Recommended Paper

Timing Attack on Random Forests: Experimental Evaluation and Detailed Analysis

Yuichiro Dan[1] Toshiki Shibahara[2,a)] Junko Takahashi[2]

Received: January 29, 2021, Accepted: September 9, 2021

Abstract: This paper proposes a novel implementation attack on machine learning. The threat of such attacks has recently become an problem in machine learning. These attacks include side-channel attacks that use information acquired from implemented devices and fault attacks that inject faults into implemented devices using external tools such as lasers. Thus far, these attacks have targeted mainly deep neural networks; however, other common methods such as random forests can also be targets. In this paper, we investigate the threat of implementation attacks to random forests. Specifically, we propose a novel timing attack that generates adversarial examples. Additionally, we experimentally evaluate and analyze its attack success rate. The proposed attack exploits a fundamental property of random forests: the response time from the input to the output depends on the number of conditional branches invoked during prediction. More precisely, we generate adversarial examples by optimizing the response time. This optimization affects predictions because changes in the response time indicate changes in the results of the conditional branches. For the optimization, we use an evolution strategy that tolerates measurement error in the response time. Experiments are conducted in a black-box setting where attackers can use only prediction labels and response times. Experimental results show that the proposed attack generates adversarial examples with higher probability than a state-of-the-art attack that uses only predicted labels. Detailed analysis of these results indicates an unfortunate trade-off that restricting tree depth of random forests may mitigate this attack but decrease prediction accuracy.

Keywords: side-channel attack, adversarial examples, black-box attack, evolution strategy

1. Introduction

Attacks on machine learning have been a critical problem since the work by Szegedy et al. [1] in 2013. The most representative of these attacks is evasion, hereinafter referred to as an adversarial example [1], [2], [3], [4], [5], [6], [7]. In an adversarial example, the attacker deceives the prediction model by adding small perturbations to the input data. Another representative attack is model extraction, which extracts or learns prediction models in machine-learning-based systems and constructs substitute models using input and output data of the systems [8].

Until recently, researchers have investigated attacks exploiting weaknesses in algorithms such as the fact that gradients of loss functions indicate directions for misclassification, and the fact that a good approximation of a prediction model can be generated using a set of its input and output data (algorithm attacks) [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. Since around 2017, researchers have indicated attacks that use weaknesses in devices that use prediction models (implementation attacks) [13], [14], [15], [16], [17], [18], [19]. For example, an attack was proposed that generates faults through laser injection causing misclassification [14]. Additionally, several attacks have been discovered that extract prediction models by measuring physical quantities such as timing and electromagnetic radiation [13], [15], [16], [19]. These implementation attacks pose significant threats especially when attackers can easily access prediction models as edge artificial intelligences (AIs).

The target of these implementation attacks has been limited to deep neural networks. In other words, implementation attacks on tree-based methods have not been reported even though such methods include practical machine learning methods [20] such as XGBoost [21] and random forests [22]. Thus, in this paper, we address the threat of implementation attacks to random forests. In particular, we propose a novel attack exploiting the response time from the input to the output (timing) for generating adversarial examples, and evaluate its attack success rate experimentally. We focus on the attack using the timing among implementation attacks (timing attacks) because measuring the timing does not require expensive equipment or advanced skills. Consequently, timing attacks are easy to execute, and more tractable for many attackers than other implementation attacks.

There are two challenges in generating adversarial examples

[1] NTT Secure Platform Laboratories (when submitted), Musashino, Tokyo 180–8585, Japan
[2] NTT Social Informatics Laboratories, Musashino, Tokyo 180–8585, Japan
[a)] toshiki.shibahara.de@hco.ntt.co.jp

The initial version of this paper was presented at the 15th International Workshop on Security (IWSEC 2020) held in Fukui, Japan on 2-4 September 2020. This workshop was co-organized by ISEC in ESS of IEICE and CSEC of IPSJ. The initial version is available online at https://doi.org/10.1007/978-3-030-58208-1_16. This paper was recommended to be submitted to IPSJ Journal by the chairperson of IWSEC 2020.
The content of this paper was presented at the 15th International Workshop on Security (IWSEC2020) in September 2020, and was recommended for publication in the Journal of Information Processing (JIP) by the program chair.

using the timing: building a strategy required to cause misclassification using the timing, and mitigating the measurement error of the timing. In regard to the former challenge, drastic changes in the timing can cause misclassification for the following reason. Because the timing correlates with the number of conditional branches invoked in a random forest during prediction, drastic changes in the timing indicate drastic changes in the results of the invoked branches. These changes in the results of the branches can ruin prediction results. The latter challenge requires a noise-tolerant optimization method to change the timing appropriately even if measurement errors exist. Such optimization methods include evolution strategies. When updating an optimum once, evolution strategies evaluate multiple candidates close to the current optimum and obtain a new optimum using a weighted average of the candidates. In this way, evolution strategies can robustly find a good optimum even if each of the evaluation results of candidates contains measurement errors. For this reason, we use the covariance matrix adaptation evolution strategy (CMA-ES), one of the most common evolution strategies.

We evaluate the threat of the proposed attack in a black-box setting where attackers can use only prediction labels and the timing. Note that we assume a victim system that conceals parameters of the prediction model and confidence values of predictions in this setting. The evaluation results show that the proposed attack generates adversarial examples with higher probability than a low frequency boundary attack (LFBA) [5], a state-of-the-art algorithm attack.

Detailed analysis of the evaluation results shows the proposed attack is less successful for images that require fewer conditional branches for prediction. This indicates restricting tree depth of random forests may mitigate the proposed attack. However, we also reveal that such restriction decreases prediction accuracy of random forests. This clarifies demand for more sophisticated mitigation measures.

The remainder of this paper is structured as follows. Section 2 introduces related work regarding implementation attacks and algorithm attacks on machine learning. Section 3 reviews random forests, CMA-ES, and LFBA as research background. Section 4 describes the assumption of the threat in this paper. In Section 5, we propose the novel timing attack. Section 6 describes the setting and the results of the experiments along with their interpretations. In Section 7, we discuss risky situations involving timing attacks, and possible directions for countermeasures against the attacks. Section 8 presents our conclusions.

2. Related Work

This section describes representative attacks on algorithms and implementations of machine learning.

2.1 Implementation Attacks on Machine Learning

Implementation attacks on machine learning have received interest since 2017 [17]. Thus far researchers have described attacks such as fault attacks [14], [17] and side-channel attacks [13], [15], [16], [18], [19]. A fault attack injects faults into implemented devices using external tools such as lasers. A side-channel attack uses information acquired from implemented devices.

For example, Liu et al. [17] conceptually advocated a fault attack causing misclassification. Although their research avoided referring to practical methods that cause faults, another study [14] specified a method that causes faults by injecting a laser into devices that run prediction models. As for side-channel attacks, there are a few attacks that extract prediction models using information on timing [15], cache hits [16], [19], or electromagnetic waves [13]. Additionally, Wei et al. [18] proposed an attack using information on power consumption for extracting input data.

As mentioned above, several studies have already addressed implementation attacks on machine learning. However, these studies mainly target deep neural networks. Regarding other common methods such as random forests [22], revealing the threat of implementation attacks to such methods requires further investigation.

2.2 Algorithm Attacks on Machine Learning

Unlike implementation attacks, algorithm attacks on machine learning have attracted attention since 2013 [1]. There are many attacks such as poisoning [10], evasion [1], [2], [3], [4], [5], [6], [7], model extraction [8], model inversion [9], membership inference [11], and hyperparameter stealing [12]. A poisoning attack forces a prediction model to learn inappropriately by mixing tampered data into the training dataset. An evasion attack deceives a trained prediction model by adding small perturbations to the input data. A model extraction attack extracts a prediction model from the input and output data of the model. An inversion attack restores the training data of a prediction model. A membership inference attack judges if given data belong to the training dataset of a prediction model. A hyperparameter stealing attack extracts hyperparameters that tune the training algorithm of a prediction model.

