IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2021, No. 3, pp. 149–175. DOI:10.46586/tches.v2021.i3.149-175

Timing Black-Box Attacks: Crafting Adversarial Examples through Timing Leaks against DNNs on Embedded Devices

Tsunato Nakai[1,2], Daisuke Suzuki[1] and Takeshi Fujino[2]

[1] Mitsubishi Electric Corporation, Kamakura, Kanagawa, Japan,
{Nakai.Tsunato,Suzuki.Daisuke}@{dy,bx}.MitsubishiElectric.co.jp
[2] Ritsumeikan University, Kusatsu, Shiga, Japan,
fujino@se.ritsumei.ac.jp

Abstract. Deep neural networks (DNNs) have been applied to various industries. In particular, DNNs on embedded devices have attracted considerable interest because they allow real-time and distributed processing on site. However, adversarial examples (AEs), which add small perturbations to the input data of DNNs to cause misclassification, are serious threats to DNNs. In this paper, a novel black-box attack is proposed to craft AEs based only on processing time, i.e., the side-channel leaks from DNNs on embedded devices. Unlike several existing black-box attacks that utilize output probability, the proposed attack exploits the relationship between the number of activated nodes and processing time without using training data, model architecture, parameters, substitute models, or output probability. The perturbations for AEs are determined by the differential processing time based on the input data of the DNNs in the proposed attack. The experimental results show that the AEs of the proposed attack effectively cause an increase in the number of activated nodes and the misclassification of one of the incorrect labels against the DNNs on a microcontroller unit. Moreover, these results indicate that the attack can evade gradient-masking and confidence reduction countermeasures, which conceal the output probability, to prevent the crafting of AEs against several black-box attacks. Finally, the countermeasures against the attack are implemented and evaluated to clarify that the implementation of an activation function with data-dependent timing leaks is the cause of the proposed attack.

Keywords: Adversarial examples · Black-box attack · Timing side-channel · Embedded devices

1 Introduction

Artificial intelligence using deep neural networks (DNNs) has achieved phenomenal success in image recognition, speech recognition, and natural language processing. Accordingly, the number of applications, systems, and devices with DNNs increased in various industries. For example, several automated driving systems use DNNs for object recognition, and face authentication systems with DNNs have been applied to biometric authentication for unlocking laptops or smartphones. Security systems with malware behavior detection or network anomaly detection systems also incorporate DNNs.

The development of DNNs is impacted by big data, hardware acceleration, and development platforms. Moreover, considerable amounts of training data have been collected from internet services or sensor data deployed in the real world. The advances in hardware technologies, such as cloud computing servers, central processing units, and graphics

Received: 2020-10-15 Accepted: 2020-12-15 Published: 2021-07-09

processing units, enable the high-speed computation of large data for DNNs. Specifically, embedded devices have increasingly attracted attention because they allow real-time and distributed processing on site. To incorporate DNNs into microcontrollers (MCUs), several frameworks, such as uTensor [uTe17] and TensorFlowLite [Ten17], may be employed.

The widespread application of DNNs, however, has been exploited by attackers; in view of this, several attacks against DNNs have been investigated [NIS19]. Adversarial examples (AEs) [SZS+14], also known as adversarial example / evasion attacks, which are among the attacks against DNNs, add small perturbations to the input data of DNNs to prompt misclassification, causing a severe real-world impact. For instance, a DNN model for image recognition erroneously identified ”stop” as ”speed limit” by only attaching a small sticker to a road sign as an AE [EEF+18]. For speech recognition, it is actually possible to craft malicious speech data as AEs against a DNN speech recognition model on smart speakers [YS19]. These are also regarded as AEs against malware detection [GPM+17]. The other attacks against DNNs have also been reported such as poisoning attacks which train poisonous models that cause misclassification by introducing malicious data into the training data [JOB+18], model inversion attacks which steal training data from the DNN models [FJR15], and model extraction attacks which steal internal information from DNNs [TZJ+16].

Existing attacks for AEs are categorized as white-box [GSS15] and black-box attacks [CW16], as using the perfect knowledge of a trained model and limited knowledge of input-output data, respectively. Several black-box attacks employ output probability to craft AEs. Accordingly, one of the countermeasures include gradient-masking [PMSW16] and confidence reduction [CAD+18], which conceals the output probability, i.e., the output is merely a predicted label without any probability.

The security of DNNs on embedded devices has been discussed recently [IGGK19, SW19]. Several of the attacks against DNNs of embedded devices (e.g., against cryptographic devices [Koc96, KJJ99]) have been severe because the attackers easily reverse-engineered them or measured the side-channel leaks, such as processing time and power consumption. Side-channel attacks have been reported since 2018; however, practically all existing works on DNN attacks focus on model extraction [HZS18, BBJP19].

In this paper, a novel attack for crafting AEs using the differential processing time according to the input data of the DNNs on embedded devices is proposed. The proposed attack is a black-box attack against embedded devices, focusing on the relationship between the number of activated nodes and the processing time of DNNs on embedded devices. This creates AEs that can reduce the output probability of a correct label. Compared with existing black-box attacks, the proposed attack only utilizes the processing time without the necessity of output probability; consequently, the attack renders the gradient-masking and confidence reduction countermeasures ineffective.

The proposed attack targets DNNs on embedded devices such as real-time object detection, anomaly detection, and natural language processing system. It is assumed that the design information of the DNNs is protected because of intellectual property. That is also the countermeasure against white-box attacks. Moreover, the output of DNNs is a predicted label by the countermeasure against black-box attacks. In this scenario, the proposed attack measures processing time, crafts AEs using data-dependent timing leaks without any probability, and feeds AEs as malicious data to embedded devices via input files. In this study, the focus is on how to craft digital AEs with small perturbations. Considering more practical threat, it is also necessary to assume physical AEs which are robust AEs against various environmental noise in the physical world. The physical AEs are the adjusted AEs to use signals from cameras and other sensors in the physical world. In this paper, the physical AEs are out of scope, however, a significant fraction of digital image AEs can cause misclassification through cameras in the physical world [KGB16]. The proposed attack has the potential ability to be applied to craft the physical AEs by

combining signal processing methods considering environmental noise [YS19]. Moreover, a lot of existing works uses digital AEs to evaluate attacks.

The proposed attack is evaluated with basic DNN models on two MCUs. The experimental results demonstrate that the proposed attack can craft effective AEs to cause the misclassification of one of the incorrect labels by increasing the number of activated nodes using data-dependent timing leaks. Furthermore, the cause of the proposed attack is analyzed by using constant time countermeasure for preventing data-dependent timing leaks.

The contributions of this work can be summarized as follows.

- A novel black-box attack to craft AEs is proposed using differential processing time according to the input data of DNNs on embedded devices. This is the first adversarial example attack using side-channel leaks.
- Two relationships are identified: between the processing time and the number of activated nodes and between the number of activated nodes and AEs.
- It is demonstrated that the proposed attack depends on the processing time of activation functions, and different attack methods for two types of activation functions are presented.
- The cause of the proposed attack is clarified by implementing a countermeasure using activation functions with constant time to prevent data-dependent timing leaks.

This paper is organized as follows. In Section 2, the background attacks against DNNs and AEs are explained. In particular, the related works on side-channel attacks against DNNs and black-box attacks of AEs are presented in Section 3. Then, in Section 4, a timing black-box attack is proposed. The evaluation and experimental results are shown in Section 5, and the cause of the proposed attack is analyzed in Section 6. Finally, the conclusions are summarized in Section 7.

2 Background

In this section, the relevant background knowledge of crafting AEs against DNNs is presented. Our proposed attack is a novel black-box attack without exploiting output probability. The taxonomy of attacks against DNNs is shown in Appendix A.

Adversarial example attacks are mainly classified into two types according to the attacker's knowledge: white-box and black-box attacks. White-box attacks craft AEs using trained model information, such as the architectures and parameters of DNNs. The attackers can extract trained model information and use this to effectively craft AEs. Black-box attacks craft AEs using the input-output data of DNNs. These are more severe than white-box attacks because they can still execute adversarial example attacks despite the absence of internal trained model information. Nevertheless, it is more difficult for black-box attacks to craft effective AEs because of the limited knowledge of the targeted DNN models.

The first black-box attack was proposed by Papernot et al. in 2016 [PMG16]. The attack can craft AEs based on AE transferability. This transferability is a property that several AEs generate to mislead a specific model, f, to in turn mislead other models trained with similar data although their architectures differ. The attackers craft substitute models using the input-output data of the targeted DNN models. Thereafter, they attack the target via white-box attacks using the crafted substitute models. The AE transferability depends on the type of DNN. The trained models with a deeply hidden layer possess the robustness of AE transferability [ASCS18]. Moreover, several attacks only use the output labels of DNNs without the output probability to craft the substitute model [PMSW16].

However, the attacks require a large input-output dataset; consequently, attacks that employ transferability are more time-consuming than other attacks.

In contrast, several black-box attacks without the necessity of training a substitute model have been proposed since 2016 [NK17, GGY+19]. Narodytska et al. crafted AEs by constructing a numerical approximation of the network gradient using the output probability of each label on DNNs [NK17]. Guo et al. formulated AEs according to the assumption of a moderate continuous output probability [GGY+19]. The attack, i.e., a simple black-box adversarial attack (SBA), is one of the most powerful black-box attacks. The foregoing attacks utilize both output probability and labels; therefore, gradient-masking [PMSW16] and confidence reduction [CAD+18], which conceal the output probability, are among the effective countermeasures.

