PAPER

# Advanced Ensemble Adversarial Example on Unknown Deep Neural Network Classifiers

**Hyun KWON**†, **Yongchul KIM**††, ***Nonmembers***, **Ki-Woong PARK**†††a), ***Member***, **Hyunsoo YOON**†, ***Nonmember***, ***and*** **Daeseon CHOI**††††b), ***Member***

**SUMMARY** Deep neural networks (DNNs) are widely used in many applications such as image, voice, and pattern recognition. However, it has recently been shown that a DNN can be vulnerable to a small distortion in images that humans cannot distinguish. This type of attack is known as an adversarial example and is a significant threat to deep learning systems. The unknown-target-oriented generalized adversarial example that can deceive most DNN classifiers is even more threatening. We propose a generalized adversarial example attack method that can effectively attack unknown classifiers by using a hierarchical ensemble method. Our proposed scheme creates advanced ensemble adversarial examples to achieve reasonable attack success rates for unknown classifiers. Our experiment results show that the proposed method can achieve attack success rates for an unknown classifier of up to 9.25% and 18.94% higher on MNIST data and 4.1% and 13% higher on CIFAR10 data compared with the previous ensemble method and the conventional baseline method, respectively.
***key words:*** *adversarial example, neural networks, ensemble adversarial example, machine learning*

## 1. Introduction

Machine learning techniques have recently begun receiving significant attention as a potential primary technology in the field of recognition. This is especially true for deep neural networks (DNNs) [1], which have shown good performance in many applications such as image recognition [2], voice recognition [3], pattern recognition [4], autonomous vehicles [5], and natural language processing [6]. Unfortunately, DNN systems have been introduced that have vulnerabilities to an adversarial example attack [7], [8]. The attack causes misclassification by the DNN system by adding only a little noise to the original image. The main goal of this attack is to maximize the misclassification while minimizing the distortion so that a human cannot recognize the difference. This attack is a serious threat to DNN classifiers. For example, when a left-turn road sign is modified by the adversarial example attack, an autonomous vehicle with DNN might recognize the sign as a right-turn sign, while human drivers cannot detect any problem with the road sign.

There are two types of attacks: the white box attack [7], [9], [10] and the black box attack. The white box attack is used when the attacker has detailed information about the target model, i.e., model architecture, parameters, and probabilities of output classifications. Hence, the success rate of a white box attack is almost 100%. Some articles that have been recently published [11], [12] show that defending a DNN from a white box attack is extremely difficult. However, it is unrealistic that an attacker would know every piece of information about the target model. Even if the attacker has an autonomous vehicle, a separate attack such as tampering is still needed to obtain the probabilities of each class for recognizing traffic signs. It is even more difficult if the target model works on servers such as in the cloud. For this reason, the black box attack is more interesting and is currently receiving more attention.

The black box attack only assumes that attackers can query target models. For example, when an autonomous vehicle is a target model, an attacker can query the autonomous vehicle by using distorted images to see the responses from the target model. A substitute-model attack proposed in [13] is a well-known black box attack example. In this scheme, an attacker can create a substitute network that is similar to the target model by repeating the query process. Once a substitute network is created, the attacker can perform a white box attack. The authors of [13] created a substitute network against Amazon and Google services and showed that their success rate on MNIST [14] data is 81.2%. Therefore, it is clear that the query process is a key task in the black box attack.

However, a query process is not applicable to the case of an unknown classifier. This means that it is neither a white box nor a black box. There are three cases that are considered unknown classifier scenarios: 1) Query-limited model. In a civilian environment, repeated queries of a system might be limited because many systems block such queries if they come from a single source IP. 2) Unreleased model. When a target model has not yet been released, it is impossible to query. However, there is a need to create effective adversarial examples for attacking even upcoming models. 3) Minor model. Creating a substitute network requires more query processing and time. However, there exist trivial models that do not require time to process queries to attack an unknown classifier. In such cases, attacks that do

Manuscript received February 25, 2018.
Manuscript revised May 11, 2018.
Manuscript publicized July 6, 2018.
†The authors are with School of Computing, Korea Advanced Institute of Science and Technology, Korea.
††The author is with Department of Electrical Engineering, Korea Military Academy, Korea.
†††The author is with Department of Computer and Information Security, Sejong University, Korea.
††††The author is with Department of Medical Information, Kongju National University, Korea.
a) E-mail: woongbak@sejong.ac.kr (Co-corresponding author)
b) E-mail: sunchoi@kongju.ac.kr (Co-corresponding author)
DOI: 10.1587/transinf.2018EDP7073

not require query processing are needed.

The work in [7], [15] introduces the concept of transferability, based on the idea that an adversarial example modified for a single target model is effective for other models that classify the same kind of data. Thus, it seems that the conventional adversarial example will affect an unknown classifier. However, the attack success rate is very low in those cases. The latest study [16] proposed an ensemble adversarial example method that employed multiple target models to attack the other models; it achieved a higher attack success rate. In this study, we propose an advanced ensemble adversarial attack, which achieves a better attack success rate than the previous method. Our contributions are as follows:

- To increase general transferability, we propose a hierarchical method that can achieve high attack success rates for unknown classifiers by step-by-step generation and an adversarial training method.
- For learning multiple target models, scalability needs to be considered, and in this study, the performance of the proposed method and the ensemble method are analyzed in terms of their scalability.
- We show the effectiveness of our proposed scheme through an experiment with unknown classifiers, using the MNIST and CIFAR10 [17] datasets.

The remainder of this paper is structured as follows: Section 2 reviews the background and related work. After the problem definition is addressed in Sect. 3, our proposed adversarial example attack is presented in Sect. 4. Section 5 briefly explains the setup of each experiment test model. The experiment results are shown in Sect. 6, and a discussion of the proposed system is presented in Sect. 7. Finally, conclusions are drawn in Sect. 8.

## 2. Background and Related Work

The study of adversarial examples was introduced by Szegedy et al. [7] in 2014. The main goal of using an adversarial example is to induce the DNN to make a mistake by adding a small amount of noise to the original image such that humans cannot tell the difference between the original and the distorted image.

The basic method for generating adversarial examples is described in Sect. 2.1. Adversarial examples can be categorized in four ways: recognition of an adversarial example, target model information, distance measure, and generation method, as described in Sects. 2.2–2.5, which follow.

### 2.1 Adversarial Example Generation

The basic architecture that generates an adversarial example consists of two elements: a target model and a transformer. The transformer takes the original sample, $x$, and original class, $y$, as input data. The transformer then creates as output a transformed example $x^* = x + w$, with noise value $w$ added to the original sample $x$. The transformed example $x^*$ is given as input data to the target model. The target model then provides the transformer with the class probability results for the transformed example. The transformer updates the noise values $w$ in the transformed example $x^* = x + w$ so that the other class probabilities are higher than the original class probabilities while minimizing the distortion distances between $x^*$ and $x$.