The most well-known attacks in machine learning include evasion attacks, also known as adversarial examples, which were first proposed in 2013 [1]. This triggered a series of studies on attacks on machine learning as mentioned above. Evasion attacks are broadly divided into white-box [1], [4], [7] and black-box attacks [2], [3], [5], [6] depending on the information used for the attacks. White-box attacks use information on prediction models such as the architecture, parameters, and output data of the models. Black-box attacks use only the output data. Black-box attacks are further divided into score-based [3], [6] and label-based [2], [5] attacks. Score-based attacks use prediction labels and confidence values, whereas label-based attacks use only prediction labels.

To evaluate the threat posed by these attacks, three metrics are generally used: the misclassification rate, perturbation size, and number of queries. The misclassification rate is the percentage of original input data that successfully cause misclassification through perturbation addition by attackers. The perturbation size is the magnitude of the perturbation generally calculated using mean squared errors (MSEs) of perturbed data to original data. The number of queries is the number of trials conducted by an attacker to obtain the output of the prediction model. In terms of these metrics, attacks with a high misclassification rate, small

perturbation size, or low number of queries pose a severe risk because they represent a high probability for misclassification to occur, difficulty to notice perturbations with the human eye, or low cost and high stealthiness, respectively.

These metrics generally have a trade-off relationship. Mitigating this trade-off is a focus of adversarial example generation methods. For example, in the case of label-based attacks, LFBA [5], a state-of-the-art algorithm attack, limits the perturbation search to the region consisting only of low frequency components in the frequency space of the input images, and generates adversarial examples with smaller perturbation and fewer queries than previous methods do.

3. Background

This section briefly reviews random forests [22], CMA-ES [23], and LFBA [5].

3.1 Random Forests

Random forests represent a supervised machine learning algorithm consisting of multiple decision trees. A decision tree is a prediction model that has a tree structure. It receives input data at the root node, invokes conditional branches at intermediate nodes depending on the value of each element of the data, and outputs prediction results corresponding to a leaf node to which the data finally reach. Accurate prediction requires appropriate branch conditions at the root and intermediate nodes. For each node, training algorithms search for the best condition that divides the training data subset reaching the node into two subsets whose label homogeneity improves the most. To measure this homogeneity, these algorithms usually use metrics such as entropy and Gini index.

In a random forest, decision trees are trained separately but work together as one prediction model. This makes the prediction accuracy higher than that of a single decision tree. Data input to a random forest are received by each decision tree, and the output value from each decision tree is aggregated as a prediction result. This aggregation involves a majority vote using the output values for classification cases, and average calculation using these values for regression cases. This paper focuses on the classification cases below.

3.2 CMA-ES

CMA-ES is an evolution strategy. Evolution strategies utilize a biological evolution mechanism to optimize objective functions defined on real vector spaces. Generally speaking, evolution strategies are tolerant of noisy objective functions. These methods update tentative optima on the basis of objective function values of multiple candidates for optima. This reduces susceptibility of tentative optima to noise in objective function values. In this category, CMA-ES [23] is a representative method that efficiently solves optimization problems with a low number of iterations. Here, we review $(\mu/\mu_w, \lambda)$-CMA-ES, a typical variation of CMA-ES.

CMA-ES optimizes a function by iterating the next three steps (a generation). In this subsection, we explain these steps by considering minimizing the objective function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is a multidimensional real vector (e.g., an input image to prediction models).

Step 1 A normal distribution with mean $\boldsymbol{m}$ (centroid) and covariance matrix C generates λ candidates for the optimum (individuals).

Step 2 The value of $f(\boldsymbol{x})$ of each individual is calculated, and μ individuals are selected in ascending order with respect to the function values.

Step 3 A weighted average of the μ individuals is obtained, and the centroid $\boldsymbol{m}$ and covariance matrix C are updated on the basis of this result.

This method generally requires tuning the initial values of $\boldsymbol{m}$ and C, and designing the objective function. Moreover, designing a conversion function often facilitates the optimization. The conversion function maps the space where individuals are expressed to the space where objects (i.e., inputs to the objective function) are expressed. This function is useful because spaces suitable for individuals are often different from spaces suitable for objects. For example, when objects are gray scale images, a suitable space for objects is the two-dimensional real space, but the frequency space transformed from the real space with a discrete cosine transform can be suitable for individuals. This is true if the low frequency components dominate changes in the objective function value. Other parameters such as λ and μ have default values depending on the dimensions of individuals as recommended by Hansen and Ostermeier [23].

3.3 LFBA

LFBA [5] is a state-of-the-art label-based algorithm attack that generates adversarial examples. This attack is derived from the Boundary Attack [2]. Below, we review the Boundary Attack before LFBA, considering the generation of an adversarial example of the original image $\boldsymbol{x}_O$ with $d \times d$ pixels.

The Boundary Attack begins by initializing a tentative image $\boldsymbol{x}_A$, which already causes misclassification. An attacker chooses an initial image, or simply generates random noise. Then, the attacker iterates the update of $\boldsymbol{x}_A$ by adding small perturbations. An iteration comprises four steps.

Step 1 The attacker generates a perturbation $\boldsymbol{\eta}$ from a normal distribution over the real space of the image, and rescales it to a small perturbation.

Step 2 The perturbation $\boldsymbol{\eta}$ undergoes projection onto the sphere around $\boldsymbol{x}_O$ with radius $\|\boldsymbol{x}_A - \boldsymbol{x}_O\|_2$.

Step 3 The adversarial example candidate $\tilde{\boldsymbol{x}}$ is obtained as $\boldsymbol{x}_A + \boldsymbol{\eta}$, and this candidate subsequently approaches slightly closer directly toward $\boldsymbol{x}_O$.

Step 4 If $\tilde{\boldsymbol{x}}$ causes misclassification, it takes the place of the current $\boldsymbol{x}_A$; otherwise $\boldsymbol{x}_A$ remains unchanged.

LFBA improves the efficiency of the above attack by restricting the search space to the low frequency region. In particular, in Step 1 above, it generates a perturbation only with low frequency components. To generate such a perturbation, LFBA uses a normal distribution over the low frequency subspace in the frequency space of the image. An inverse discrete cosine transform is used to convert the perturbation into a perturbation in the real space. This restriction to the low frequency region effectively reduces

the dimensions of the search space. Thus, LFBA requires fewer queries than the Boundary Attack.

4. Threat Model

In this section, we describe the threat model assumed in this paper in terms of the victim and attacker. First, in regard to the victim, we assume a service provider that receives query $\boldsymbol{x}$ from a user and responds with a corresponding prediction label, l. The prediction is carried out by a prediction model $F(\boldsymbol{x}; \boldsymbol{w})$, where parameter $\boldsymbol{w}$ is trained with a labeled training dataset $D = \{(\boldsymbol{x}, l)\}$. As a countermeasure to algorithm attacks, the victim conceals model $F(\boldsymbol{x}; \boldsymbol{w})$, dataset D, and the confidence values of the predictions. This is because disclosure of such information makes generating adversarial examples more effective [4], [6]. On the other hand, the victim does not know of the threat of the timing attacks, and no countermeasures for such attacks are applied. Thus, the victim unintentionally leaks information regarding the internal processing in the prediction model through the response time for query.

Second, the assumption of the attack is as follows. As described in Section 2.2, adversarial example attacks are broadly divided into white-box and black-box attacks. In this paper, we focus on label-based attacks among the black-box attacks. This is because it is difficult to directly use the prediction model and confidence values, which are both concealed by the victim. Instead, the attacker attempts to generate adversarial examples by efficiently exploiting the response time (i.e., timing).

5. Timing Attack on Random Forests

In this section, we describe how the proposed attack generates adversarial examples using the timing. There are two challenges in generating adversarial examples: building a strategy required to cause misclassification using the timing, and mitigating the measurement error of the timing. In regard to the former challenge, an appropriate change in the timing can cause misclassification, considering the internal processing in a random forest. This is because changes in the timing indicate changes in the sum of the distance from the root node to the leaf node to which input data reach through conditional branches in each decision tree (depth). To change the timing appropriately, we should know the relationship between the timing and the possibility of misclassification. In this paper, this relationship is discussed in Section 5.2, and experimentally confirmed in Section 6.1. Note that this is not a part of the attack, but simply a preliminary experiment to design the proposed attack.