In this paper, a timing black-box attack is proposed as a novel black-box attack against embedded devices. The proposed attack features AE crafting by employing data-dependent timing leaks as side-channel information in a black-box attack scenario. The proposed attack is only focused on the input data-dependent processing time without utilizing the output probability; hence, gradient-masking and confidence reduction countermeasures are ineffective. Furthermore, the proposed attack can craft AEs within a short duration because it does not require the generation of a substitute model. Note that the proposed attack is non-targeted and causes misclassification to one of the incorrect labels.

3 Related Works

This section introduces the related works pertaining to side-channel attacks against DNNs and black-box attacks for AEs.

3.1 Side-Channel Attacks against DNNs

Since 2018, side-channel attacks against DNNs have been reported, and virtually all existing works only focus on model extraction attacks.

Hua et al. showed the leak in the architecture of a convolution neural network (CNN) by focusing on the memory access pattern (memory address/time) in the CNN accelerator [HZS18]. They also revealed the model parameters of the CNN by concentrating on the memory access pattern in the zero pruning accelerator. There are the other papers that focus on the memory access pattern [YFT18, HDK+18, AM18]; these are called cache-based side-channel attacks.

Batina et al. demonstrated the reverse engineering of DNNs on an MCU based on the side channel information, such as timing and electromagnetic emanations (EM) [BBJP19]. Yoshida et al. extracted the architecture and parameters of a DNN model using the EM on the field-programmable gate array accelerator [YKSF19]. Duddu et al. profiled the processing time of several DNN models to reveal the type of DNN architectures [DSRB18]. Dubey et al. proposed countermeasures against differential power analysis attacks to protect the DNN model parameters [DCA19].

This study focuses on the differential processing time of DNNs on embedded devices as side-channel information. This means that in the proposed attack, data-dependent timing side-channel leaks are employed to craft AEs.

3.2 Black-Box Attacks for AEs

The fast gradient sign method (FGSM) [GSS15] is among the most powerful and fastest white-box attacks. The FGSM adds perturbations to each input data component in the vector of increasing DNN loss. Given DNNs with parameter θ and loss $J(\theta; x; y)$, the

FGSM yields the following AE x_{adv}:

$$x_{adv} = x + \epsilon * \mathrm{sign}(\nabla_x J(\theta; x; y)), \tag{1}$$

where x is the input data, y is the output data, and ϵ is a reasonably small perturbation value. The attackers should know the internal trained model information, such as θ. In the case of DNNs on embedded devices, one of the countermeasures is encrypting the internal trained model to conceal it [IBCK18]. For embedded devices, reverse engineering countermeasures are effective against white-box attacks.

The SBA, which extends the FGSM, focuses on the changes in output probability instead of the function loss [GGY+19]. This attack is one of the most powerful and fastest black-box attacks; it introduces small perturbations in the vector of decreasing output probability of a correct label (non-targeted attacks) or increasing output probability of an incorrect label (targeted attacks). Therefore, it is possible for either $x + \varepsilon$ or x - ε (where x is a part of the input data, and ϵ is a perturbation) to change the output probability of each input data component. The attackers only use the output probability according to each input data part. In the case of DNNs on embedded devices, one of the countermeasures is the confidence reduction countermeasure, which limits the output probability of each label; basically, the output is only a label.

In this paper, a black-box attack without an output probability (caused by the confidence reduction countermeasure) is proposed. This means that the proposed attack focuses on embedded devices and non-targeted attacks.

4 Timing Black-Box Attacks

This section shows the concept of the proposed attack. The threat model, proposed timing black-box attacks, and attack algorithm are also elaborated.

4.1 Threat Model

The proposed attack scenario and the attacker capability are explained as follows. The main goal of attackers is to craft AEs by measuring the processing time of DNNs in a black-box scenario without an output probability because of the confidence reduction countermeasure.

4.1.1 Attack Scenario

The attack scenario mainly focuses on embedded devices. The DNNs on these devices have attracted interest because they allow distributed and real-time processing on site. In view of this, however, attackers can easily measure the processing time if they gain access to these devices. In addition, an ENISA report mapping the assets and threats of artificial intelligence notes physical attacks to hardware [ENI20]. The threat model of the physical attacks fits our attack.

In this study, the focus is set on simple MCUs as embedded devices for proof of concept. Several frameworks, such as uTensor [uTe17] and TensorFlowLite [Ten17], incorporate DNNs into simple MCUs for low power and low cost. The security of DNNs on simple MCUs such as Arduino [Ard05] has been discussed recently [SW19, BBJP19]. These devices have single-core and deterministic timing behavior, however, the deployed DNN sizes are small due to limited resources. Embedded devices with neural network accelerator for large DNNs and high speed, such as Jetson nano GPU [Jet19] and Coral Edge TPU [Cor19], are discussed in Appendix B.

In this scenario, the attacker measures the processing time, crafts AEs by using data-dependent timing leaks, and feeds AEs to embedded devices via input files. Two security

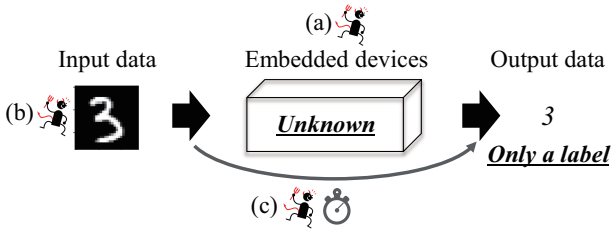


Figure 1: Attack scenario of timing black-box attacks against DNNs on embedded devices with naive countermeasures

functions are assumed as naive countermeasures for DNNs on embedded devices. The first function is model encryption; it is a countermeasure against reverse engineering and white-box attacks. The other function is confidence reduction, which is a countermeasure against several black-box attacks, such as the SBA. The attack crafts AEs by using data-dependent timing leaks without using model information and output probability. The advantage of the attack scenario is that our attack can craft AEs using processing time with fewer queries than existing black-box attack scenarios if the model is protected (encrypted) and the output is only label. The attackers can get the processing time by modifying the execution code to measure the time from input to output of DNN operation. In case of embedded devices, it is possible to measure the time by acquiring the power consumption or electromagnetic radiation.

Our attack can also be applied to a scenario such as a profiling attack. The attackers craft AEs on a target device, and then input the AEs to other devices for misclassification. These devices have the same protected model. For example, one of the realistic attack scenarios is an attack to Optical Character Recognition (OCR) with DNN. A pre-computed noise is applied to an input image as an AE. The input image file looks fine, but it can fool the character recognition. Some papers have proposed AE attacks to OCR [SS18, CX20].

4.1.2 Attacker's Capability

Figure 1 shows the attack scenario against embedded devices from the viewpoint of attackers. First, the attackers do not know the targeted internal trained model information, such as the (a) architectures and parameters. They also do not have information on the output probability because of the confidence reduction countermeasures. The output is only a predicted label. Second, they can (b) feed their input data into the targeted device. Third, they are capable of (c) measuring the processing time from the input to the output of the targeted device. Therefore, the attack achieves success to craft AEs simply by measuring the processing time through feeding the input data. The attackers can measure the processing time from input to output of DNN operation.

4.2 Proposed Attacks

This subsection presents the proposed attack concept, focusing on the number of activated nodes and data-dependent timing leaks of activation functions. The basic idea of the attack is that increasing the number of activated nodes causes misclassification because increasing propagated values affects the output of DNNs.

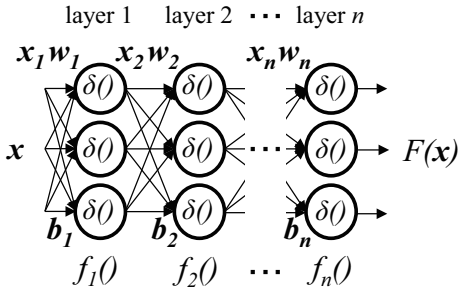


Figure 2: Simple DNN model

4.2.1 AEs and Activated Nodes

Goodfellow et al. explained the existence of AEs in linear models [GSS15]. Given the original input data, x, and a perturbation, η, the AE, $\tilde{x}$, is as follows.

$$\tilde{x} = x + \eta. \tag{2}$$

Consider the dot product between a weight vector, w, and $\tilde{x}$:

$$w^T \tilde{x} = w^T x + w^T \eta. \tag{3}$$

According to Eq. (3), the perturbations cause the activation, $w^T \tilde{x}$, to increase by $w^T \eta$ in the linear model. If $\eta = \varepsilon$ sign(w), then the activation will increase by εmd, where ε is a small perturbation, w has d dimensions, and the average magnitude of a weight vector element is m. Therefore, though perturbations only cause small changes to the input data, they greatly affect the output data by devising the perturbation vectors in higher dimensions such as DNNs.

