - m$ 
7: for  $k = 1$  to  $J$  do
8:    $O_k = O_k / sum$ 
9: end for
  
```

(a) Power waveform of normal CNN

(b) Power waveform of countermeasured CNN with random number initialization

Fig. 21 Power waveform of CNN on the fault attack point

waveform of the additive process in class 0 is shown in Fig. 21. We confirmed that the processing time of the additive calculation varies by the random number initialization. We conducted a fault attack using the same setting and pattern as in the experiment against the unprotected CNN, but the attack’s success rate was 0%, confirming the effectiveness of the countermeasures.

7. Conclusion

Deep learning makes important decisions in systems such as autonomous cars and facial recognition. When they are executed on edge devices, physical attacks need to be considered. We evaluated a fault attack using pattern matching of power waveforms against NNs implemented in microcontrollers. We chose the Softmax function as our attack target and the ATXmega128 is used as the target microcontroller.

This attack consisted of an profiling phase in which the specific patterns required for pattern matching were selected and an attack phase in which the clock glitch is injected at the pattern matched timing.

We observed the power waveform of the target microcontroller during inference in the profiling phase. We confirmed that the waveform of the addition process executed at the softmax function showed the same power waveform trend regardless of the input image. This pattern was used to set the trigger timing.

In the attack phase, the pattern selected in the profiling phase was used to trigger the fault attack. By injecting a clock glitch after the class 0 addition process, 98.6% of the test data was classified into adversarial class.

As a countermeasure to make this attack difficult, we confirmed the effectiveness of shifting the attack's timing by initializing the variables used in the softmax function with random numbers. Using the fact that the floating-point execution time is variable, we confirmed that the attack timing's reproducibility worsens.

We consider two kinds future work; one is the attack feasibility against the practical microcontroller such as ARM, and the other is the possibility of fault attacks on processes other than the softmax function in NNs. In the former case, the basic investigation was tried. C source of Softmax function was compiled by ARM compiler, and found that the conditional branch (BNE) instruction is used as same as in AVR. Therefore, we think that the proposed fault attack is applicable in ARM microcontroller which is often used in edge devices.

From the experimental results in the paper, we conclude that the countermeasures against fault attacks need to be considered when deep learning applications are executed on the edge devices.

Acknowledgments

We would like to thank Takaya Kubota and Mitsuru Shiozaki for useful discussions. This work was supported by JST-Mirai Program Grant Number JPMJMI19B6, Japan.

References

- [1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," International Conference on Learning Representations, 2014.
- [2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," International Conference on Learning Representations, 2015.
- [3] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.
- [4] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," arXiv preprint arXiv:1705.07204, 2017.
- [5] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo, "No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation," IEEE Transactions on Information Forensics and Security, vol.12, no.11, pp.2640–2653, 2017.
- [6] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," arXiv preprint arXiv:1412.5068, 2014.
- [7] D. Boneh, R.A. DeMillo, and R.J. Lipton, "On the importance of checking cryptographic protocols for faults," Advances in Cryptology — EUROCRYPT '97, ed. W. Fumy, Berlin, Heidelberg, pp.37–51, Springer Berlin Heidelberg, 1997.
- [8] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," Annual international cryptology conference, pp.513–525, Springer, 1997.
- [9] A.S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp.1211–1220, Oct 2019.
- [10] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: a stealthy framework for misleading deep neural networks," 2019 56th ACM/IEEE Design Automation Conference (DAC), pp.1–6, 2019.
- [11] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," Proceedings of the 28th USENIX Conference on Security Symposium, SEC'19, USA, p.497–514, USENIX Association, 2019.
- [12] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp.2204–2206, 2018.
- [13] Xilinx, "DPU for Convolutional Neural Network v1.2.,"
- [14] W. Liu, C.H. Chang, F. Zhang, and X. Lou, "Imperceptible misclassification attack on deep learning accelerator by glitch injection," 2020 57th ACM/IEEE Design Automation Conference (DAC), pp.1–6, 2020.
- [15] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Csi neural network: Using side-channels to recover your artificial neural network information." Cryptology ePrint Archive, Report 2018/477, 2018. <https://eprint.iacr.org/2018/477>.
- [16] T. Fukunaga and J. Takahashi, "Practical fault attack on a cryptographic lsi with iso/iec 18033-3 block ciphers," 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp.84–92, 2009.
- [17] J. Balasch, B. Gierlichs, and I. Verbauwhede, "An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus," 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp.105–114, 2011.
- [18] T. Korak and M. Hoefer, "On the effects of clock and power supply tampering on two microcontroller platforms," 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp.8–17, 2014.
- [19] A. Beckers, J. Balasch, B. Gierlichs, and I. Verbauwhede, "Design and implementation of a waveform-matching based triggering system," Constructive Side-Channel Analysis and Secure Design, ed. F.X. Standaert and E. Oswald, Cham, pp.184–198, Springer International Publishing, 2016.
- [20] C. Clavier, J.S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," Cryptographic Hardware and Embedded Systems — CHES 2000, ed. Ç.K. Koç and C. Paar, Berlin, Heidelberg, pp.252–263, Springer Berlin Heidelberg, 2000.
- [21] J.S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," Cryptographic Hardware and Embedded Systems - CHES 2009, ed. C. Clavier and K. Gaj, Berlin, Heidelberg, pp.156–170, Springer Berlin Heidelberg, 2009.
- [22] J.G.J. van Woudenberg, M.F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp.91–99, 2011.

Yuta Fukuda Yuta Fukuda received his B.E. in electronic engineering from Ritsumeikan University in 2020. He is currently a master's student at the Graduate School of Science and Technology, Ritsumeikan University. His research interests include machine learning and hardware security. He is a member of IEICE.

Kota Yoshida Kota Yoshida received his B.E. and M.E. in electronic engineering from Ritsumeikan University in 2017 and 2019. He is currently a doctoral student at the Graduate School of Science and Technology, Ritsumeikan University. His research interests include machine learning and hardware security. He is a member of IEICE, IEEE.

Takeshi Fujino Takeshi Fujino was born in Osaka, Japan, on March 17, 1962. He received his B.E. and M.E., and Ph.D. in electronic engineering from Kyoto University, Kyoto, Japan, in 1984, 1986, and 1994. He joined the LSI Research and Development center, Mitsubishi Electric Corp. in 1986. Since then, he had been engaged in the development of micro-fabrication processes, such as electron beam lithography, and embedded DRAM circuit design. He has been a professor at Ritsumeikan University since

2003. His research interests include hardware security such as side-channel attacks and physically unclonable functions. He is a member of IEICE, IPSJ, JSAP, IEEE.