The latter challenge requires a noise-tolerant optimization method. This is because the timing must be changed along the appropriate direction even if measurement error in the timing exists. To confront this problem, we use CMA-ES [23] in this paper. CMA-ES is a common method in evolution strategy, which is a category of optimization methods using noise tolerance.

The reminder of this section overviews the proposed attack, the objective function, and the conversion function. We originally designed the last two functions for the application of CMA-ES to the proposed attack. Hereinafter, we consider attacking gray scale images expressed in $[0, 1]^{d \times d}$ for simplicity.

5.1 Notation and Flow of Proposed Attack

In this subsection, we describe the notation and flow in the algorithm of the proposed attack. The notation is defined as follows: $\boldsymbol{x}_O$ is the original non-adversarial image before the attack; l_O is its label; Γ is the distribution that generates individuals of CMA-ES; $\boldsymbol{m}_O$ is the initial value of the centroid of Γ; C_O is the initial value of the covariance matrix of Γ; λ is the number of individuals generated per generation; $gen_{\max}$ is the upper limit of the generation; $f(\boldsymbol{x}; \boldsymbol{x}_O, l_O)$ is the objective function of $\boldsymbol{x}$ when $\boldsymbol{x}_O$ and l_O are given; $\tilde{\phi}$ is the tentative minimum value of $f(\boldsymbol{x}; \boldsymbol{x}_O, l_O)$; $\tilde{\boldsymbol{z}}$ is the tentative optimum individual corresponding to $\tilde{\phi}$; $g(\boldsymbol{z}; \boldsymbol{x}_C, \boldsymbol{z}_C)$ is the conversion function that outputs the image corresponding to individual $\boldsymbol{z}$ when the central image $\boldsymbol{x}_C$ and individual offset $\boldsymbol{z}_C$ are given; $\mathrm{LFBA}(\boldsymbol{x}, N; \boldsymbol{x}_O, l_O)$ is the result of LFBA after N iterations starting from initial image $\boldsymbol{x}$ to generate an adversarial example of $\boldsymbol{x}_O$; and $\boldsymbol{x}_A$ is the generated adversarial example.

Algorithm 1 shows the flow of the proposed attack. First, we initialize Γ with $\boldsymbol{m}_O$ and C_O in line 1. We also initialize $\boldsymbol{x}_C$ with $\boldsymbol{x}_O$, $\boldsymbol{z}_C$ with $\boldsymbol{m}_O$, $\tilde{\phi}$ with ∞, $\tilde{\boldsymbol{z}}$ with $\boldsymbol{m}_O$ in line 2. Next, CMA-ES repeats the optimization at most $gen_{\max}$ times in the following flow.

Algorithm 1 Proposed timing attack for generating adversarial examples

Require:
- $\boldsymbol{x}_O$: original image
- l_O: original label
- Γ: distribution that generates individuals of CMA-ES
- $\boldsymbol{m}_O$: initial centroid of Γ
- C_O: initial covariance matrix of Γ
- $f(\boldsymbol{x}; \boldsymbol{x}_O, l_O)$: objective function of generated image $\boldsymbol{x}$ when $\boldsymbol{x}_O$ and l_O are given
- $\tilde{\phi}$: tentative minimum value of $f(\boldsymbol{x}; \boldsymbol{x}_O, l_O)$
- $\tilde{\boldsymbol{z}}$: tentative optimum individual corresponding to $\tilde{\phi}$
- $g(\boldsymbol{z}; \boldsymbol{x}_C, \boldsymbol{z}_C)$: transformer of individual $\boldsymbol{z}$ when $\boldsymbol{x}_C$ and $\boldsymbol{z}_C$ are given
- $\mathrm{LFBA}(\boldsymbol{x}, N; \boldsymbol{x}_O, l_O)$: the result of LFBA with N iterations starting from initial image $\boldsymbol{x}$ to generate an adversarial example of pair $(\boldsymbol{x}_O, l_O)$

Ensure:
- $\boldsymbol{x}_A$: an adversarial example of $\boldsymbol{x}_O$

1: Initialize Γ with $\boldsymbol{m}_O$ and C_O
2: $\boldsymbol{x}_C \leftarrow \boldsymbol{x}_O$, $\boldsymbol{z}_C \leftarrow \boldsymbol{m}_O$, $\tilde{\phi} \leftarrow \infty$, $\tilde{\boldsymbol{z}} \leftarrow \boldsymbol{m}_O$
3: **for** $gen = 1$ to $gen_{\max}$ **do**
4: Generate λ individuals $\boldsymbol{z}_i$ $(i = 1, \cdots, \lambda)$ with Γ
5: **for** $i = 1$ to λ **do**
6: $\phi_i \leftarrow f(g(\boldsymbol{z}_i; \boldsymbol{x}_C, \boldsymbol{z}_C); \boldsymbol{x}_O, l_O)$
7: **end for**
8: Update Γ by $\boldsymbol{z}_i$ and ϕ_i $(i = 1, \cdots, \lambda)$
9: **if** $\phi_\iota < \tilde{\phi}$ for $\iota = \mathrm{argmin}_i \phi_i$ **then**
10: $\tilde{\phi} \leftarrow \phi_\iota$, $\tilde{\boldsymbol{z}} \leftarrow \boldsymbol{z}_\iota$
11: $\boldsymbol{x}_A \leftarrow g(\tilde{\boldsymbol{z}}; \boldsymbol{x}_C, \boldsymbol{z}_C)$
12: $\boldsymbol{x}_C \leftarrow g(\boldsymbol{m}; \boldsymbol{x}_C, \boldsymbol{z}_C)$, $\boldsymbol{z}_C \leftarrow \boldsymbol{m}$ ($\boldsymbol{m}$ is the centroid of Γ)
13: **end if**
14: **if** $\boldsymbol{x}_A$ is misclassified **then**
15: Break this loop
16: **end if**
17: **end for**
18: $\boldsymbol{x}_A \leftarrow \mathrm{LFBA}(\boldsymbol{x}_A, gen_{\max} - gen; \boldsymbol{x}_O, l_O)$
19: **return** $\boldsymbol{x}_A$

To begin with, Γ generates λ individuals $z_1, z_2, \cdots, z_\lambda$ in line 4. Then, we evaluate value ϕ_i of the objective function of each individual in line 6. Subsequently, we update Γ on the basis of z_i and ϕ_i in accordance with the procedure of CMA-ES in line 8. At this point, if $\phi_\iota < \tilde{\phi}$ for $\iota = \mathrm{argmin}_i \phi_i$, ϕ_ι and z_ι respectively replace $\tilde{\phi}$ and $\tilde{z}$ in line 10. Additionally, we substitute $g(\tilde{z}; \boldsymbol{x}_C, z_C)$ for $\boldsymbol{x}_A$ in line 11. Simultaneously, in line 12, we replace $\boldsymbol{x}_C$ with the image converted from the post-update centroid of Γ when pre-update $\boldsymbol{x}_C$ and z_C are given. In this line, we also replace z_C with the updated centroid of Γ. Then, if $\boldsymbol{x}_A$ causes misclassification, the timing attack with CMA-ES halts and LFBA assumes the attack in line 18; otherwise, the timing attack continues returning to line 3. This is because after misclassification, we only have to optimize a noiseless quantity, that is, the distance between $\boldsymbol{x}_A$ and $\boldsymbol{x}_O$. Such a task is more suitable for LFBA. Finally, we obtain $\boldsymbol{x}_A$ as the output of this attack in line 19.

In the above attack, we need to design the objective function $f(\boldsymbol{x}; \boldsymbol{x}_O, l_O)$ and the conversion function $g(z; \boldsymbol{x}_C, z_C)$ for applying CMA-ES to the timing attack. Additionally, changing the arguments of the conversion function (line 12) is an original contrivance. The reminder of this section describes the details of their designs and intensions.