Crafting the AEs is considered here by focusing on the number of activated nodes in DNNs. Figure 2 shows a simple DNN example to explain the concept. The DNN model, $F()$, consists of n successive layers of perceptrons from the input data, $\boldsymbol{x}$ to the output data:

$$F(\boldsymbol{x}) = f_n(...f_2(f_1(\boldsymbol{x_1}))), \tag{4}$$

$$\boldsymbol{x_{i+1}} = f_i(\boldsymbol{x_i}), \tag{5}$$

$$x_{i,j} = \delta(\sum_k (w_{i,j,k}\, x_{i-1,k}) + b_{i,j}), \tag{6}$$

where the calculation of f_i of the ith layer includes the input data, $\boldsymbol{x_{i+1}}$, whose vector component, $x_{i,j}$, from the kth to the jth perceptron is calculated by the activation function, δ, weight vectors, $w_{i,j,k}$, and bias vectors, $b_{i,j}$. According to Eqs. (5) and (6), the DNN model nests the activation functions, δ. In the DNNs, the activation function of each node defines the output of the nodes given the input data. The node output can either be activated and non-activated. Accordingly, the DNN output data are affected depending on whether the nodes are activated or non-activated. Crafting the AEs is considered by selecting the perturbation vectors for increasing the number of activated nodes, i.e., nodes that are ready to propagate values. This is because the AEs aim to severely affect the output data via the small perturbation, thus explaining the linear model (Eq. (3)).

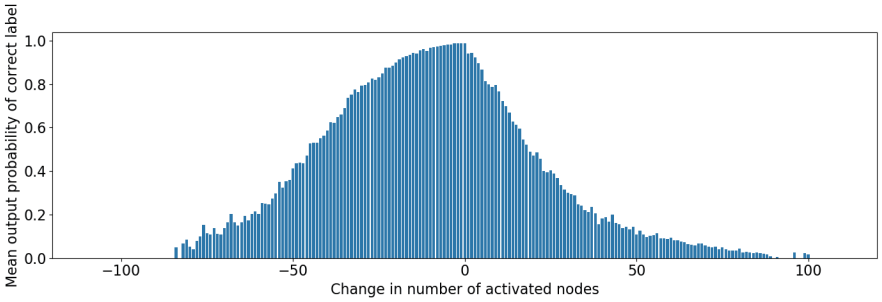


Figure 3: Simulation of relationship between the change in the number of all layers' activated nodes and the mean output probability of the correct label in the MLP model with ReLU.

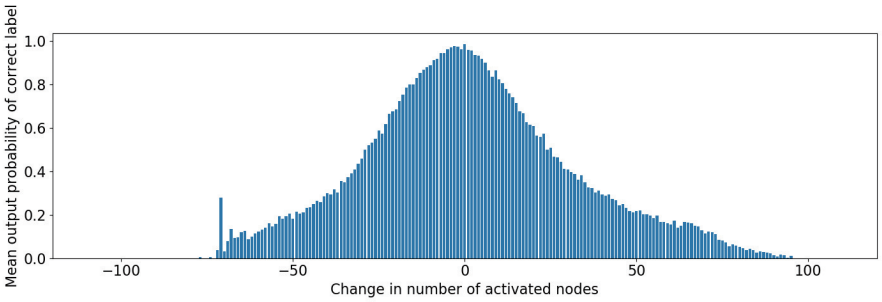


Figure 4: Simulation of relationship between the change in the number of all layers' activated nodes and the mean output probability of the correct label in the MLP model with Sigmoid.

4.2.2 Simulation of Activated Nodes

The connection between the number of activated nodes and the output probability of correct labels is simulated. The simulation uses a basic multilayer perceptron (MLP) model on a Linux OS computer (Ubuntu 18.04) with an Intel Core i-5 CPU. Table 1 summarizes the architecture of the MLP model, which is trained with 60,000 elements of training data from the Modified National Institute of Standards and Technology (MNIST) database of handwritten digits [MNI98]. The simulation uses two activation functions, rectified linear unit (ReLU) and Sigmoid functions.

The simulation focuses on activation nodes in the first layer (128 nodes), which is sensitive to the input data, to manipulate the number of activated nodes. The randomly selected activation nodes in the first layer [0-128] are rewritten with an activated value (average of all activated node's values) or a non-activated value (0), respectively. When the activation nodes are rewritten, the simulation observes the increase or decrease in the number of activated nodes in all layers and the output probability of the correct label.

Figure 3 and Figure 4 show the relationship between the change in the number of all layers' activated nodes and the mean output probability of the correct label in the MLP model with ReLU and Sigmoid functions, respectively. Furthermore, Figure 5 and Figure 6 show the number of successful attacks (misclassification) on 1,000 samples per the change of all layers' activated nodes. The output probability is the mean value on 1,000 samples per the change of all layers' activated nodes.

Table 1: MLP model architecture for MNIST dataset

Layer Name	Layer Type	Model
Input layer	Input	784
1st layer	Fully Connected	128
2nd layer	Fully Connected	64
3rd layer	Fully Connected	32
Output layer	Softmax	10

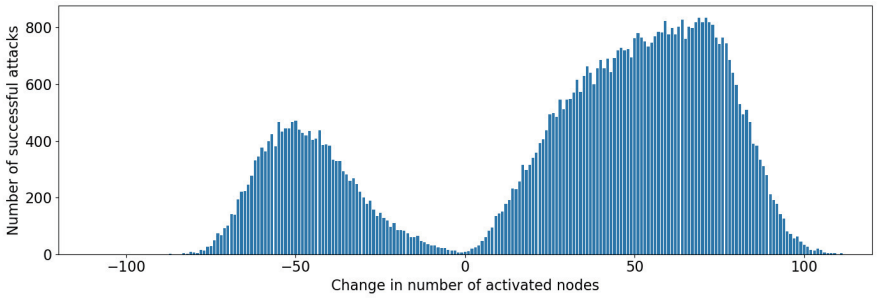


Figure 5: Simulation of relationship between the change in the number of all layers' activated nodes and the number of successful attacks (misclassification) in the MLP model with ReLU.

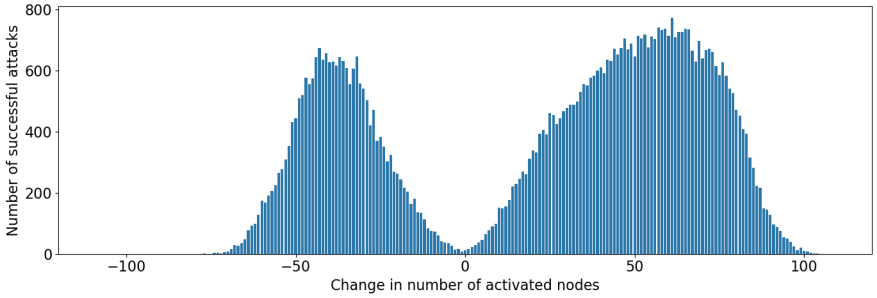


Figure 6: Simulation of relationship between the change in the number of all layers' activated nodes and the number of successful attacks (misclassification) in the MLP model with Sigmoid.

The simulation results show that the output probability decreases as the increase or decrease in the number of activated nodes in both ReLU and Sigmoid functions. The decrease in the output probability causes misclassification and increases the number of successful attacks. This phenomenon is especially noticeable when the number of activated nodes increases. Increasing the number of activated nodes results in more successful attacks in simulations and therefore this is a strategy an attacker would use to develop an AE. Appendix C shows similar simulation results for CNN because CNN mainly consists of convolution layers including activation functions.

4.2.3 Activation Nodes and Data-dependent Timing Leaks

It is hypothesized that the change in the number of activated nodes may be observed from the processing time of activation functions. Batina et al. reported that several types of activation functions have different processing times depending on the input data [BBJP19].

For example, ReLU function, which is one of the most commonly used activation functions, is as follows:

$$\delta(z) = \begin{cases} z & \text{if } z > 0 \text{ (the processing time : } t_\alpha), \\ 0 & \text{if } z \leq 0 \text{ (the processing time : } t_\beta), \end{cases} \tag{7}$$

where z represents the input data; t_α andt_β are the processing times of the activated and non-activated cases, respectively. It outputs all zeros for negative input data (non-activated) and directly yields the output data for positive input data (activated). The data-dependent timing leaks of a ReLU function mean that the processing time is longer if the input data are positive because of the algorithm and its implementation. In other words, the processing time tends to be longer if the node is activated by the ReLU function, that is, $t_\alpha > t_\beta$; hence, the ReLU function is involved.

In the case of ARM Cortex-M4 on the evaluation board, a conditional branch is completed in a single cycle if the branch is available; otherwise, it is completed in three cycles. The source code of the original activation function written in C++ is as follows.

```
if (in[i] > 0 ) {              //δ(z) : z > 0 in the equation (7)
    out[i] = in[i];
} else {                       //δ(z) : z ≤ 0 in the equation (7)
    out[i] = 0;
}
```

Therefore, depending on the input data, the activation function indicates two cycles as differential processing times.

In the case of a sigmoid function, which is one of the activation functions, the processing time tends to be shorter if the node is activated [BBJP19]. Therefore, the data-dependent timing leaks also indicate whether the nodes in the DNNs have been activated or not by sigmoid functions, that is, $t_\alpha < t_\beta$.