### 2.2 Categorization by Recognition on Adversarial Example

According to the class that the target model recognizes from the adversarial examples, we can divide the adversarial examples into two subcategories: targeted adversarial examples and untargeted adversarial examples. A targeted adversarial example is an adversarial example that causes the target model to recognize the adversarial image as a particular intended class. It can be expressed mathematically as follows.

Given a target model and original sample $x \in X$, the problem is an optimization problem that generates a targeted adversarial example $x^*$,

$$x^* : \underset{x^*}{\operatorname{argmin}}\ L(x, x^*) \ \text{ s.t. } \ f(x^*) = y^* \tag{1}$$

where $L(\cdot)$ is a distance measure between the original sample $x$ and the transformed example $x^*$, and $y^*$ is a particular intended class. $f(\cdot)$ is an operation function that provides class results for the input values of the target model.

An untargeted adversarial example is an adversarial example that causes the target model to recognize the adversarial image as a class other than the original class. It can be expressed mathematically as follows.

Given a target model and original sample $x \in X$, the problem is an optimization problem that generates an untargeted adversarial example $x^*$,

$$x^* : \underset{x^*}{\operatorname{argmin}}\ L(x, x^*) \ \text{ s.t. } \ f(x^*) \neq y \tag{2}$$

where $y \in Y$ is the original class.

Because the untargeted adversarial example has the advantages of less distortion from the original image and shorter learning time than the targeted adversarial example, we focus on the untargeted adversarial example scenario in this study.

### 2.3 Categorization by Target Model Information

Attacks that generate adversarial examples can also be divided into two types according to how much information about the target is required for the attack: white box attacks and black box attacks. As mentioned in the previous section, the black box attack is a more realistic approach because it only requires the response of an input; no additional information about the target is needed.

### 2.4 Categorization by Distance Measure

There are three ways to measure the distortion between the original sample and the adversarial example [9], [12]. The first distance measure, $L_0$, represents the sum of the number of all changed pixels:

$$\sum_{i=0}^{n} |x_i - x_i{}^*| \tag{3}$$

where $x_i$ is the original $i^{th}$ pixel and $x_i{}^*$ is an adversarial example $i^{th}$ pixel. The second distance measure, $L_2$, represents the standard Euclidean norm, as follows:

$$\sum_{i=0}^{n} \sqrt{(x_i - x_i{}^*)^2} \tag{4}$$

The third distance measure, $L_\infty$, is the maximum distance value between $x_i$ and $x_i{}^*$. Therefore, the smaller the three distance measures, the more similar the sample image is to the original sample from a human's perspective.

### 2.5 Categorization by Adversarial Example Generation Method

There are four typical attacks that generate adversarial examples. The first method is the fast-gradient sign method (FGSM) [15], which can find $x^*$ through $L_\infty$:

$$x^* = x + \epsilon \cdot \text{sign}(\triangledown loss_{F,t}(x)) \tag{5}$$

where $F$ is an object function and $t$ is a target class. In every iteration of the FGSM, the gradient is updated by $\epsilon$ from the original $x$, and $x^*$ is found through optimization. This method is simple and demonstrates good performance.

The second method is iterative FGSM (I-FGSM) [18], which is an updated version of FGSM. Instead of changing the amount $\epsilon$ in every step, a smaller amount, $\alpha$, is updated and eventually clipped by the same $\epsilon$:

$$x_i{}^* = x_{i-1}{}^* - \text{clip}_\epsilon(\alpha \cdot \text{sign}(\triangledown loss_{F,t}(x_{i-1}{}^*))) \tag{6}$$

This I-FGSM method provides better performance than the FGSM.

The third is the Deepfool method [10], which is an untargeted attack and uses the $L_2$ distance measure. This method generates an adversarial example that is more efficient than FGSM and as close as possible to the original image. To generate an adversarial example, the method involves the construction of a neural network and looks for $x^*$ using the linearization approximation method. However, because the neural network is not completely linear, we have to find the adversarial example through multiple iterations; i.e., it is a more complicated process than FGSM's.

The fourth method is the Carlini attack [9], which is the latest attack method and delivers better performance than the FGSM and I-FGSM methods. This method can achieve a 100% success rate even against the distillation structure [11], which was recently introduced in the literature. The key principle of this method involves the use of a different objective function:

$$D(x, x^*) + c \cdot f(x^*) \tag{7}$$

Instead of using the conventional objective function $D(x, x^*)$, this method proposes a way to find an appropriate binary $c$ value. In addition, it suggests a method to control the attack success rate even with some increased distortion by reflecting the confidence value as follows:

$$f(x^*) = \max(Z(x^*)_t - \max\{Z(x^*)_i : i \neq t\}, -k) \tag{8}$$

where $Z(\cdot)$ represents the pre-softmax classification result vector and $t$ is a target class.

In this study, we construct the model by applying the Carlini attack, which is the most powerful of the four methods, and use $L_2$ as the distance measure.

### 2.6 Transferability for Generalized Attack

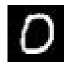
The concept of transferability was first introduced by Szegedy et al. [7]. An adversarial example created through an arbitrary target model that the attacker already knows can also attack unknown models. Using the MNIST dataset, transferability was shown when the models and learning data were different. If the MNIST learning data were different in the same model, the attack success rate was about 8% in a general case. Goodfellow et al. [15] first proposed an algorithm for universal perturbations. Moosavi Dezfooli et al. [19] have proposed an advanced algorithm for universal perturbation following the study of Goodfellow et al. [15]. This method is a generalized attack method on decision boundaries similar to the ensemble method that can fake multiple classifiers. By comparing generalized attacks on different networks using ImageNet [20], it was shown that the greater the number of images, the greater the fooling rate, and an attack success rate of approximately 50% was achieved. Liu et al. [16] recently proposed an ensemble-based approach to generate transferable adversarial examples. Additionally, Baluja and Fischer [21] generate adversarial examples against a target network or set of networks. They showed that the adversarial examples generated using ensemble-based approaches can successfully attack black box image classification systems.

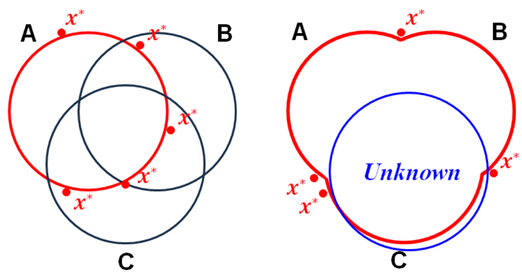
In these previous studies, however, there was not enough research conducted on scalability analysis. As the number of target models increases, a more advanced ensemble method with better performance is required. Therefore, this study proposes a hierarchical ensemble for scalability.