5.2 Design of Objective Function

The objective function for CMA-ES used in this paper is:

$$f(\boldsymbol{x}; \boldsymbol{x}_O, l_O) = \begin{cases} S(t(\boldsymbol{x})) & (l(\boldsymbol{x}) = l_O) \\ \mathrm{MSE}(\boldsymbol{x}, \boldsymbol{x}_O) - 1 & (l(\boldsymbol{x}) \neq l_O) \end{cases}. \tag{1}$$

Here, $l(\boldsymbol{x})$ is the prediction label obtained by inputting $\boldsymbol{x}$ to the prediction model, $t(\boldsymbol{x})$ is the time spent from the query of $\boldsymbol{x}$ to the response of $l(\boldsymbol{x})$, $S(t)$ is a function that estimates the possibility of misclassification from t, and $\mathrm{MSE}(\boldsymbol{x}, \boldsymbol{x}_O)$ is the mean squared error of $\boldsymbol{x}$ to $\boldsymbol{x}_O$ (i.e., the perturbation size). In this expression, it may seem that the case where $l(\boldsymbol{x}) \neq l_O$ is unnecessary, when the attack is assumed by LFBA. However, we consider this case to select the best solution even when multiple individuals cause misclassification at the same time.

In Eq. (1), $S(t)$ adheres to the two conditions below.

Condition 1 The possibility of misclassification increases as $S(t)$ decreases.

Condition 2 $S(t)$ is always positive.

Condition 1 is to cause misclassification by maximizing the possibility of its occurrence. Condition 2 always makes the function value in cases where $l(\boldsymbol{x}) \neq l_O$ smaller than that in cases where $l(\boldsymbol{x}) = l_O$. This ensures that misclassified images become the optimum solutions. Indeed, under the assumption that each pixel value is in $[0, 1]$, $\mathrm{MSE}(\boldsymbol{x}, \boldsymbol{x}_O)$ is less than 1. Therefore, it is true under Condition 2 that the function value becomes smaller in cases of misclassification than otherwise.

Strictly speaking, the above conditions are not sufficient to specify $S(t)$. In this paper, we infer that $S(t) = t$. The reason for this inference is twofold. First, from Condition 1, the simplest and most plausible forms of $S(t)$ are $\pm t$. This is because if we change t monotonically to a drastic extent, the internal processing also drastically changes. Second, maximizing t is likely to be more difficult than minimizing it. This difficulty can be derived from the fact that each conditional branch divides the space expressing images into two portions. This division shrinks the volume of subspaces consisting of images of deep depth, making such subspaces difficult to find. Indeed, we experimentally observed the difficulty as described in Section 6.1. Here, note that $S(t) = t$ meets Condition 2.

5.3 Design of Conversion Function

The conversion function for CMA-ES used in this paper is

$$g(z; \boldsymbol{x}_C, z_C) = \mathrm{clip}(\boldsymbol{x}_C + \epsilon \tanh(\mathrm{IDCT}_r(z - z_C))). \tag{2}$$

Here, z is an individual, $\boldsymbol{x}_C$ is the central image to add perturbation, z_C is an offset of the individual, $\mathrm{clip}(\boldsymbol{x})$ is a function to clip $\boldsymbol{x}$ within $[0, 1]^{d \times d}$, $\epsilon \ll 1$ is a constant to limit the perturbation size, and $\mathrm{IDCT}_r(z)$ is an inverse discrete cosine transform from the low frequency region of the frequency space to the real space. In Eq. (2), the clip function takes two terms. The first term $\boldsymbol{x}_C$ is the current optimal image, and the second term is a small perturbation. In the second term, we use an inverse discrete cosine transform to efficiently optimize adversarial examples by reducing the search space to a low frequency region, which is known to be effective for adversarial examples [5]. In other words, we omit a high frequency region to search. Furthermore, we limit the norm of the second term using a small constant ϵ and tanh. Based on Eq. (2), we can effectively search for better adversarial examples in the vicinity of the current optimal one. Below, we describe the roles of z_C and $\boldsymbol{x}_C$, as well as the definition and reason for adopting $\mathrm{IDCT}_r(z)$ in detail.

First, the definition of $\mathrm{IDCT}_r(z)$ is

$$\mathrm{IDCT}_r(z) = \mathrm{IDCT}(\boldsymbol{\eta}),$$

where IDCT is the inverse discrete cosine transform for the vertical and horizontal axes, and

$$\eta_{i,j} = \begin{cases} z_{i,j} & (1 \leq i, j \leq rd) \\ 0 & \text{otherwise} \end{cases}.$$

Here, r is a constant corresponding to the reduction rate of the dimension. We introduce this transformation to improve the efficiency of the optimization by the dimension reduction, which is based on an idea from previous research [5].

Finally, the roles of z_C and $\boldsymbol{x}_C$ are described with the reasons for their introduction and validity in line 12 in Algorithm 1. Equation (2) means that at most $\pm\epsilon$ perturbations are added to image $\boldsymbol{x}_C$. On the other hand, as shown in line 2 of Algorithm 1, because $\boldsymbol{x}_C$ is initialized by $\boldsymbol{x}_O$, the proposed attack initially searches for adversarial examples in the vicinity of $\boldsymbol{x}_O$. Therefore, if we fix $\boldsymbol{x}_C$ to $\boldsymbol{x}_O$, this limits the search for adversarial examples to a narrow region around $\boldsymbol{x}_O$. Thus, we intended to expand the search area by replacing $\boldsymbol{x}_C$ with the image generated with the centroid of Γ when $\tilde{z}$ changes in line 10. This is what line 12 in Algorithm 1 represents. More precisely, for CMA-ES to operate properly, the image transformed from individual z under post-update $\boldsymbol{x}_C$ must be at least approximately the same as the image transformed from the same individual z under pre-update $\boldsymbol{x}_C$. Therefore, in line 12,

z_C simultaneously shifts to the centroid of Γ as $\boldsymbol{x}_C$ changes. Indeed, suppose the old $\boldsymbol{x}_C$ is $\boldsymbol{x}_{\text{old}}$, the new $\boldsymbol{x}_C$ is $\boldsymbol{x}_{\text{new}}$, the old z_C is z_{old}, and the new z_C is z_{new}, then the following equation holds if the effect of clipping and terms $O(\|z_{\text{new}} - z_{\text{old}}\|^2)$ are ignored:

$$g(z; \boldsymbol{x}_{\text{new}}, z_{\text{new}}) \simeq g(z; \boldsymbol{x}_{\text{old}}, z_{\text{old}}).$$

6. Experiments

To prove the threat of the timing attack to random forests, we conducted two experiments described in the following subsections. More specifically, we first verify the difficulty in finding images of deep depth to support presumption $S(t) = t$. Then, we evaluate the threat of the proposed attack by comparing the proposed attack to LFBA [5].

The victim environment prepared for these experiments comprises a machine, an operating system (OS), a machine learning framework, a prediction model, and a dataset. The machine is an HP ProDesk 600 G2 SFF with an Intel Core i5 -6500 CPU and 16.0 GB memory. The OS is Windows 7 Professional. The machine learning framework is scikit-learn [24].

The prediction model is `RandomForestClassifier` with 100 decision trees, and other parameters are set to default values. The number of trees was determined based on the preliminary experiment. We trained models with 10, 100, and 1,000 trees. The prediction accuracy of the model with 10 trees was low, and the accuracies of the models with 100 and 1,000 trees were high and similar. Since the computational cost of random forests with many trees is high, we selected the most cost-effective model that achieved high prediction accuracy with less computational cost, i.e., the model with 100 trees. Such selection is reasonable for machine-learning-based systems. Note that this setting also avoids the condition where attackers are too advantageous. Specifically, the more trees the model has, the more easily the proposed attack succeeds. Changes in the timing increase in proportion to the number of trees because a model with many trees has many branches. The large change in the timing facilitates the proposed attack because it relatively makes measurement errors small.

The dataset is the Modified National Institute of Standards and Technology (MNIST) database. The MNIST training dataset is used to train the prediction model, whereas the MNIST test dataset is used in the following experiments. The proposed attack is expected to be effective without depending on datasets because the proposed attack does not exploit characteristics of datasets but exploit those of random forests. We select MNIST in this paper because MNIST has been frequently used in the experiments of adversarial examples and because reproductive experiments are easy to conduct.

By combining these components, we prepared the victim environment in which depths are leaked from the timing. The correlation coefficient between the timing and the depth is 0.89.