4.2.4 Crafting AEs using Data-dependent Timing Leaks

The idea of the proposed attack is to craft AEs by selecting the perturbation vectors to increase the number of activated nodes using the data-dependent timing leaks of activation functions. According to the DNN model in Eq. (5), the number of all nodes, P_{f_i} in $f_i(\boldsymbol{x_i})$ is

$$P_{f_i} = \alpha_{f_i} + \beta_{f_i}, \tag{8}$$

where α_{f_i} and β_{f_i} are the number of activated and non-activated nodes in $f_i(\boldsymbol{x_i})$, respectively. Moreover, the processing times, T_{f_i} of $f_i(\boldsymbol{x_i})$ and $T(\boldsymbol{x})$ of $F(\boldsymbol{x})$, are

$$T_{f_i} = \alpha_{f_i} t_\alpha + \beta_{f_i} t_\beta, \tag{9}$$

$$T(\boldsymbol{x}) = T_{f_1} + T_{f_2} + ... + T_{f_n}, \tag{10}$$

where t_α and t_β are the processing times of the activated and non-activated nodes on the activation function, δ, respectively. Note that the nodes are not processed in parallel. According to the Eqs. (8)-(10), the number of activated nodes, α_{f_i}, is significantly correlated to the processing time, $T(\boldsymbol{x})$. Hence, the proposed attack measures the processing time and crafts AEs by deciding the perturbation vectors for increasing the number of activated nodes using the processing time.

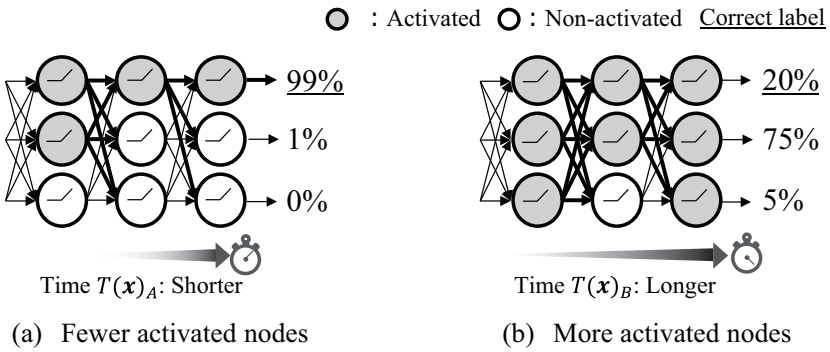


Figure 7: The attack concept against DNNs with ReLU functions is that these functions have longer processing times when the nodes are activated by the data-dependent timing leaks depending on the input data. Increasing the number of activated nodes affects the DNN output data because of the increasing number of propagated values.

Figure 7 shows an example of the attack concept in the case of a DNN with ReLU functions. The proposed attack adds a small perturbation to the input data for increasing the number of activated nodes. If the number of activated nodes increases, then the output probability of the correct label is more affected than the output probability of fewer activated nodes shown in Figure 7 (a). Moreover, a ReLU function uses a longer processing time if the output node is activated. The DNN with more activated nodes is more time-consuming in the case shown in Figure 7 (b). Therefore, according to Eqs. (9) and (10) and $t_\alpha > t_\beta$ in the ReLU function, the proposed attack focuses on $T(\boldsymbol{x})_A < T(\boldsymbol{x})_B$, as shown in Figure 7.

4.3 Algorithm of Proposed Attacks

Figure 8 shows the strategy of the proposed attack for AEs. First, (a) add a small perturbation, ε, to a part of the input data, x, of embedded devices with DNNs. Next, (b) measure the processing time when the input data are fed to the embedded devices, and decide the perturbation vectors ($x + \varepsilon$ or $x - \varepsilon$) for each part of the input data (x) according to the changes in the measured processing time. Finally, (c) craft AEs to add the selected perturbations to the original input data.

Even if the confidence reduction countermeasures hide the output probability, the proposed attack can craft AEs in a black-box attack scenario because these perturbations are determined without the output probability. The proposed attack depends on the type of activation function. To ensure accuracy of DNNs, the common activation operations are limited to ReLU and sigmoid functions; therefore, the attackers can estimate the type of activation function used. Basically, if an activation function has data-dependent timing leaks (depending on whether or not the activation nodes are enabled), the attacks can test both the abovementioned perturbation vectors, which may require a long or short processing time.

Algorithm 1 shows the proposed attack based on the assumption that the activation functions, such as ReLU functions, require more time if the output node is activated. The presented attack targets embedded devices with DNN $F(x)$, which contains the input data, x (with size n and label y). The parameter for crafting an AE is the perturbation bound, ε, which is set in advance.

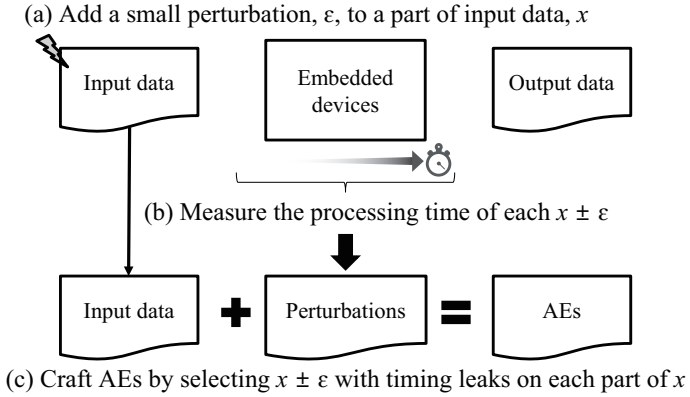


Figure 8: Strategy of timing black-box attacks for AEs

Algorithm 1 Algorithm of timing black-box attacks in ReLU function case

Require: Input x, Input size n, Index i, DNN $F(y\text{:Label}|x)$
Output: AE x_{adv}
Parameters: Perturbation bound ε

1: $time()$ //Get the current time
2: **for** $i = 0$ to n **do**
3: $x' = x$
4: $x'_i = x'_i + \varepsilon$ //Add the perturbation $(+\ \varepsilon)$
5: $t1 = time()$
6: $F(y|x')$ //Prediction
7: $t2 = time()$
8: $elapsed\_time1 = t2$ - $t1$ //Measure the processing time
9: $x' = x$
10: $x'_i = x'_i$ - ε //Add the perturbation $(-\ \varepsilon)$
11: $t1 = time()$
12: $F(y|x')$ //Prediction
13: $t2 = time()$
14: $elapsed\_time2 = t2$ - $t1$ //Measure the processing time
15: **if** $elapsed\_time1 > elapsed\_time2$ **then**
16: $ptb_i = 1$ //Decide the perturbation vector $(+)$
17: **else**
18: ptb_i = -1 //Decide the perturbation vector $(-)$
19: **end if**
20: **end for**
21: $x_{adv} = x + \varepsilon\ ptb$ //Craft the AE
22: **return** x_{adv}

In Algorithm 1, the perturbations, ε, is first added to the input data, x', copied from the original input data, x; a pixel of the input data, x'_i, is $x_i + \varepsilon$ or $x_i - \varepsilon$. Moreover, the process time of each perturbation vector ($+\varepsilon$ and $-\varepsilon$) is measured, i.e., $elapsed\_time1$ and $elapsed\_time2$. Thereafter, these measured processing times, $elapsed\_time1$ and $elapsed\_time2$, are compared. The activated nodes consume more time; hence, let ptb_i be the perturbation vector ($+\varepsilon$ or $-\varepsilon$) of the longer processing time for the pixel. The above process is executed for each pixel of the input data, x_i, with the image size, n. Finally, an AE x_{adv} is crafted by adding the perturbation vectors, $\varepsilon\ ptb$, to the input data, x.

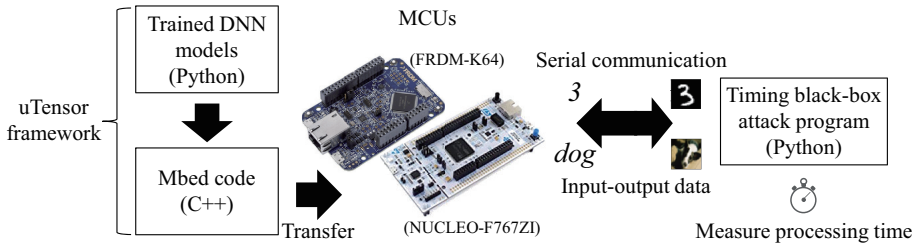


Figure 9: Experimental environment

Table 2: CNN model architecture for CIFAR-10 dataset

Layer Name	Layer Type	Model
Input layer	Input	32x32x3
1C layer	Convolution	2x2x3x16
2C layer	Convolution	3x3x16x32
1P layer	Pooling	1x2x2x1
3C layer	Convolution	3x3x32x32
4C layer	Convolution	3x3x32x32
2P layer	Pooling	1x2x2x1
5C layer	Convolution	1x1x32x64
6C layer	Convolution	1x1x64x128
1F layer	Fully Connected	128
2F layer	Fully Connected	64
Output layer	Softmax	10

5 Experiments and Evaluation

This section presents the experimental results and evaluation of the proposed attack on two MCUs as embedded devices. First, the relationship between the number of activated nodes and AEs crafted by data-dependent timing leaks from a simple DNN on an MCU is investigated. The attack crafts the AEs to increase the number of activated nodes using the time leaks. Second, the proposed attack is evaluated for comparison with random noise under the same condition without the use of output probability. The result shows that the proposed attack can craft effective AEs with small perturbations. Moreover, two common activation functions, which are ReLU and sigmoid, are compared because the proposed attack is based on the time leaks of activation functions. Finally, the presented attack is evaluated on other MCUs for CNN.