## 3. Problem Definition

Table 1 shows an adversarial example image that can be misclassified as target class "2" by the target model for the original sample "0". In Table 1, the score for the "2" class (9.98) is slightly higher than the score for the original class (9.96). This result shows that the process of creating an adversarial example continues until any other class score is slightly higher than the original class score. This is because the distance between the original sample and the adversarial example should be minimized. Therefore, when the target model

**Table 1** Classification of adversarial example

| Description | Original sample ("0") | Adversarial example ("2") |
|---|---|---|
| |  |  |
| Class score | [**37.4** -6.44 0.27 -10.88 -3.49 4.86 -1.50 -3.85 -1.03 7.68] | [ **9.96** -4.78 **9.98** -0.64 -5.94 -5.06 -6.39 -3.68 -2.47 -2.97] |




(a) A single target model (b) Multiple target models

**Fig. 1** An example of transferability for adversarial examples of single target model A and multiple target models A, B, and C.

is changed, there is a possibility that the adversarial example will be correctly classified by the changed target model as the original class because the score of the misclassified class is similar to that of the original class.

Figure 1 (a) shows an example of transferability for adversarial examples for a single target model A. Transferability is the property that an adversarial example for a target model can attack another target model. In Fig. 1 (a), A, B, and C are convolutional neural network models, and the red circle is the decision boundary of the target model A. If the sample lies within the red circle, the sample is correctly classified as the original class by the target model A. The adversarial example is generated around the red circle of the target model A because it is incorrectly classified as the wrong class while minimizing the distance from the original sample. In Fig. 1 (a), each red dot is an adversarial example $x^*$. Because these adversarial examples are aimed at model A, it is possible that they can be correctly classified by the other models, B and C. Therefore, to generate an adversarial example that is misclassified by multiple models, it should be outside the decision boundaries of those models.

Figure 1 (b) shows an example of transferability for ensemble adversarial examples of models A, B, and C. Ensemble adversarial examples can be misclassified by all three models because the ensemble adversarial examples generated are not included within the decision boundaries of any of the models. Therefore, the ensemble adversarial examples generated for multiple models may have higher attack success rates for an unknown classifier than the adversarial examples generated for a single target model. This is because the decision boundary of the unknown classifier is more likely to be included in the decision boundaries of multiple models than in that of a single model decision boundary.

Although the conventional ensemble method has a high attack success rate for an unknown classifier, it is difficult to generate an ensemble adversarial example when there is a large number of target models. Because the conventional ensemble method has too many conditions to be satisfied for misclassification by all the target models, the gradient descent will have a higher probability of becoming stuck in a local minimum during the process of generating ensemble adversarial examples. Therefore, it is possible that the adversarial examples generated for multiple target models will be correctly recognized by some of the target models. To avoid this problem, we used a hierarchical approach to consider the scalability of the target model. This method generates an ensemble adversarial example for two of target models (even though there are many target models) to reduce the probability of becoming stuck in a local minimum of the gradient descent. In addition, this method increases the probability of misclassification by all target models by using step-by-step generation and an adversarial training method.

## 4. Proposed Scheme

The proposed method generates an ensemble adversarial example by hierarchically creating and training adversarial examples.

Figure 2 shows an example of the generation of a hierarchical adversarial example. This method consists of two procedures: generation of an ensemble adversarial example and an adversarial training method. First, ensemble adversarial examples are generated for each pair of target models by hierarchically partitioning a number of pretrained target models. Second, in the adversarial training process [15], one of the two target models is trained on the ensemble adversarial examples generated above. These two procedures are repeated until there are only two target models remaining. When this point is reached, hierarchical adversarial examples are generated for the last two target models.

### 4.1 Step 1: Generation of Ensemble Adversarial Example for Two Target Models

Given two target models $D_a$ and $D_b$ by hierarchically partitioning a number of pretrained target models, the process for generating an ensemble adversarial example targeting the two models is as follows:

The transformer takes the original sample $x \in X$ and the original class $y \in Y$ as input values and generates a transformed example $x^*$ as an output value. Neither target $D_a$

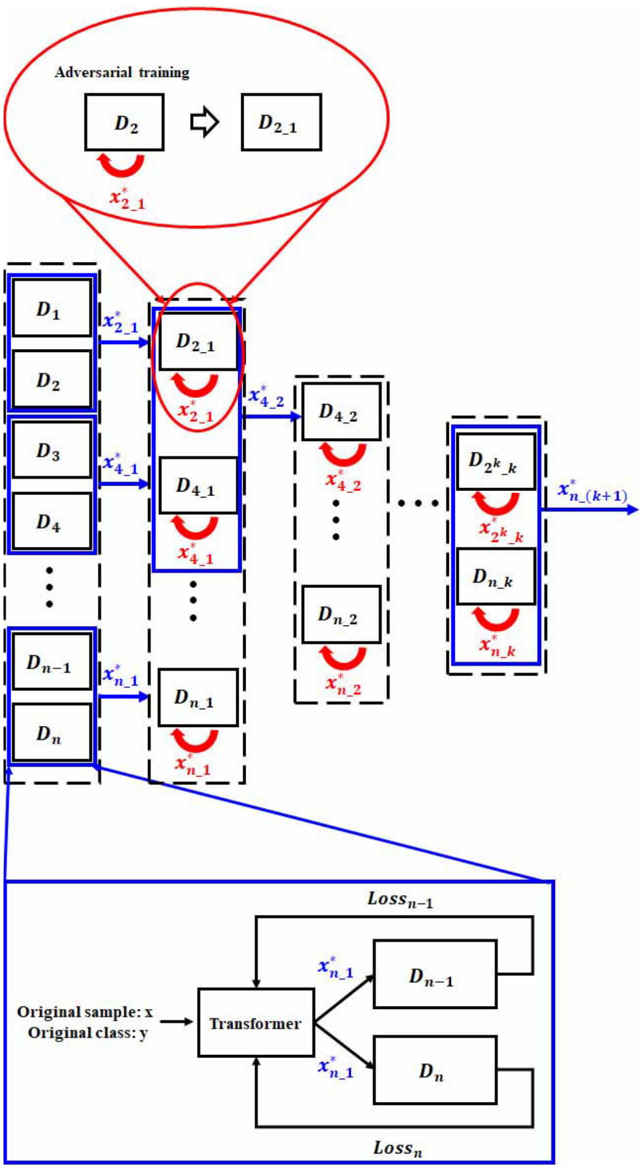



**Fig. 2** Example of the proposed architecture.

nor $D_b$ changes during the transformation; they take $x^*$ as input values and provide the respective classification results (losses) to the transformer. The goal of this architecture is that $x^*$ will be misclassified by both targets $D_a$ and $D_b$ with minimal distortion.