As for the proposed attack, parameters $\epsilon = 0.05$, $r = 1/4$, $z_O = 0$, $C_O = 25.0 I_{rd}$ are set on the basis of preliminary experiments, where I_{rd} is an $rd \times rd$ identity matrix. The other parameters of CMA-ES are set to the default values as recommended by Hansen and Ostermeier [23].

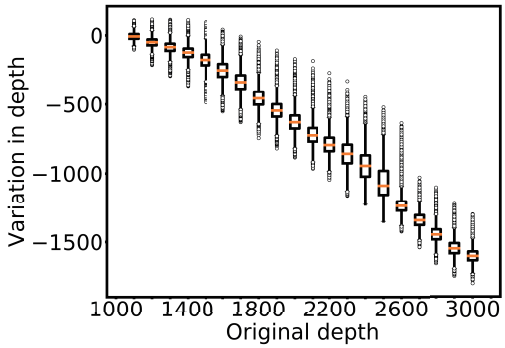


Fig. 1 Boxplot of variations in depth caused by adding perturbations to input images to a random forest. The original depth of the images without perturbations is represented on the horizontal axis. The variation in depth caused by adding perturbations to the images is represented on the vertical axis. Regardless of the original depth, the variations in depth tend to be negative.

6.1 Verification of Difficulty in Finding Images of Deep Depth

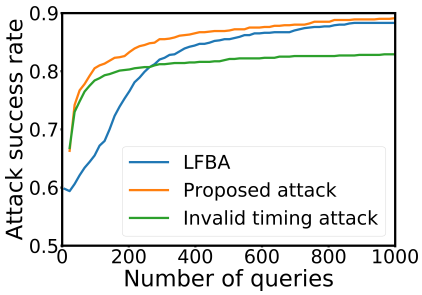
This subsection describes the experiment to verify the difficulty in finding images of deep depth. This difficulty supports presumption $S(t) = t$ as mentioned in Section 5.2. In this experiment, we examined the distribution of depth variations caused by adding random small perturbations to original images. We conduct this experiment to show that perturbations shorten the depths for almost all original images regardless of the original depths of the images without perturbations. Below, we describe the procedures and the results of this experiment.

The experiment procedure is as follows. First, to each of the 10,000 images in the test dataset, 10,000 random perturbations are generated and added. Each perturbation has a value of ± 0.01 per pixel with the probability of 50% for each sign. Next, depths of the images with and without perturbations are recorded. Then, variations in depth are calculated and aggregated to each bin with the width of 100 by depth of the images without perturbations. Finally, the distributions are expressed as a boxplot.

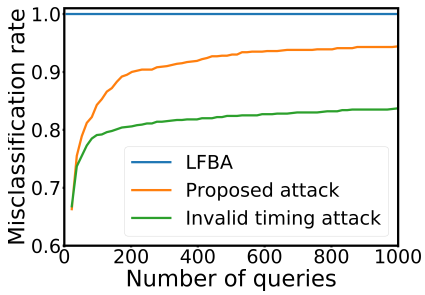
Figure 1 shows the results of this experiment. The original depth is represented on the horizontal axis, and the variation in depth when perturbations are added is represented on the vertical axis. Regardless of the original depths, the variations in depth tend to be negative. This shows the difficulty in finding perturbations that make depths deep. The cause of this difficulty can be the smallness of the subspaces that comprises images of deep depth in the space expressing images, as is explained in Section 5.2. Hereinafter, considering this difficulty, $S(t) = t$ is assumed because the optimization in this direction is easier than that in the opposite direction.

6.2 Threat Evaluation of Proposed Attack

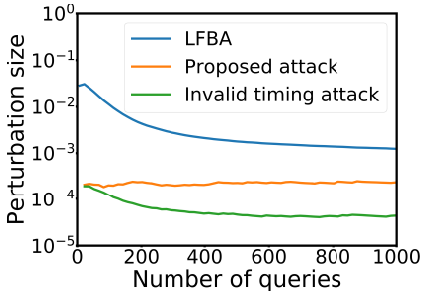
This section describes the experiment to evaluate the threat of the proposed attack. In this experiment, we compare the proposed attack to LFBA [5], a state-of-the-art algorithm attack, to show the superiority of the proposed attack to algorithm attacks. Additionally, we compare the proposed attack to an invalid timing attack. We define an invalid timing attack as an attack that uses the same algorithm as the proposed attack but substitutes a constant value of 1 for t in Eq. (1) instead of the valid timing.



(a) Comparison of attack success rates. The number of queries and the attack success rate are represented on the horizontal and vertical axes. Proposed attack exhibits higher success rates than invalid timing attack or LFBA.



(b) Comparison of misclassification rates. The number of queries and misclassification rate are represented on the horizontal and vertical axes. LFBA, proposed attack, and invalid timing attack exhibit high rates in this order.



(c) Comparison of perturbation sizes. The number of queries and average perturbation size of misclassified images are represented on the horizontal and vertical axes. LFBA exhibits the worst size of the three attacks.

Fig. 2 Comparison of three attacks: the proposed attack, invalid timing attack and LFBA.

This invalidates the optimization of the timing. The comparison of these two attacks is expected to indicate that the valid timing critically contributes to the superiority of the proposed attack.

In this experiment, we measure four metrics: the misclassification rate, average perturbation size, attack success rate, and Simpson coefficient between two sets of original images successfully attacked by the proposed attack and by LFBA. The definitions and the meanings of the first two metrics are mentioned in Section 2.2. The attack success rate is defined as the ratio of misclassified images with MSEs less than 0.001. This perturbation size is sufficiently small for human eyes to overlook according to the literature [5]. The Simpson coefficient is an indicator measuring the overlap of two sets, say A and B, defined as $|A \cap B| / \min(|A|, |B|)$. We use this indicator to quantitatively evaluate differences in characteristics between the proposed attack and LFBA.

Among these metrics, we mainly compare the attack success rate. In particular, if the proposed attack has a higher success rate than the invalid timing attack and LFBA, we judge that the timing attack is a threat at least to the victim environment we prepared. In such a case, using the timing weakens the effect of the countermeasure that conceals the prediction models and the confidence values. We use the remaining metrics to analyze the results in detail.

The procedure for this experiment is described below. First, we extract 1,000 original images correctly classified without attacks from the test MNIST dataset, which constitutes a victim dataset to be attacked. Second, we record the results of the three attacks for each image in the victim dataset. The procedure in each attack is as follows. To begin with, for each image, we execute the attack program that repeats queries. After a certain number of queries, the attack program updates the solution once considering the responses. At this time, three values are recorded: the Boolean value if a misclassification occurred, the perturbation size, and the number of queries so far. The attack program iterates this procedure comprising a certain number of queries and an update. When the attack completes, we aggregate these records per query and calculate three metrics for each query: the misclassification rate, the average perturbation size of the misclassified images, and the attack success rate. Note that we aggregate the records every 15 queries. This query interval is the largest number of queries required for an iteration of all the attacks. Third, the Simpson coefficient is similarly calculated depending on these records of the proposed attack and LFBA.

The remainder of this subsection describes the results and the interpretation of the data in the order of the attack success rate, misclassification rate, perturbation size, and Simpson coefficient. First, **Fig. 2** (a) shows the attack success rate. The number of queries and the attack success rate are represented on the horizontal and vertical axes, respectively. The proposed attack exhibits higher success rates than those for the invalid timing attack and LFBA. This means that by using the timing, the proposed attack decreases the effectiveness of the countermeasure, which conceals the prediction model and the confidence values.

Second, Fig. 2 (b) shows the misclassification rate. The number of queries and the misclassification rate are represented on the horizontal and vertical axes, respectively. LFBA, the proposed attack, and the invalid timing attack exhibit high misclassification rates in this order. Particularly, the misclassification rate is higher for the proposed attack than for the invalid timing attack. This indicates that optimizing the timing increases the probability of misclassification and contributes to high attack success rates.