5.1 Setup

The experimental setup is based on the threat model presented in Section 3.1; Figure 9 shows the experimental environment. The attack is evaluated on two MCUs as embedded devices.

A basic MLP and a CNN model are considered. These models are commonly used in modern applications and include the basic layers of DNNs. Accordingly, the evaluation using the two models is considered a worst-case scenario because the models are regarded as a minimum configuration of other modern DNNs. Table 1 summarizes the architecture of the MLP model, which is trained with 60,000 elements of training data from the MNIST database. The accuracy of the MLP model is 97.47%. Table 2 lists the architecture of the CNN model, which is trained with 50,000 elements of training data from the CIFAR-10

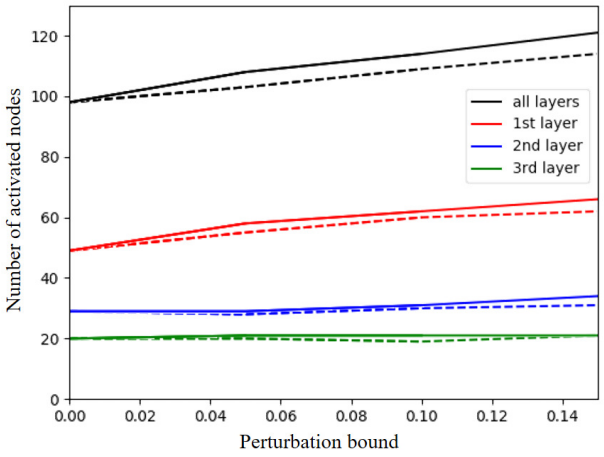


Figure 10: Relationship between perturbation bound and number of activated nodes crafted by AEs of proposed attack using data-dependent timing leaks (solid line) and random noise (dashed line) in one AE. Proposed attack can increase number of activated nodes with small perturbations decided by data-dependent timing leaks compared with random noise. The attack causes misclassification at 0.15 perturbation.

dataset (Canadian Institute for Advanced Research) [CIF09]. The accuracy of the CNN model is 81.96%.

The DNN development platform is TensorFlow 2.0 in Python 3.7. In this work, the uTensor framework, which is one of the first open-source frameworks to place DNNs on MCUs [uTe17], is used. The evaluation boards are FRDM-K64F [FRD14], which has an ARM Cortex-M4 MCU, and NUCLEO-F767ZI [NUC16], which has an ARM Cortex-M7 MCU. The MLP model is deployed in FRDM-K64F, and the CNN model is deployed in NUCLEO-F767ZI based on memory size.

In this experiment, the proposed attack program inputs test data into the evaluation board over serial communication and receives an output label from the board. The processing time between the input and output of the MCU is measured using time.perf_counter() function in the program. Note that the output is only a label and has no probability.

5.2 Activated Nodes and Data-dependent Timing Leaks

The change in activated nodes by the AEs, which are crafted by the proposed attack using data-dependent timing leaks, is also investigated. According to Section 4.2, the attack focuses on the relationship between activated nodes, AEs, and data-dependent timing leaks. In this experiment, the attack crafts AEs against the MLP model on the MCU (FRDM-K64F); the activation functions are ReLU functions. Therefore, the proposed attack focusing on the processing time tends to be longer if the node is activated, that is, $t_\alpha > t_\beta$, in the ReLU functions. The MLP model is a white-box model to explain the relationship between the perturbation bound and the number of activated nodes. If the output of each activation function is not zero (activated), then the number of activated nodes in the MLP model is counted. The perturbation bound is added to the input data at increments of 0.05 until it causes misclassification.

Figure 10 shows the relationship between the perturbation bound governed by the data-dependent timing leaks and the number of activated nodes in one AE; the comparison with

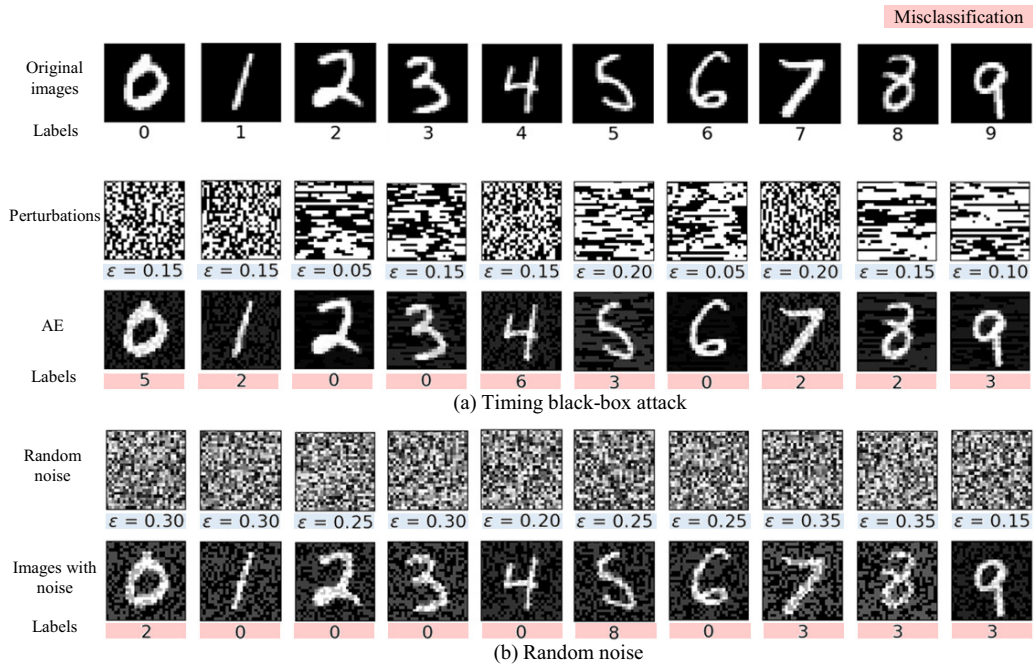


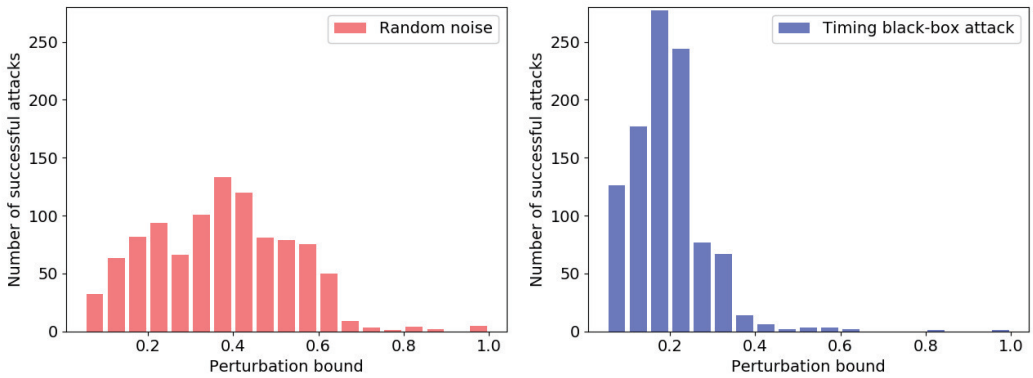
Figure 11: AEs crafted by proposed attack and random noise against MLP model on MCU. Original images are shown in first row, and attack results are presented in second (perturbations) and third (AEs) rows. Random noise results are shown in fourth (random noise) and fifth (images with noise) rows. Perturbation bound, ε, and misclassification labels are shown in the results.

random noise is also presented. Our attack causes misclassification with 0.15 perturbation. The number of activated nodes on each layer is counted; thereafter, their total number is calculated by finding the sum of all activated nodes from all layers. According to Figure 10, it is evident that the attack increases the number of activated nodes (solid line) using the data-dependent timing leaks; this number is greater than that activated by random noise (dashed line). Although the random noise also slightly increases the number of activated nodes, it cannot effectively increase this number with a small perturbation. Therefore, it is confirmed that the perturbations decided by the data-dependent timing leaks increase the number of activated nodes with small perturbations more effectively.

5.3 Evaluation of AEs

The proposed attack against the MLP model on the MCU is evaluated by comparing it with random noise; the activation operations are ReLU functions. The proposed attack is a black-box attack without using output probability. Accordingly, the proposed attack is compared with random noise under the same condition.

Figure 11 shows the experimental results of the attack and the random noise against the MLP model on the MCU. The perturbation bound, ε, is increased in 0.05 steps until it causes misclassification. The random noise is within [-1,1] in 0.05 steps. The figure shows that our AE samples in all types of MNIST labels, i.e., 0-9, are crafted by the proposed attack using small perturbations in the third row. The average perturbation bound of our AEs from the second row is 0.135. In contrast, the random noise, ε, is larger than the



(a) Random noise (b) Timing black-box attack

Figure 12: Histogram of successful attacks on 1,000 samples of MNIST dataset. Proposed attack is compared with random noise. Each perturbation bound is plotted when misclassification is caused by proposed attack and random noise.

perturbations (ε) of our attack for misclassification in the fifth row. The average random noise is 0.27 from the fourth row. Therefore, the proposed attack can craft more effective AEs with small perturbations (ε), which are approximately half as large as the random noise.