The attack type in the proposed method is that of an untargeted adversarial example. It can be expressed mathematically as follows: $f_a(\cdot)$ and $f_b(\cdot)$ are the operation functions of $D_a$ and $D_b$, respectively. Given the original input $x$ and pretrained target models $D_a$ and $D_b$, the problem is an optimization problem that generates an adversarial example $x^*$:

$$x^* : argmin_{x^*} L(x, x^*) \text{ s.t. } f_i(x^*) \neq y \ (i = a,\ b) \tag{9}$$

where $L(\cdot)$ is the distance measure between $x$ and $x^*$, and $y$ is the original class. $f_i(\cdot)$ is an operation function that provides the classification result for the input value by target model $i$.

**Algorithm 1** Ensemble adversarial example for the two target models

*Origianl sample*: $x$
*Original class*: $y$
*Number of iteration*: $r$
**procedure** GENERATION($x$, $y$, $r$)
  $w \leftarrow 0$
  $x^* \leftarrow 0$
  **for** r step **do**
    $x^* \leftarrow \frac{tanh(x+w)}{2}$
    $g_a(x^*) \leftarrow Z_a(x^*)_y - max\left\{Z_a(x^*)_j : j \neq y\right\}$
    $g_b(x^*) \leftarrow Z_b(x^*)_y - max\left\{Z_b(x^*)_j : j \neq y\right\}$
    $g(x^*) \leftarrow c_a \cdot g_a(x^*) + c_b \cdot g_b(x^*)$
    $temp \leftarrow \sqrt{(x^* - \frac{tanh(x)}{2})^2 + g(x^*)}$
    *Update w by minimizing the gradient of temp*
  **end for**
  *return* $x^*$
**end procedure**

The procedure to generate an ensemble adversarial example that targets two models has two steps. The first step is a pre-learning process for both targets $D_a$ and $D_b$ to correctly classify the original samples as their original classes:

$$f_i(x) = y \in Y,\ i = a, b \tag{10}$$

The second step is to generate ensemble example $x^*$ using a transformer on the original sample $x$ and original class $y$. In this study, the transformer architecture [9] was modified, and $x^*$ was defined as:

$$x^* = \frac{\tanh(x + w)}{2}$$

where $w$ is a modifier and is used when optimizing with a gradient. tanh is applied to soften the gradient. The loss of each target $D_a$ and $D_b$ for $x^*$ is provided to the transformer. The transformer then generates an ensemble adversarial example by calculating $loss_T$ and minimizing $loss_T$. $loss_T$ is defined as follows:

$$loss_T = loss_{\text{distortion}} + c_a \cdot loss_{f_a} + c_b \cdot loss_{f_b} \tag{11}$$

where $loss_{\text{distortion}}$ is the distortion of $x^*$, $loss_{f_i}$ is the classification loss of target model $i$, and $c_i$ is a weight parameter of target model $i$. $loss_{\text{distortion}}$ is the distance measure between $x$ and $x^*$:

$$loss_{\text{distortion}} = \sqrt{\left(x^* - \frac{\tanh(x)}{2}\right)^2}$$

To satisfy $loss_{f_i} \neq y$, $loss_{f_i}$ should be minimized:

$$loss_{f_i} = g_i(x^*),\ i = a, b \tag{12}$$

where $g_i(k) = Z_i(k)_{\text{org}} - max\left\{Z_i(k)_j \ : \ j \neq y\right\}$ and $y$ is the original class. $f_i(x^*)$ predicts the probability of the original class to be lower than the probability of the other class by optimally minimizing $loss_{f_i}$. The process of the proposed method is given in detail as Algorithm 1.

### 4.2 Step 2: Adversarial Training

One of the two target models in the pair trains on the ensemble adversarial examples generated in Step 1. The number of training epochs is ten. When the target model trains on the adversarial examples, the decision boundaries of the target model expand to correctly classify the ensemble adversarial examples into their original class. As shown in Fig. 2, Step 1 and Step 2 are repeated until the last pair of models is reached. At this point, hierarchical adversarial examples are generated for the last two target models.

## 5. Experiment Setup

The experiment setup consisted of the adversarial examples generated by the proposed method tested against unknown classifiers, which are different architectures. We used TensorFlow [22], one of the most widely used open source libraries for machine learning. Xeon E5-2609 1.7-GHz servers were used in the experiment.

### 5.1 Datasets

MNIST [14] and CIFAR10 [17] were used as datasets in the experiment. MNIST is a standard dataset with handwritten images from 0 to 9. This dataset is widely used for machine learning because the learning time is fast and it can be easily applied to experiments. CIFAR10 is a standard dataset that consists of ten images of planes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. For MNIST, 60,000 training data and 10,000 test data were used, and for CIFAR10, 50,000 training data and 10,000 test data were used.

### 5.2 Pretraining of Target Models

The pretrained target models have two configurations: homogeneous architecture and heterogeneous architecture. Homogeneous architectures are neural networks with the same structures but different parameters, whereas heterogeneous architectures are different neural networks with different parameters.

#### 5.2.1 MNIST

Table A· 1 of the Appendix shows the $D_{\text{same-}i}$ $(1 \leq i \leq 12)$ models used to generate homogeneous architectures, which are convolutional neural network architectures [23]. Table A· 2 shows the $D_{\text{same-}i}$ $(1 \leq i \leq 12)$ model parameters. As shown in Table A· 3, pretrained target models $D_{\text{same-}i}$ $(1 \leq i \leq 12)$ were created by learning different training data, while maintaining the accuracy of the original sample at over 97%. The target models thus generated were fixed models, so they remained unchanged during the generation of adversarial examples.

Table A· 4 of the Appendix shows the $D_{\text{differ-}i}$ $(1 \leq i \leq 12)$ models used to generate heterogeneous architectures. Table A· 6 shows the $D_{\text{differ-}i}$ $(1 \leq i \leq 12)$ model parameters and the accuracy of the original data. Models $D_{\text{differ-}i}$ $(1 \leq i \leq 12)$ were trained by learning about 60,000 training data. The results (Table A· 4) show that the pretrained target models $D_{\text{differ-}i}$ $(1 \leq i \leq 12)$ have more than 98% accuracy for about 10,000 test data.