Third, Fig. 2 (c) shows the average perturbation size. The number of queries and the average perturbation size of the misclassified images are represented on the horizontal and vertical axes, respectively. The perturbation size remains much order of magnitude smaller in the proposed attack and the invalid timing attack than in LFBA. The difference between the proposed attack and the invalid timing attack increased as the number of queries increased, and the perturbation size of the former became larger than that for the latter. This means that optimizing the timing negatively affects the perturbation size. However, this causes only a slight deterioration in the success rate, and the increase in misclassification rate compensates for the negative effect. This effect results from the design of the objective function. Before misclassification, the function forces the optimization to reduce only the timing and to ignore the perturbation size in accordance with Eq. (1).

Fourth, **Fig. 3** shows the Simpson coefficient. The number of queries is represented on the horizontal axis. The Simpson coeffi-

cient, which indicates the degree of overlap of the sets of images successfully attacked by the proposed attack and LFBA, is represented on the vertical axis. The coefficient is 1 if and only if (iff) one set includes the other; 0 iff the two sets are exclusive; and in the middle if they partially overlap. Overall, as the figure shows, the Simpson coefficient remains at nearly 1 during the attacks. In conjunction with the fact that the success rate is higher in the proposed attack than in LFBA, this means that the set successfully attacked by the proposed attack includes almost all of the set that can be attacked by LFBA. This suggests that there are only a few incorrigible images that can be attacked by LFBA but not the proposed attack.

6.3 Detailed Analysis of the Proposed Attack

This subsection describes the analysis to investigate more properties of the proposed attack in more detail. In this analysis, we investigate dependence of the attack success rate on the depth of images. The result of this analysis will show characteristics of the proposed attack and LFBA. Additionally, we investigate dependence of the misclassification rate and perturbation size on the depth. Analyzing these quantites will clarify what causes the depth dependence of the attack success rate.

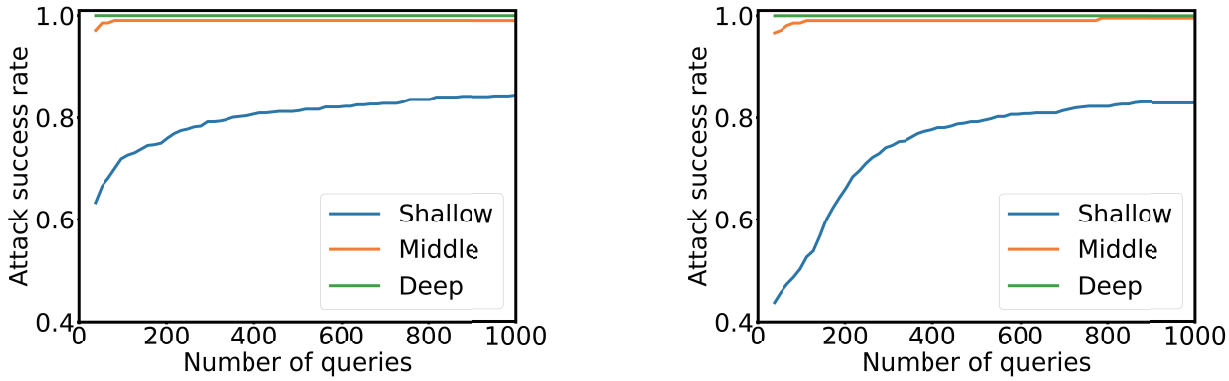
Fig. 3 Simpson coefficient, which indicates the degree of overlap of image sets successfully attacked by the proposed attack and LFBA. The number of queries and Simpson coefficient are represented on the horizontal and vertical axes. Overall, the Simpson coefficient remained at almost 1 during the attacks. In addition to the results in Fig. 2, this means that the set of the proposed attack includes the set of LFBA.

In this analysis, we measure three metrics per depth: the misclassification rate, average perturbation size, and attack success rate. The definitions and meanings of these metrics are the same as in Section 6.2. To investigate the depth dependence of these metrics, we divide depths in three ranges: shallow, middle and deep. These ranges divide $[d_{\min}, d_{\max}]$ into three equal parts, where $d_{\min} = 1{,}102$ is the minimum depth for the original images used in Section 6.2, whereas $d_{\max} = 2{,}993$ is the maximum one. To measure these metrics, we use the experimental results of the proposed attack and LFBA in Section 6.2.

The remainder of this subsection describes the results and the interpretation of the analysis. First, **Fig. 4** shows the attack success rate per depth for the proposed attack and LFBA. The number of queries and the attack success rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the success rates for the shallow, middle, and deep depths, respectively. Both the proposed attack and LFBA exhibit the lowest success rates for the shallow depth and highest rates for the deep depth. This result shows that depth shallowness decreases the success rate.

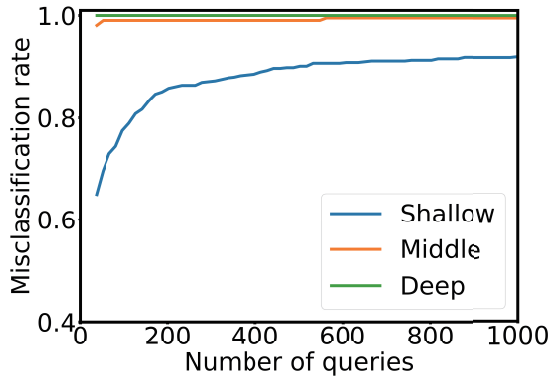
To clarify the cause of the depth dependence of the success rate, we analyze the misclassification rate and perturbation size per depth. **Figure 5** shows the values of these metrics for the proposed attack and Fig. 5 (a) shows the misclassification rate per depth. The number of queries and the misclassification rate are represented on the horizontal and vertical axes, respectively. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths, respectively. These curves show that the misclassification rate decreases when the depths of images are shallow. Additionally, the misclassification rate decreases by a comparable amount to the success rate.

Figure 5 (b) shows the average perturbation size per depth for the proposed attack. The number of queries and the misclassification rate are represented on the horizontal and vertical axes, respectively. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths, respectively. These curves show that the average perturbation size increases when the depths of images are shallow. However, even the perturbation sizes for the shallow depth are much less than 0.001. This means

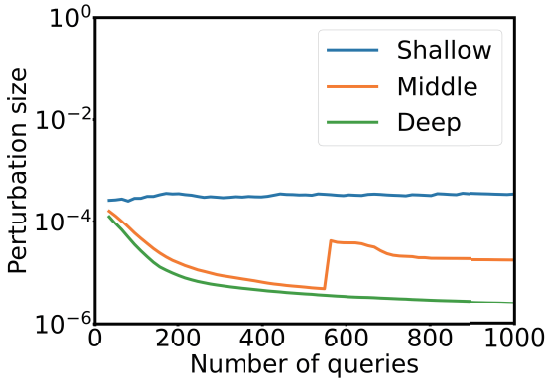


(a) Proposed attack. (b) LFBA.

Fig. 4 Attack success rate per depth. The number of queries and the attack success rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the success rates for the shallow, middle, and deep depths. Both the proposed attack and LFBA exhibit the lowest success rates for the shallow depth and highest rates for the deep depth.

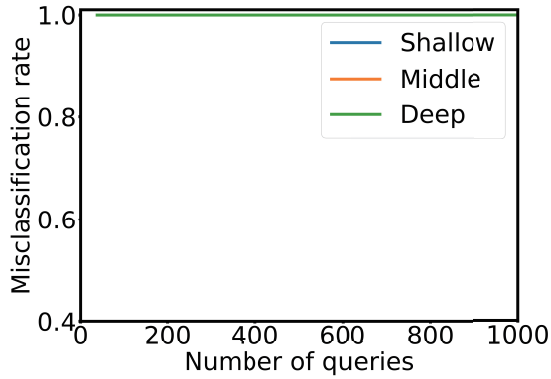


(a) Misclassification rate per depth.The number of queries and the misclassification rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths. These curves show that the misclassification rate decreases when the depths of images are shallow.