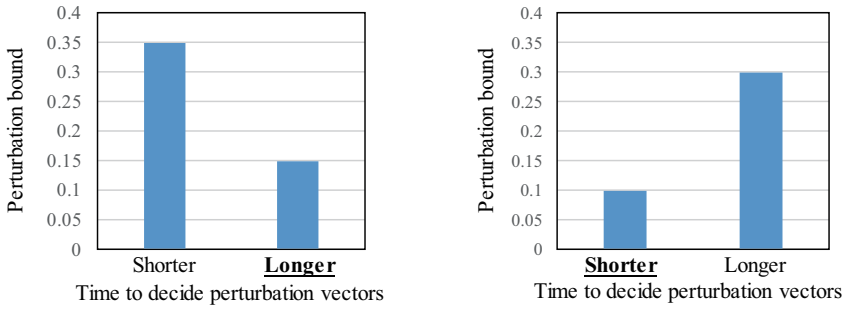
Figure 12 shows the experimental results in 1,000 samples of the MNIST dataset as well as the histogram of successful attacks on the perturbation bound until it causes misclassification on each sample. The histogram indicates that practically all the attacks with small perturbations from 0.05 to 0.25 cause misclassification. By contrast, that of the random noise is from 0.25 to 0.45 for misclassification. The average perturbation bound of the attack is 0.16; the peak signal-to-noise ratio (PSNR), which is commonly used to measure the quality of image compression, is 25.18 dB. Typical values for the PSNR in high-quality images are around 30 dB, where higher are better. By contrast, that of the random noise is 0.35; the PSNR is 11.73 dB. It is clear that the proposed attack tends to craft AEs with small perturbations compared with the random noise in 1,000 samples of the MNIST dataset.

The number of queries for the attack depends on input pixel, perturbation vectors (+ and -), and increments of perturbation bound. For example, it takes 4,704 queries for AEs of 0.15 perturbation at increments of 0.05 in MNIST. However, attacks not using output probability need at least 23,000-300,000 queries in MNIST [CLC+19].

5.4 Activation Functions and Data-dependent Timing Leaks

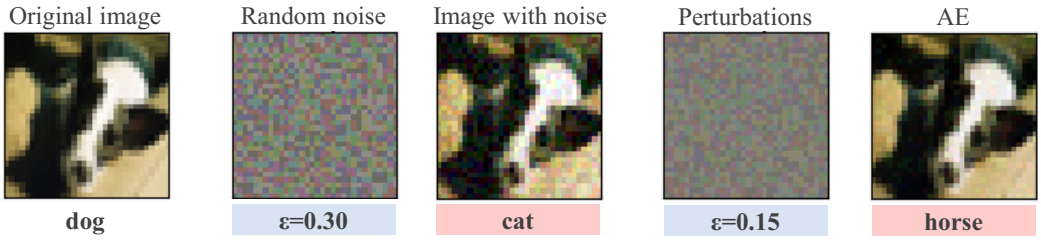
Two types of common activation functions, which are ReLU and sigmoid, are implemented and evaluated against the MLP model on the MCU. According to Section 4.2, the proposed attack is based on the data-dependent timing leaks of activation functions. If the node is activated, the data-dependent timing leaks of a ReLU function require a long processing time, whereas those of the sigmoid function tend to be shorter. These data-dependent timing leak models are based on the report of Batina et al. that activation functions have different processing times depending on the input data [BBJP19].

Figure 13 shows the comparison of results between the ReLU and sigmoid functions on the proposed attack. As shown in Figure 13 (a), the attack can craft AEs with a smaller perturbation bound by selecting the perturbation vectors for the longer processing time.



(a) ReLU function (b) Sigmoid function

Figure 13: Comparison of results between ReLU and sigmoid functions in proposed attack against MLP model. Perturbation bound are shown in the results. Attack depends on type of activation function: ReLU has longer processing time than sigmoid function. Proposed attack can craft AEs according todata-dependent timing leaks of activation functions.



(a) Random noise (b) Timing black-box attack

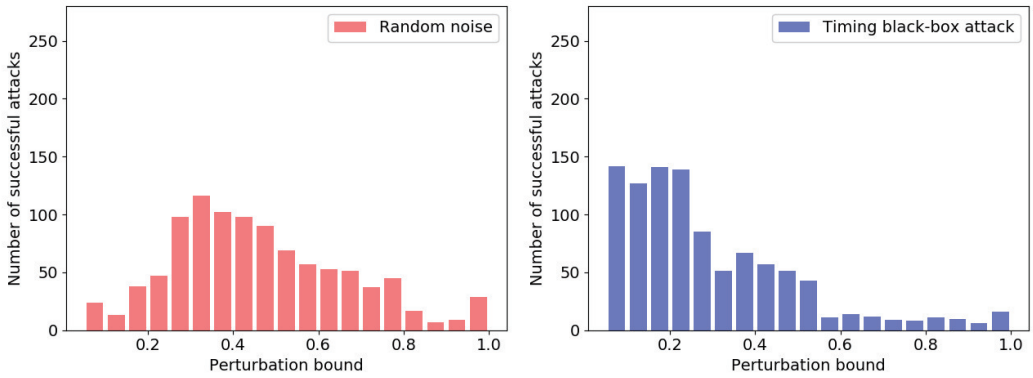
Figure 14: AEs of proposed attack and random noise against CNN model on MCU. Perturbation bound, ε, and misclassification labels are shown in the results.

In contrast, the sigmoid function has data-dependent timing leaks in which the processing time tends to be shorter if the node is activated. As shown in Figure 13 (b), the presented attack can craft AEs with a smaller perturbation bound by selecting the perturbation vectors for the shorter processing time. Therefore, it is clear that the proposed attack can craft AEs by selecting the perturbation vectors for increasing the number of activated nodes using the processing time of the activation function. It is confirmed that the other samples yield the same results.

5.5 Evaluation against CNN

The attack against the CNN model implemented on the MCU (NUCLEO-F767ZI) is also evaluated; the activation functions are ReLU functions.

The experimental results of the attack are shown in Figure 14. The perturbation bound, ε, is increased in 0.05 steps until it causes misclassification. As shown in the figure, the AE causes the misclassification of the CIFAR-10 label (from ”dog” to ”horse”) with small perturbations, which are half as large as the random noise. Therefore, the proposed attack can also craft effective AEs with small perturbations against the CNN model. Furthermore, the attack is successful on different MCUs for the MLP model.



(a) Random noise (b) Timing black-box attack

Figure 15: Histogram of successful attacks on 1,000 samples of CIFAR-10 dataset. Proposed attack is compared with random noise. Each perturbation bound is plotted when misclassification is caused by proposed attack and random noise.

Figure 15 shows the experimental results in 1,000 samples of the CIFAR-10 dataset as well as the histogram of successful attacks on the perturbation bound until it causes misclassification on each sample. The histogram indicates that practically all the attacks with small perturbations from 0.05 to 0.25 cause misclassification. By contrast, that of the random noise is from 0.25 to 0.55 for misclassification. The average perturbation bound of the attack is 0.26; the PSNR is 18.17 dB. By contrast, that of the random noise is 0.44; the PSNR is 9.70 dB. It is clear that the proposed attack also tends to craft AEs with small perturbations compared with the random noise in 1,000 samples of the CIFAR-10 dataset.

6 Detailed Analysis

The cause of the proposed attack is analyzed by using constant-time against data-dependent timing leaks on the MCU for the MLP model; the activation functions are ReLU functions.

6.1 Methods

An activation function with constant time is implemented against the timing black-box attacks. The constant time is assumed to prevent data-dependent timing leaks depending on the input data of the activation function because they require the same processing time regardless of the input data. The constant time serves to verify whether or not data-dependent timing leaks are prevented; it also works as a simple countermeasure.

The countermeasure implements two cycles of additional no-operation instructions to the original activation function if the branch is not available. Therefore, the improved activation function requires the same processing time regardless of the input data. The countermeasure is deployed using an inline assembly code on the MCU. The source code of the improved activation function with the constant time written in assembly language is as follows:

```
        mov r0, \%[in]
        mov r1, \%[zero]
        cmp r1, r0
        blt else
        nop                         //dummy operation
        nop                         //dummy operation
        mov \%[out], r1
else:
        mov \%[out], r0
```