#### 5.2.2 CIFAR10

Table A· 1 of the Appendix shows $D_{\text{same-}i}$ $(1 \leq i \leq 8)$ models used to generate homogeneous architectures, which are six-layer neural networks with four convolution layers [9], [11]. Table A· 2 shows the $D_{\text{same-}i}$ $(1 \leq i \leq 8)$ model parameters. As shown in Table A· 3, pretrained target models $D_{\text{same-}i}$ $(1 \leq i \leq 8)$ were created by learning different training data, while maintaining the accuracy of the original sample at approximately 80%. This method uses the CIFAR10 model of the Carlini method [9] and the distillation method [11]. The CIFAR10 model is a state-of-the-art method of using unaugmented data. The target models thus generated were fixed models, so they remained unchanged during the generation of adversarial examples.

Table A· 5 of the Appendix shows $D_{\text{differ-}i}$ $(1 \leq i \leq 8)$ models used to generate heterogeneous architectures using the VGG-19 model [24]. Table A· 6 shows the $D_{\text{differ-}i}$ $(1 \leq i \leq 8)$ model parameters and the accuracy of the original data. Models $D_{\text{differ-}i}$ $(1 \leq i \leq 8)$ were trained by learning approximately 50,000 training data. The results (Table A· 5) show that the pretrained target models $D_{\text{differ-}i}$ $(1 \leq i \leq 8)$ have approximately 80% or 91% accuracy for about 10,000 test data.

### 5.3 Test Adversarial Example Generation for Each Method

For testing, adversarial examples were generated for the baseline method, the ensemble method [16], and the proposed method. The goal of including the baseline method was to generate an adversarial example aimed at one normal target. The goal of the ensemble method was the conventional ensemble adversarial example method aimed at multiple target models. In terms of data, 1000 random test data were used to generate adversarial examples for testing.

Adam [25] uses the optimizer as a parameter of the transformer to generate the adversarial example s. On MNIST, the learning rate was 0.1, the initial constant was 0.001, and the number of iterations was 1000. On CIFAR10, the learning rate was 0.01, the initial constant was 0.001, and the number of iterations was 10,000.

### 5.4 Test Classifier

#### 5.4.1 MNIST

As shown in Table A· 7 and Table A· 8 of the Appendix, the test classifier consisted of three different unknown classifiers [26]: a five-layer fully connected network, a five-layer fully connected network with dropout 0.75, and a convnet (a five-layer neural network with three convolution layers). These three classifiers provided 95.41%, 98.17%, and 98.90% original sample accuracy, respectively.

#### 5.4.2 CIFAR10

As shown in Table A· 9 and Table A· 10 of the Appendix, the test classifier consisted of two different unknown classifiers [26]: an eight-layer neural network with six convolution layers, and the Mishkin method [27]. These two classifiers provided approximately 82% and 90% original sample accuracy, respectively.

## 6. Experimental Results

The experimental results are divided into two sections, homogeneous architectures and heterogeneous architectures, according to the type of pretrained target models.

### 6.1 MNIST

#### 6.1.1 Homogeneous Architectures

For the pretrained target models $D_{\text{same-}i}$ ($1 \leq i \leq 12$), Table 2 shows examples of original samples and adversarial examples generated by the baseline method, ensemble method, and proposed method. In Table 2, the adversarial examples generated by these three methods depict images similar to the original samples perceived by humans.

For the pretrained target models $D_{\text{same-}i}$ ($1 \leq i \leq 12$), Fig. 3 shows the training time, average distortion, and attack success rates of the baseline method, the ensemble method, and the proposed method when testing unknown classifiers that were a five-layer neural network, a five-layer neural network with dropout, and a convnet classifier.

Figure 3 (a) shows the training time required by each method to generate 1000 adversarial examples. For the baseline method, the training time was 0.2 h. For the ensemble method and the proposed method, as the number of target models increased, the training time increased. With the homogeneous architecture, the training time for the ensemble method was faster than the training time of the proposed method.

Figure 3 (b) shows the average distortion of the baseline method, the ensemble method, and the proposed method. The baseline method results in less distortion because it generates adversarial examples that target a single model. The ensemble method is more distorted than the baseline method because it generates an adversarial example that targets multiple models. The proposed method results in more distortion than the ensemble method because it generates an adversarial example that targets multiple models step by step. This is because the "divide and conquer" approach requires more processing.

Figure 3 (c) shows the attack success rates of the baseline method, the ensemble method, and the proposed method when testing an unknown classifier that was a five-layer neural network. The baseline method shows a 73% attack success rate because it generates an adversarial example targeting a single model. In the ensemble method, the attack success rate increased slightly as the number of local target models increased because in solving optimization problems it is difficult to satisfy the conditions of the multiple target models at the same time. The proposed method demonstrated higher attack success rates than the ensemble methods. This is because the proposed method targets a smaller number of local models than the previous ensemble method, similar to the concept of "divide and conquer." Therefore, as the number of target models increased, the attack success rate of the proposed method increased at a rate faster than the other methods because of its better learning.

Figure 3 (d) shows the attack success rates of the baseline method, the ensemble method, and the proposed method when testing an unknown classifier that was a five-layer network with dropout 0.75. The attack success rate of the baseline method was 64.2% for this unknown classifier. The proposed method and the ensemble method had higher attack success rates than the baseline method for an unknown classifier. In particular, the proposed method had a higher attack success rate than the ensemble method because it generates an adversarial example with a high probability of escaping the decision boundary of the unknown classifier due to the hierarchical technique.

Figure 3 (e) shows the attack success rates of the baseline method, the ensemble method, and the proposed method when testing an unknown classifier that was a convnet (a five-layer neural network with three convolution layers). As the figure depicts, the proposed method had higher attack success rates than the ensemble method and the baseline method.