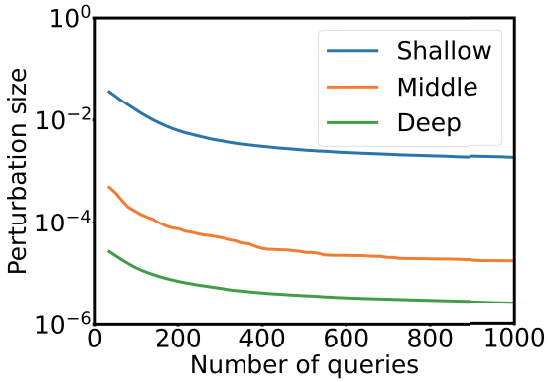


(b) Average perturbation size per depth. The number of queries and the misclassification rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths. Even the perturbation sizes for the shallow depth are much less than 0.001.

Fig. 5 Misclassification rate and perturbation size per depth in the proposed attack case.



(a) Misclassification rate per depth. The number of queries and the misclassification rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths. These curves show that the misclassification rate is 100% independent of the depth.



(b) Average perturbation size per depth. The number of queries and the misclassification rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths. These curves show that the average perturbation size increases when the depths of images are shallow.

Fig. 6 Misclassification rate and perturbation size per depth in LFBA case.

that the perturbation size increase hardly affects the success rate for the proposed attack.

From the above results in Fig. 5, we can conclude that the proposed attack achieves a low success rate for images of shallow depth mainly due to the misclassification rate decrease for such images. This conclusion is reasonable considering the large volume of subspaces that comprise images of shallow depth in the image-expressing space. Due to this largeness, the original images of shallow depth are probablistically far away from their classification boundaries. This can prevent misclassification from occurring and then decrease the success rate.

For LFBA, **Fig. 6** shows the misclassification rate and perturbation size per depth. Figure 6 (a) shows the misclassification rate per depth. The number of queries and the misclassification rate are represented on the horizontal and vertical axes, respectively. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths, respectively. These curves show that the misclassification rates are 100% independent of the depth. This result is reasonable because LFBA begins attacks with images far away from the original images. This approach guarantees the misclassification will occur at the cost of the perturbation size.

Figure 6 (b) shows the average perturbation size per depth for LFBA. The number of queries and the misclassification rate are represented on the horizontal and vertical axes. The blue, orange, and green curves represent the results for the shallow, middle, and deep depths, respectively. These curves show that the average perturbation size increases when the depths of images are shallow. Additionally, the value of this metric for shallow depth drastically exceeds 0.001. This means the perturbation size increase affects the success rate in LFBA case unlike in the proposed attack case.

From the above results in Fig. 6, we can conclude that the LFBA exhibits a low success rate for images of shallow depth mainly due to the perturbation size increase for such images. This conclusion is reasonable considering the large area of classifica-

tion boundaries around images of shallow depth in the image-expressing space. On such large classification boundaries, the perturbation size becomes difficult to minimize. This can increase the average perturbation size, and then decrease the success rate.

7. Discussion

In this section, we discuss risky situations involved with the timing attack, limitations of the proposed attack, and possible directions for countermeasures.

7.1 Risky Situations and Limitations of Timing Attack

In this subsection, we discuss situations where the risk of the timing attack increases, along with limitations of the proposed attack. This discussion considers two viewpoints: the viewpoint of the manner for mutual communication between the victim and the attacker, and that of the victim environment. First, the manner of communications includes local or remote communications. Here, remote communications means the case where the victim environment and the attacker program are on different machines and communicate through a network. Local communications means the case where they are on the same machine and communicate locally. Between these cases, local communications can be more risky due to the absence of measurement errors resulting from communications delay. For example, local communications are actualized when the attacker has the prediction model readily available in the case of an edge AI.

Second, the victim environment comprises a machine, an OS, a framework, a prediction model, and a dataset. Among these, a case in which the machine or OS cause fewer interruptions can increase the risk due to a decrease in measurement errors of the timing. Such a situation can occur when the machine has many CPU cores and the OS executes few processes simultaneously. This also means the proposed attack is invalidated when the victim environment is busy. However, the attacker may efficiently circumvent this difficulty by outlier detection in the timing because interruptions can cause jumps in the timing.

Regarding the framework, the programming language used in the framework can affect the risk. For example, a framework based on C++ may be easier to attack than a framework based on Python due to the absence of interpreters, which can generate superfluous measurement errors in the timing. This also means that the proposed attack may fail more often when the victim environment uses a framework based on a language executed on a complex software stack. This limitation can be critical unlike the interruptions mentioned above because measurement errors in the timing will be continuous in this case. Thus, the attacker must iterate the timing measurement over and over for the same perturbed image. This increases the number of queries.

As for the prediction model, a large number of decision trees can increase the risk because this amplifies depth variations of the random forest, making variations in the timing more prominent. Additionally, in regard to the dataset, the diversity of the dataset can be relevant because a high degree of diversity will require a large number of conditional branches for accurate prediction. This also makes variations in the timing more salient.

7.2 Possible Directions for Countermeasures

There are two possible directions for countermeasures for the timing attack in principle: taking measures subsequent to or prior to training. The first direction includes two strategies that are common in the context of timing attacks on cryptographic systems: making the processing time constant, and limiting the attacker ability to control queries. For example, in the former case, there is a method that waits to respond until a predetermined fixed time elapses. In the latter case, there is a method that adds random perturbation to input data. The former example can prevent timing attacks because attackers always observe the fixed time as processing time and cannot optimize adversarial examples based on the timing. This example prevents the attacks without deteriorating the prediction accuracy but increases the average processing time because the fixed time must be longer than the processing time required for the maximum depth. The long processing time is inconvenient for users of machine-learning-based systems. In the latter case, the average processing time does not deteriorate, but the prediction accuracy decreases due to the perturbation.

The second direction can lead to a method that restricts the depth of random forests. This method is reasonable because the analysis in Section 6.3 reveals that the shallow depth of images decreases the attack success rate of the proposed attack and LFBA. This result indicates that the depth restriction can reduce the attack success rate of timing attacks at no cost of that rate of other adversarial example attacks. This indication comes from the fact that the proposed attack and LFBA uses opposite strategies for each other: the proposed attack begins with perturbed images near original images and then moves the images further away from the original ones; LFBA begins with perturbed images far from original images and then moves the images closer to the original ones. To the best of our knowledge, either of these strategies are adopted by exiting adversarial example attacks. Thus, the results in Section 6.3 indicate that restricting the depth of random forests is effective for every adversarial example attack.

We can readily implement the depth restriction for the present prediction model `RandomForestClassifier` because this prediction model has a parameter for such restriction, `max_depth`. This parameter restricts the depth of each tree in random forests to the parameter value. Thus, when the number of decision trees in a random forest is `n_estimators`, `max_depth` can restrict the random forest depth to `n_estimators` $\times$ `max_depth`. We can roughly estimate the effect of this countermeasure from the result in Fig. 4. In this figure, the success rate for the shallow depth is nearly 20% lower than those for the middle and deep depths. Now that the upper bound depth of the shallow depth is 1,732, restricting random forest depth to 1,732 will decrease the success rate by roughly 20%. We can implement this restriction by setting `max_depth` to 17 because `n_estimators` = 100 in this paper.

However, this restriction can conflict with prediction accuracy. To estimate the impact of the `max_depth` restriction on the accuracy, we train the same random forest as in Section 6 with different `max_depth` (**Fig. 7**). In Fig. 7, `max_depth` and the prediction accuracy are represented on the horizontal and vertical axes, respectively. The blue curve shows that the accuracy critically falls

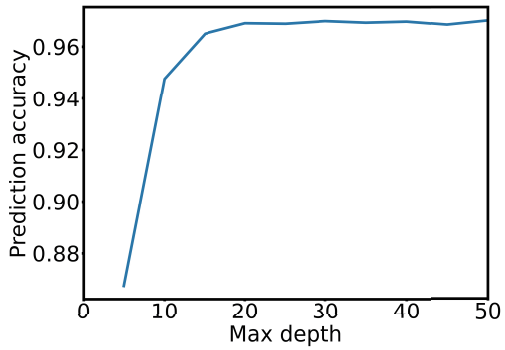


Fig. 7 Depth dependence of the prediction accuracy of the random forest. `max_depth`, and the prediction accuracy are represented on the horizontal and vertical axes. The blue curve shows that the accuracy critically falls below `max_depth` = 20.

down below `max_depth` = 20. Combined with the estimation in the preceding paragraph, this result shows that the `max_depth` restriction required to decrease the success rate by 20% is on the edge of this accuracy plunge. This suggests that we should carefully tune `max_depth` to achieve a satisfactorily low attack success rate and high prediction accuracy. Additionally, when we simultaneously need a higher prediction accuracy and a lower attack success rate, we should take other countermeasures. Investigating such countermeasures will be a topic for future work.