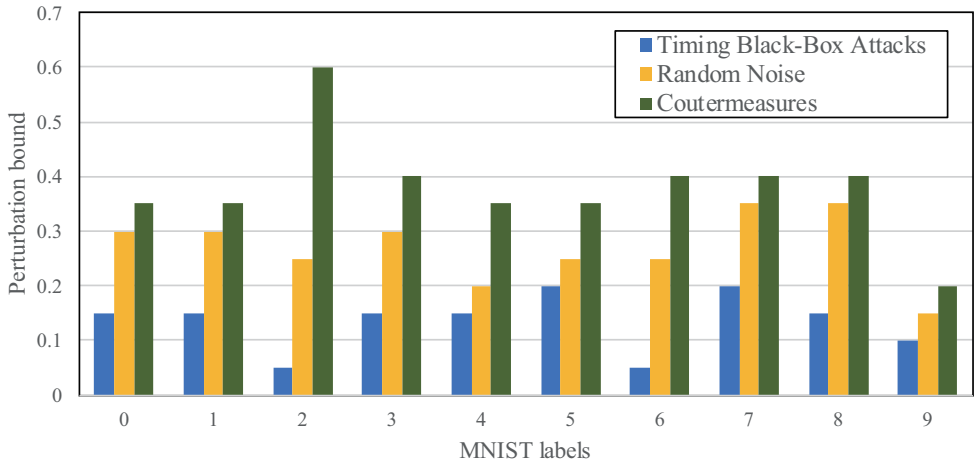


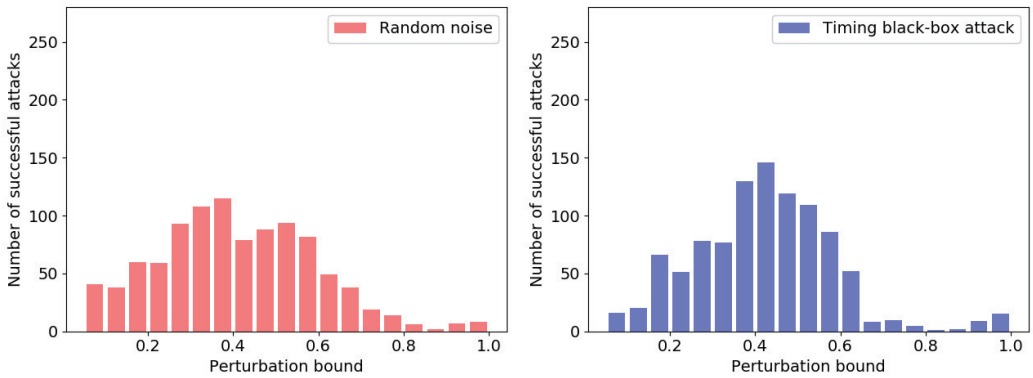
Figure 16: Comparison of timing black-box attacks and countermeasures

6.2 Evaluation of Constant Time Countermeasures

The experimental results of the proposed attack against the countermeasures are shown in Figure 16. The perturbation bound is compared between the countermeasures and non-countermeasures on each MNIST label (0-9). These results are shown in Figure 11 for comparison. The perturbation bound, ε, is increased in 0.05 steps until it causes misclassification.

The figure further shows that the perturbation bound is increased by the countermeasures. With the timing black-box attack, the perturbation bound is larger than the countermeasures. Moreover, the average perturbation bound is 0.38, which is approximately 2.8 times as large as the non-countermeasures. Furthermore, the perturbation bound is relatively larger than the random noise. These results depend on other processing times that are unrelated to the crafting of AEs. It is assumed to influence by other processing such as preprocessing or multiplication. Therefore, the cause of the proposed attack is the implementation of an activation function with data-dependent timing leaks depending on the input data.

The countermeasures in the other 1,000 samples of the MNIST dataset are also evaluated. Figure 17 shows the histogram of successful attacks on the perturbation bound until misclassification is caused on each sample. According to the histogram, practically all the attacks cause misclassification caused by perturbations (from 0.25 to 0.55); the random noise is also from 0.2 to 0.55. It is evident that countermeasures tend to prevent the occurrence of data-dependent timing leaks in the 1,000 samples of the MNIST dataset.



(a) Random noise (b) Timing black-box attack

Figure 17: Histogram of successful attacks against countermeasure on 1,000 samples of MNIST dataset. Proposed attack is compared with random noise. Each perturbation bound is plotted when misclassification is caused by proposed attack and random noise.

The code size overhead of countermeasures for the constant time is + 56 bytes. The execution time overhead is calculated by the number of non-activated nodes and the processing time of constant time. For example, of the 224 nodes shown in Figure 10, the MLP model initially has approximately 100 nodes that are activated. In this case, the execution time overhead is up to +124×2 cycles: approximately 2 *us* in ARM Cortex-M4 (120 MHz), which accounts for 0.001% of the total processing time.

7 Conclusion

In this paper, a novel black-box attack for crafting AEs using differential processing time according to the input data of DNNs is proposed. It is a new approach for crafting AEs using side-channel information, such as processing time. The proposed attack is more severe than conventional attacks because it only uses the processing time and renders the gradient-masking and confidence reduction countermeasures ineffective. The attack is evaluated against the MLP with ReLU and sigmoid functions and the CNN models on two MCUs as embedded devices. The experiments show that the data-dependent timing leaks are related to the number of activation nodes in the DNN. The proposed attack increases the number of activation nodes using data-dependent timing leaks. Furthermore, it can craft effective AEs with small perturbations in comparison to random noise under the same condition. The perturbation bound of the AE is approximately half as large as the random noise. It is clarified that the implementation of an activation function with data-dependent timing leaks is the cause of the attack, by implementing and evaluating constant time countermeasures.

Acknowledgments

This work was supported by JST-Mirai Program Grant Number JPMJMI19B6, Japan.

References

[AM18] Manaar Alam and Debdeep Mukhopadhyay. How secure are deep learning algorithms from side-channel based reverse engineering? *CoRR*, abs/1811.05259, 2018.

[Ard05] Arduino, 2005. https://www.arduino.cc.

[ASCS18] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, and Mani B. Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. *CoRR*, abs/1805.11090, 2018.

[BBJP19] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 515–532, Santa Clara, CA, 2019.

[CAD+18] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *CoRR*, abs/1810.00069, 2018.

[CIF09] The cifar-10 dataset, 2009. https://www.cs.toronto.edu/ kriz/cifar.html.

[CLC+19] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019.

[Cor19] Google coral dev board, 2019. https://coral.ai/products/dev-board.

[CW16] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[CX20] Lu Chen and Wei Xu. Attacking optical character recognition (ocr) systems with adversarial watermarks, 2020.

[DCA19] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection. *arXiv e-prints*, page arXiv:1910.13063, Oct 2019.

[DSRB18] Vasisht Duddu, Debasis Samanta, D. Vijay Rao, and Valentina E. Balas. Stealing neural networks via timing side channels. *CoRR*, abs/1812.11720, 2018.

[EEF+18] 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. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, June 2018.

[ENI20] Artificial intelligence cybersecurity challenges, 2020. https://www.enisa.europa.eu/publications/artificial-intelligence-cybersecurity-challenges.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, New York, NY, USA, 2015. ACM.

[FRD14] Frdm-k64f, 2014. http://www.nxp.com/frdm-k64f.

[GGY+19] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple black-box adversarial attacks. *CoRR*, abs/1905.07121, 2019.

[GPM+17] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 62–79, Germany, jan 2017. Springer Verlag.

[GSS15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[HDK+18] Sanghyun Hong, Michael Davinroy, Yigitcan Kaya, Stuart Nevans Locke, Ian Rackow, Kevin Kulda, Dana Dachman-Soled, and Tudor Dumitras. Security analysis of deep neural networks operating in the presence of cache side-channel attacks. *CoRR*, abs/1810.03487, 2018.

[HZS18] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, pages 4:1–4:6, New York, NY, USA, 2018. ACM.

[IBCK18] M. Isakov, L. Bu, H. Cheng, and M. A. Kinsy. Preventing neural network model exfiltration in machine learning hardware accelerators. In *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 62–67, 2018.

[IGGK19] M. Isakov, V. Gadepally, K. M. Gettings, and M. A. Kinsy. Survey of attacks and defenses on edge-deployed neural networks. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, 2019.

[Jet19] Nvidia jetson nano, 2019. https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems/jetson-nano.

[JOB+18] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35, May 2018.

[JYP+17] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, 2017.

[KGB16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[MNI98] The mnist database of handwritten digits, 1998. http://yann.lecun.c om/exdb/mnist.

[NIS19] A taxonomy and terminology of adversarial machine learning, 2019. https://doi.org/10.6028/NIST.IR.8269-draft.

[NK17] N. Narodytska and S. Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, July 2017.

[NUC16] Nucleo-f767zi, 2016. https://www.st.com/en/evaluation-tools/nucleo-f767zi.html.

[PMG16] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[PMSW16] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the science of security and privacy in machine learning. *CoRR*, abs/1611.03814, 2016.

[SS18] Congzheng Song and Vitaly Shmatikov. Fooling OCR systems with adversarial text images. *CoRR*, abs/1802.05385, 2018.

[SW19] Daniel Situnayake and Pete Warden. *TinyML*. O'Reilly Media, Inc., . edition, December 2019. CHAPTER20 Privacy, Security, and Deployment.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna Estrach, Dumitru Erhan, Ian Goodfellow, and Robert Fergus. Intriguing properties of neural networks. 1 2014. 2nd International Conference on Learning Representations, ICLR 2014 ; Conference date: 14-04-2014 Through 16-04-2014.

[Ten17] Tensorflow for mobile & iot, 2017. https://www.tensorflow.org/lite.

[TZJ+16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, Austin, TX, 2016.

[uTe17] Ai inference library based on mbed and tensorflow, 2017. https://github.com/uTensor/uTensor.

[YFT18] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. *CoRR*, abs/1808.04761, 2018.

[YKSF19] K. Yoshida, T. Kubota, M. Shiozaki, and T. Fujino. Model-extraction attack against fpga-dnn accelerator utilizing correlation electromagnetic analysis. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 318–318, April 2019.

[YS19] Hiromu Yakura and Jun Sakuma. Robust audio adversarial example for a physical attack. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pages 5334–5341. AAAI Press, 2019.