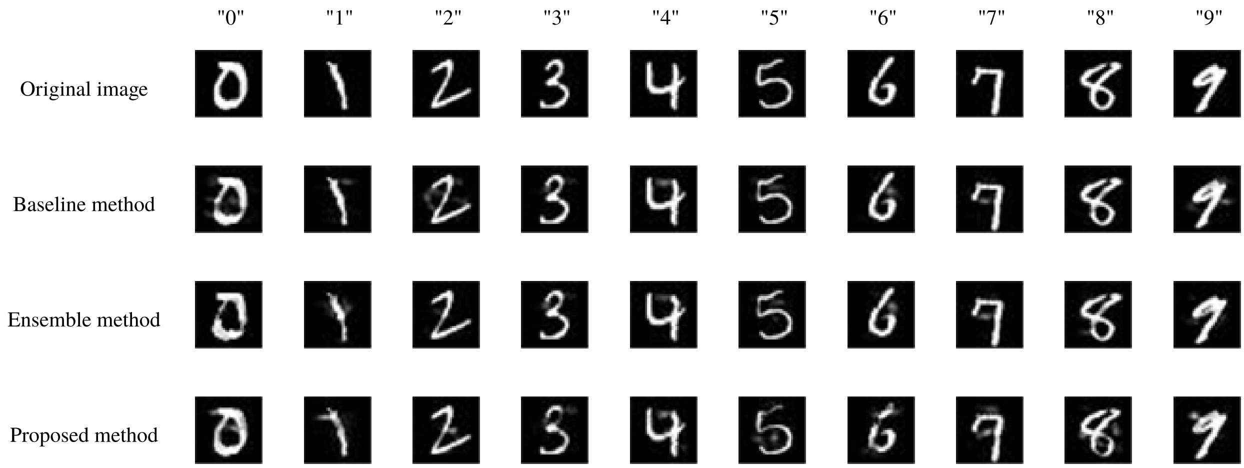
#### 6.1.2 Heterogeneous Architectures

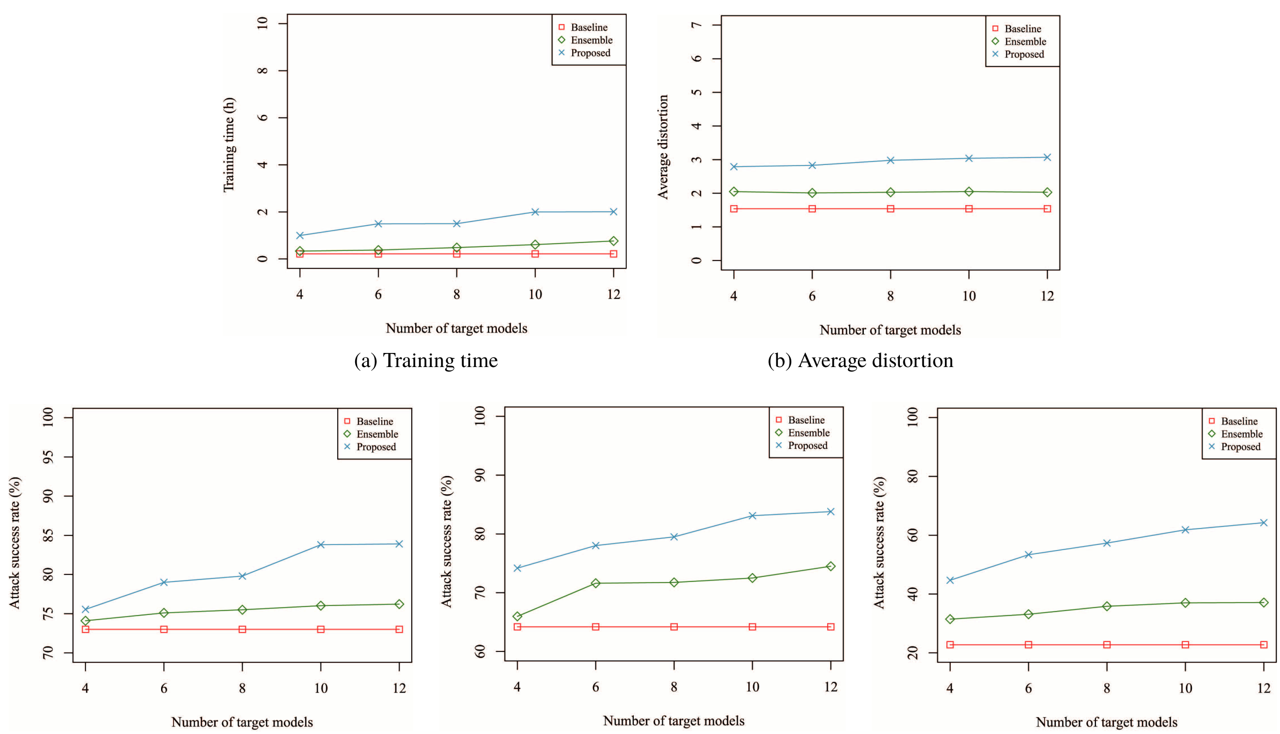
Table 3 shows examples of original samples and adversarial examples generated by the baseline method, ensemble method, and proposed method when the number of target models was 12. Table 3 shows that the adversarial examples generated by these three methods produce images that are similar to the original samples derived by human perception, similar to the cases shown in Table 2.

For the pretrained target models $D_{\text{differ-}i}$ ($1 \leq i \leq 12$), Fig. 4 shows the training time, average distortion, and attack success rates of the baseline method, the ensemble method, and the proposed method when testing unknown classifiers corresponding to the five-layer neural network, five-layer neural network with dropout, and convnet classifiers.

The pattern results in Fig. 4 are similar to the pattern results in Fig. 3. Although the proposed method results in slightly more distortion than the other methods, the proposed method demonstrates a higher attack success rate than the other methods while maintaining human perception. Figure 4 focuses on the differences from the results that are shown in Fig. 3.

**Table 2** A sampling of adversarial examples for baseline method, ensemble method, and proposed method for pretrained target models $D_{\text{same-}i}$ $(1 \leq i \leq 12)$.




(a) Training time

(b) Average distortion

(c) Attack success rate for the unknown five-layer network

(d) Attack success rate for the unknown five-layer network with dropout 0.75

(e) Attack success rate for the unknown convnet

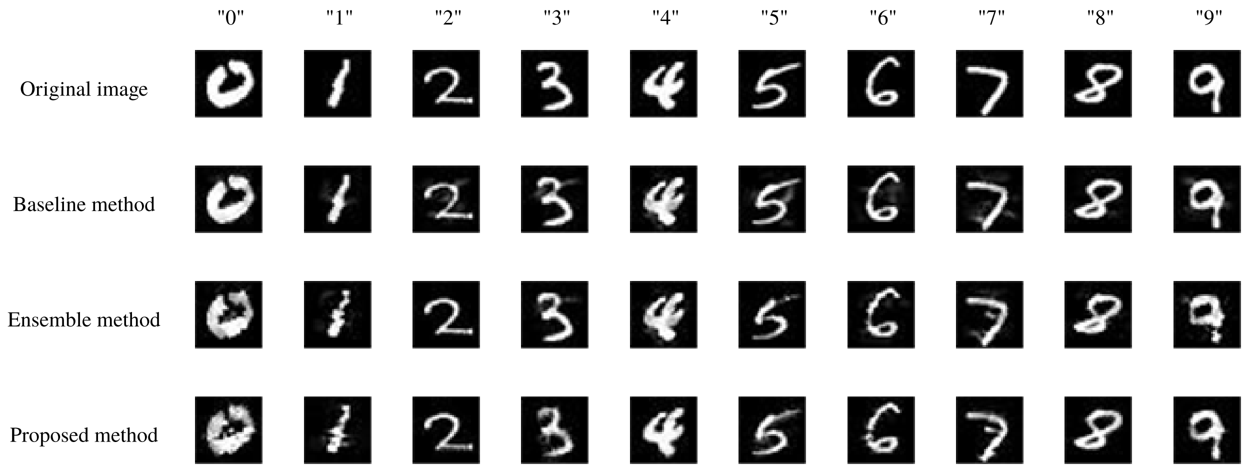
**Fig. 3** Results of comparison of baseline method, ensemble method, and proposed method for pre-trained target models $D_{\text{same-}i}$ $(1 \leq i \leq 12)$.