7.3 Generality of Proposed Attack

We propose and evaluate the attack on random forests [22] in this paper, but the proposed attack is expected to be effective for other tree-based methods such as decision tree and XGBoost [21]. The proposed attack exploits the common characteristic in tree-based methods: drastic changes in the timing indicate drastic changes in the results of the invoked branches. Attackers can ruin prediction results by changing the results of the branches based on the timing. We select random forests in this paper from the perspectives of prediction accuracy and reliability of experiments. With decision tree, accurate models targeted by attackers are difficult to train. With XGBoost, accurate models are possible to train, but reliable experiments are slightly difficult to conduct. Since XGBoost has many hyperparameters, experimental results might cause a suspicion that attacks succeed due to specific hyperparameters. For this reason, we select random forests that can achieve high prediction accuracy and have fewer hyperparameters than XGBoost.

8. Conclusion

In this paper, to prove the threat of implementation attacks to random forests, we presented a novel timing attack to generate adversarial examples, and evaluated its threat experimentally. The proposed attack searches for a misclassified image in the vicinity of the normal one by optimizing the timing using the covariance matrix adaptation evolution strategy (CMA-ES), a common evolution strategy.

In the experiment, we compared the proposed timing attack with a state-of-the-art algorithm attack. The results show that the attack success rate of the former exceeded that of the latter in the black-box setting where attackers can use only prediction labels and the timing. This suggests the threat of implementation attacks to random forests in the sense that such attacks can make it less effective to conceal the prediction model and the confidence values as a countermeasure to adversarial example generation.

The detailed analysis of the experimental results indicates that restricting the tree depth of random forests may mitigate the proposed attack but decrease prediction accuracy of random forests. This clarifies the need for more sophisticated countermeasures, which will be a topic for future work.

References

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R.: Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199 (2013).
[2] Brendel, W., Rauber, J. and Bethge, M.: Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models, *Proc. 6th International Conference on Learning Representations* (2018).
[3] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J. and Hsieh, C.-J.: ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models, *Proc. 10th ACM Workshop on Artificial Intelligence and Security*, pp.15–26 (2017).
[4] Goodfellow, I.J., Shlens, J. and Szegedy, C.: Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572 (2014).
[5] Guo, C., Frank, J.S. and Weinberger, K.Q.: Low Frequency Adversarial Perturbation, *Proc. 35th Conference on Uncertainty in Artificial Intelligence* (2019).
[6] Guo, C., Gardner, J., You, Y., Wilson, A.G. and Weinberger, K.: Simple Black-box Adversarial Attacks, *Proc. 36th International Conference on Machine Learning*, pp.2484–2493 (2019).
[7] Kantchelian, A., Tygar, J.D. and Joseph, A.: Evasion and hardening of tree ensemble classifiers, *Proc. 33rd International Conference on Machine Learning*, pp.2387–2396 (2016).
[8] Tramèr, F., Zhang, F., Juels, A., Reiter, M.K. and Ristenpart, T.: Stealing Machine Learning Models via Prediction APIs, *Proc. 25th USENIX Security Symposium*, pp.601–618 (2016).
[9] Fredrikson, M., Jha, S. and Ristenpart, T.: Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures, *Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp.1322–1333 (2015).
[10] Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C. and Li, B.: Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning, *Proc. 2018 IEEE Symposium on Security and Privacy*, pp.19–35 (2018).
[11] Shokri, R., Stronati, M., Song, C. and Shmatikov, V.: Membership Inference Attacks Against Machine Learning Models, *Proc. 2017 IEEE Symposium on Security and Privacy*, pp.3–18 (2017).
[12] Wang, B. and Gong, N.Z.: Stealing Hyperparameters in Machine Learning, *Proc. 2018 IEEE Symposium on Security and Privacy*, pp.36–52 (2018).
[13] Batina, L., Bhasin, S., Jap, D. and Picek, S.: CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel, *Proc. 28th USENIX Security Symposium*, pp.515–532 (2019).
[14] Breier, J., Hou, X., Jap, D., Ma, L., Bhasin, S. and Liu, Y.: Practical Fault Attack on Deep Neural Networks, *Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp.2204–2206 (2018).
[15] Duddu, V., Samanta, D., Rao, D.V. and Balas, V.E.: Stealing neural networks via timing side channels, arXiv preprint arXiv:1812.11720 (2018).
[16] Hong, S., Davinroy, M., Kaya, Y., Locke, S.N., Rackow, I., Kulda, K., Dachman-Soled, D. and Dumitraş, T.: Security analysis of deep neural networks operating in the presence of cache side-channel attacks, arXiv preprint arXiv:1810.03487 (2018).
[17] Liu, Y., Wei, L., Luo, B. and Xu, Q.: Fault injection attack on deep neural network, *Proc. 2017 IEEE/ACM International Conference on Computer-Aided Design*, pp.131–138 (2017).
[18] Wei, L., Luo, B., Li, Y., Liu, Y. and Xu, Q.: I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators, *Proc. 34th Annual Computer Security Applications Conference*, pp.393–406 (2018).
[19] Yan, M., Fletcher, C.W. and Torrellas, J.: Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures, arXiv preprint arXiv:1808.04761 (2018).
[20] Oracle: An Introduction To Building a Classification Model Using

Random Forests In Python, available from ⟨https://blogs.oracle.com/datascience/an-introduction-to-building-a-classification-model-using-random-forests-in-python⟩.
[21] Chen, T. and Guestrin, C.: XGBoost: A Scalable Tree Boosting System, *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.785–794 (2016).
[22] Breiman, L.: Random forests, *Machine learning*, Vol.45, No.1, pp.5–32 (2001).
[23] Hansen, N. and Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies, *Evolutionary Computation*, Vol.9, No.2, pp.159–195 (2001).
[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E.: Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, Vol.12, pp.2825–2830 (2011).

Editor's Recommendation

In this paper, a timing attack on implementation of the Random Forest algorithm is proposed. The attacker's assumptions are reasonable, and the effectiveness of the proposed method is shown by evaluation using the MNIST data set. Since the IWSEC2020 program committee highly evaluated the novelty and usefulness of this paper, it was selected as a recommended paper.

(Program co-chairs of IWSEC 2020:
Kazumaro Aoki, Akira Kanaoka)

Yuichiro Dan received his B.S. degree in physics from Osaka University, Japan, in 2014. He also received his M.S. degree in physics from Kyoto University, Japan, in 2017. In 2017, he joined NTT Secure Platform Laboratories, where he is currently researching hardware security.

Toshiki Shibahara is currently a researcher at NTT Social Informatics Laboratories, Tokyo, Japan. He received his B.E. degree in engineering and M.E. degree in information science and technology from The University of Tokyo, Japan in 2012 and 2014. He also received his Ph.D. degree in information science and technology from Osaka University, Osaka, Japan in 2020. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2014, he has been engaged in research on cyber security and machine learning.

Junko Takahashi received her B.S. and M.S. degrees in physics from Waseda University, Japan, in 2004 and 2006, respectively, and her Ph.D. degree in engineering from the University of Electro-Communications, Japan, in 2012. She joined the Information Sharing Platform Laboratories, Nippon Telegraph and Telephone Corporation (NTT) in 2006. Currently, she is a researcher with the NTT Social Informatics Laboratories. She has studied hardware security such as side-channel analysis and automotive security. Recently, she has been studying security of optical circuit. She is a member of the Information and Communication Engineers (IEICE) and Information Processing Society of Japan (IPSJ). She has been a member of technical committee on hardware security in the IEICE and special interest group on system architecture in the the IPSJ. She was awarded the 2008 symposium on cryptography and information security (SCIS) paper prize and her paper in JIP Vol.25 was selected as a specially selected paper in the IPSJ in 2017.