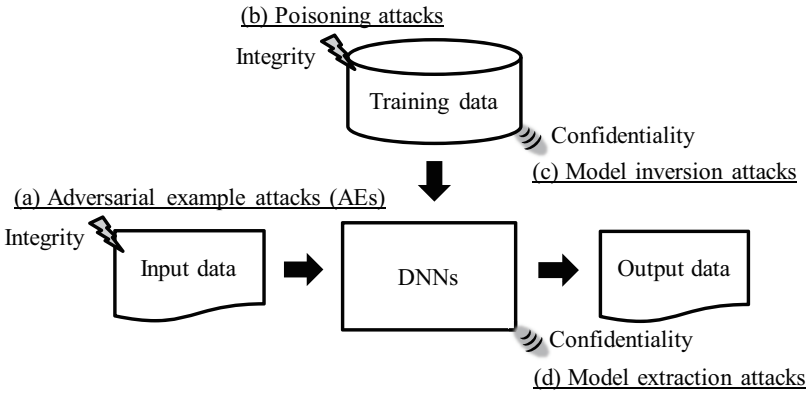


Figure 18: Taxonomy of attacks against DNNs

Appendix

A Taxonomy of Attacks against DNNs

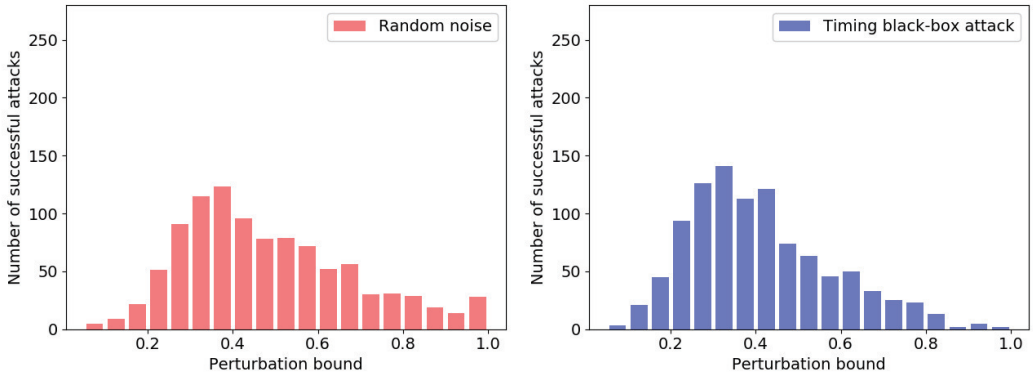
Figure 18 shows the taxonomy of attacks against DNNs in the training and prediction phases. The attacks against DNNs are mainly classified into four types: (a) adversarial example attacks known as AEs [SZS+14], (b) poisoning attacks [JOB+18], (c) model inversion attacks [FJR15], and (d) model extraction attacks [TZJ+16]. The AEs cause misclassification by adding small perturbations to the input data of DNNs. The poisoning attacks train poisonous models that cause misclassification by introducing malicious data into the training data. These attacks violate the integrity of DNNs by causing misclassification. The model inversion attacks steal training data from the DNN models and input-output data; hence, these attacks violate the confidentiality of training data. The model extraction attacks steal internal information from DNNs. The attacks violate the intellectual property of DNN models and promote other attacks because of DNN internal information leaks.

In this study, the focus is set on adversarial example attacks known as AEs. A lot of adversarial example attacks have been proposed recently.

B Attacks against Embedded devices with Neural network accelerator

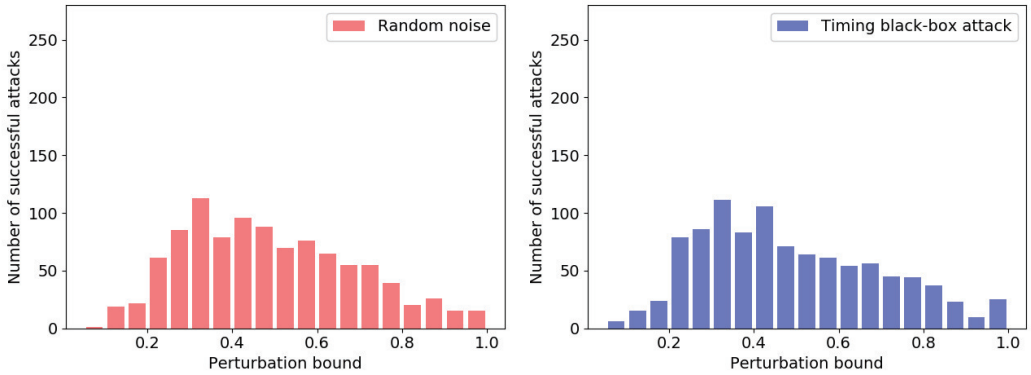
The attack against embedded devices with neural network accelerators is experimented with and discussed. The experiments are on Jetson nano GPU [Jet19] and Coral Edge TPU [Cor19]. Each experiment is on the MLP model which is shown in Table 1.

Figure 19 shows the histogram of successful attacks on Jetson nano GPU in 1,000 samples of the MNIST dataset. The histogram indicates that practically all the attacks with almost the same perturbation bound of the random noise cause misclassification. The attack is no successful on Jetson nano GPU because it is assumed that data-dependent timing leaks cannot be observed due to parallelization of activation functions by GPU.



(a) Random noise (b) Timing black-box attack

Figure 19: Histogram of successful attacks against Jetson nano GPU on 1,000 samples of MNIST dataset. Proposed attack is compared with random noise. Each perturbation bound is plotted when misclassification is caused by proposed attack and random noise.



(a) Random noise (b) Timing black-box attack

Figure 20: Histogram of successful attacks against Coral Edge TPU on 1,000 samples of MNIST dataset. Proposed attack is compared with random noise. Each perturbation bound is plotted when misclassification is caused by proposed attack and random noise.

Figure 20 shows the histogram of successful attacks on Coral Edge TPU in 1,000 samples of the MNIST dataset. According to the histogram, practically all the attacks with almost the same perturbation bound of the random noise cause misclassification. The attack is also no successful on Coral Edge TPU because it is assumed that activation functions are implemented in hardware and executed in parallel for the number of nodes in the systolic array. However, in systolic arrays, an implementation that omits operations on deactivated nodes has been proposed for speedup (not applicable to Coral Edge TPU) [JYP+17]. This implementation has improved the processing time by 1.4 times. It is assumed that data-dependent timing leaks can be observed in this implementation because the processing time depends on the number of activated nodes.

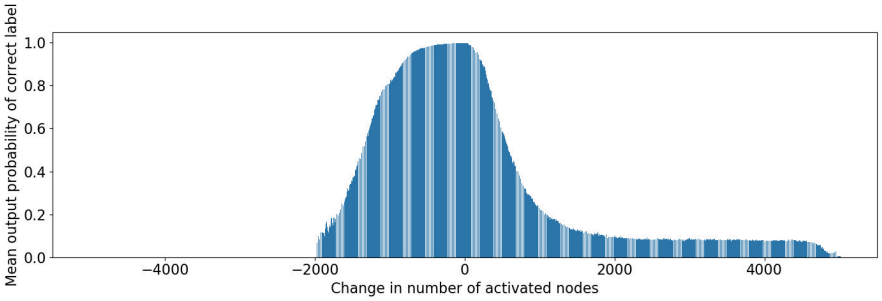


Figure 21: Simulation of relationship between the change in the number of all layers' activated nodes and the mean output probability of the correct label in the CNN model.

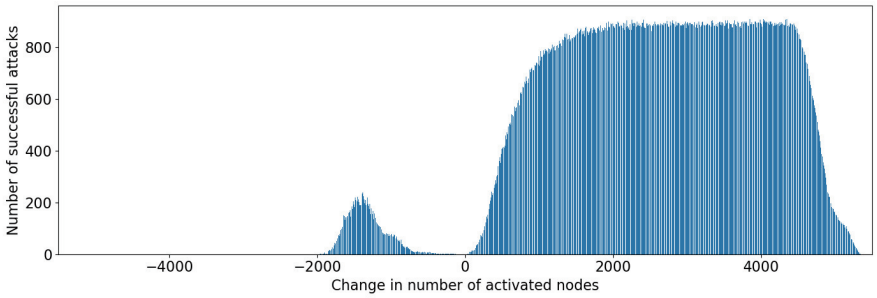


Figure 22: Simulation of relationship between the change in the number of all layers' activated nodes and the number of successful attacks (misclassification) in the CNN model.

C Simulation of Activated Nodes for CNN

The connection between the number of activated nodes and the output probability of correct labels is also simulated using a CNN model on a Linux OS computer (Ubuntu 18.04) with an Intel Core i-5 CPU. The CNN model is the same as in Table 2.

Figure 21 shows the relationship between the change in the number of all layers' activated nodes and the mean output probability of the correct label in the CNN model. Furthermore, Figure 22 shows the number of successful attacks (misclassification) on 1,000 samples per the change of all layers' activated nodes. The output probability is the mean value on 1,000 samples per the change of all layers' activated nodes. Because CNN mainly consists of convolution layers including activation functions, changes in the number of activated nodes affect the accuracy as well as the results of the MLP model in Section 4.2.2. Especially the phenomenon (the mechanism of our attack) is noticeable in CNN. It is assumed that the max-pooling layer selects the injected nodes instead of the original node.