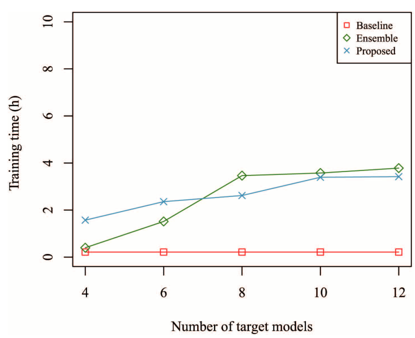
Unlike Fig. 3 (a), Fig. 4 (a) shows that the training time for the ensemble method was higher than the training time for the proposed method. This is because the ensemble method trains multiple different target models simultaneously in heterogeneous architectures. Additionally, the overall average training time for each method was generally higher than those shown in Fig. 3 (b).

Unlike Fig. 3 (b), Fig. 4 (b) shows that the distortion by the ensemble method increased to almost the same levels as that by the proposed method as the number of target models increased. This result shows that the distortion of the ensemble method increases as the number of target models increases in the case of heterogeneous architecture.

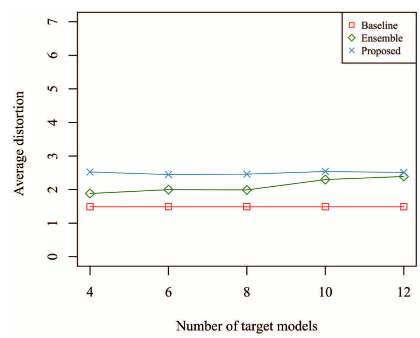
Figure 4 (c) and Fig. 4 (d) show that the proposed method demonstrates a better attack success rate than the other methods, similar to Fig. 3 (d) and Fig. 3 (d). However, the difference in the attack success rate between the proposed method and the ensemble method shown in Fig. 4 (c)

**Table 3** A sampling of adversarial examples for baseline method, ensemble method, and proposed method for pretrained target models $D_{\text{differ-}i}$ $(1 \leq i \leq 12)$.

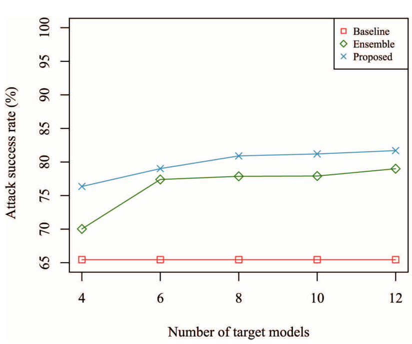


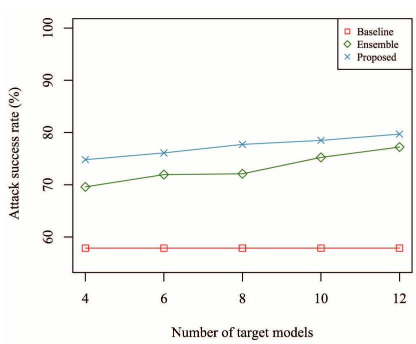

(a) Training time



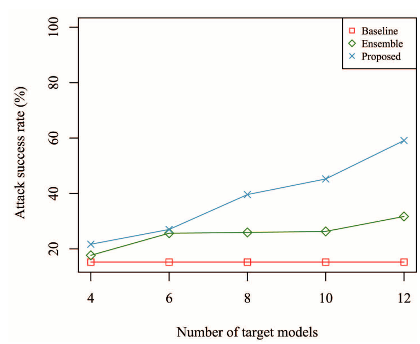

(b) Average distortion




(c) Attack success rate for the unknown five-layer network




(d) Attack success rate for the unknown five-layer network with dropout 0.75




(e) Attack success rate for the unknown convnet

**Fig. 4** Results of comparison of baseline method, ensemble method, and proposed method for pretrained target models $D_{\text{differ-}i}$ $(1 \leq i \leq 12)$.

and Fig. 4 (d) is 67.62% and 49.02% less than that of Fig. 3 (c) and Fig. 3 (d), respectively.

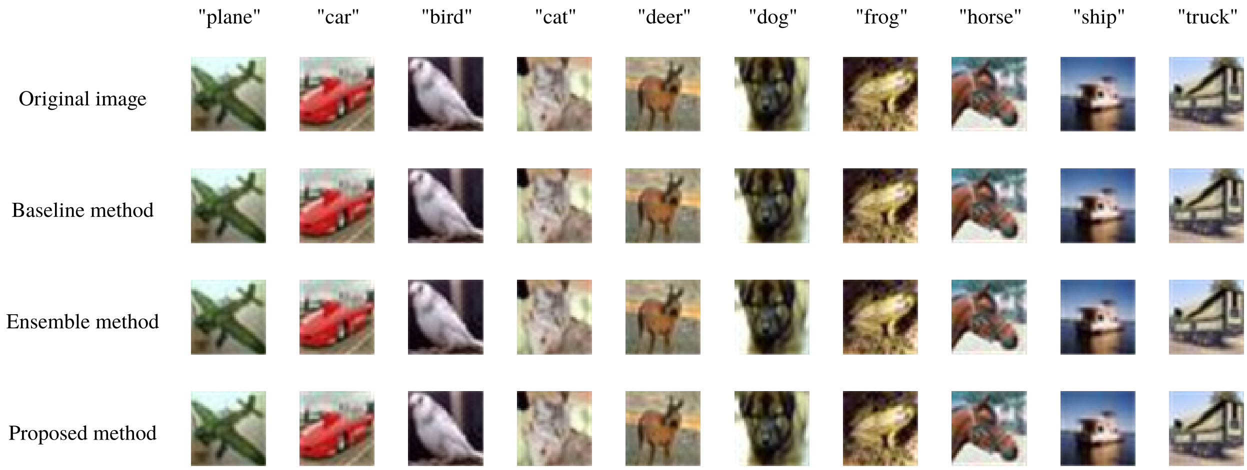
Figure 4 (e) shows that the attack success rate of the proposed method increased at a higher rate than the other methods as the number of target models increased, as depicted in Fig. 3 (e). For the unknown classifier of the convnet, the proposed method was more effective, as shown by Fig. 3 (e) and Fig. 4 (e).

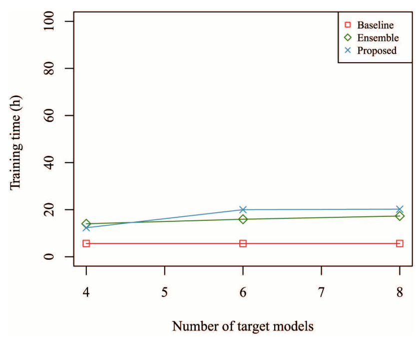
## 6.2 CIFAR10

### 6.2.1 Homogeneous Architectures

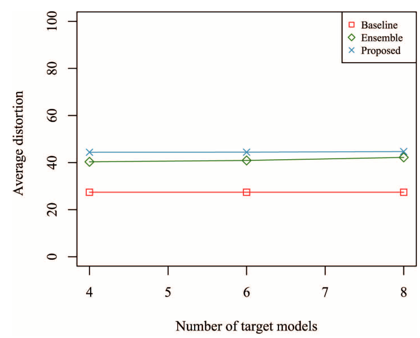
For pretrained target models $D_{\text{same-}i}$ $(1 \leq i \leq 8)$, Table 4 shows examples of original samples and adversarial examples generated by the baseline method, ensemble method, and proposed method. The adversarial examples generated by these three methods are more similar to the original

**Table 4** A sampling of adversarial examples for baseline method, ensemble method, and proposed method for pretrained target models $D_{\text{same-}i}$ $(1 \leq i \leq 8)$.

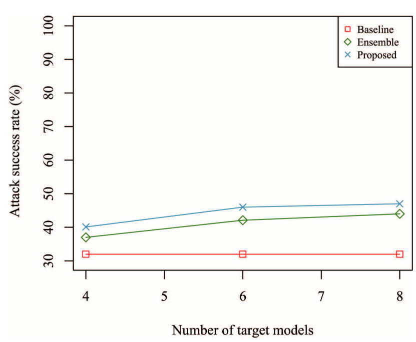


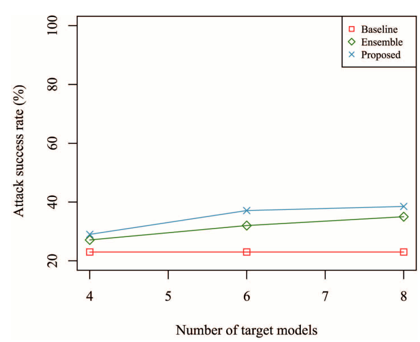

(a) Training time




(b) Average distortion




(c) Attack success rate for the unknown eight-layer network with dropout 0.5




(d) Attack success rate for the unknown Mishkin method

**Fig. 5** Results of comparison of baseline method, ensemble method, and proposed method for pre-trained target models $D_{\text{same-}i}$ $(1 \leq i \leq 8)$.

samples than those on MNIST.

For pretrained target models $D_{\text{same-}i}$ $(1 \leq i \leq 8)$, Fig. 5 shows the training time, average distortion, and attack success rates of the baseline method, the ensemble method, and the proposed method when testing the unknown classifiers that were the eight-layer neural network with six convolution layers, and the Mishkin method.

Figure 5 (a) shows the training times for the baseline method, the ensemble method, and the proposed method. Unlike the case in Fig. 3 (a) (for MNIST), the training time for the ensemble method was similar to the training time for the proposed method. However, the average training time for each method was longer than that in Fig. 3 (a) (on MNIST).

Similar to Fig. 3 (b) (MNIST), Fig. 5 (b) shows that the average distortion by the proposed method was greater than with the other methods. However, there was no difference in human perception, as shown in Table 4.

**Table 5** A sampling of adversarial examples for baseline method, ensemble method, and proposed method for pretrained target models $D_{\text{differ-}i}$ $(1 \leq i \leq 8)$.

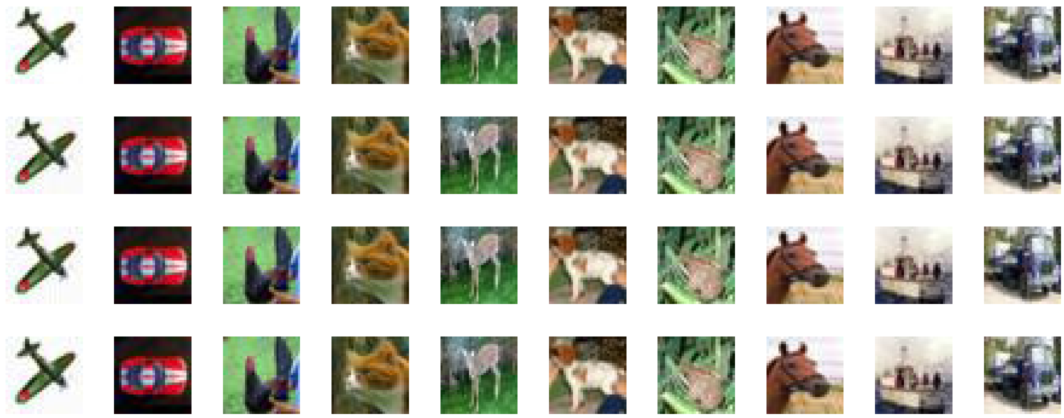
"plane" "car" "bird" "cat" "deer" "dog" "frog" "horse" "ship" "truck"

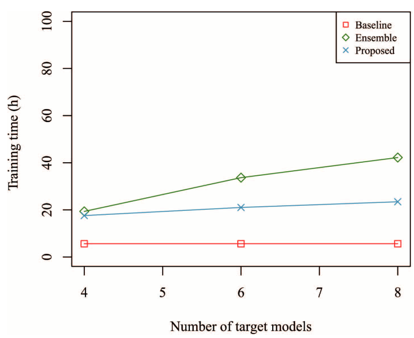
Original image

Baseline method

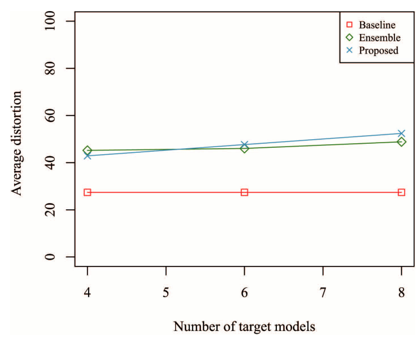
Ensemble method

Proposed method

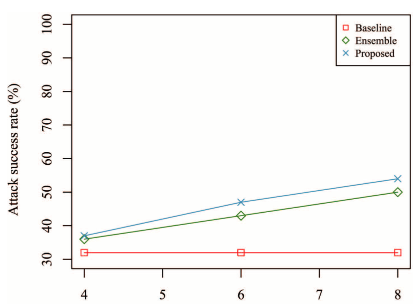




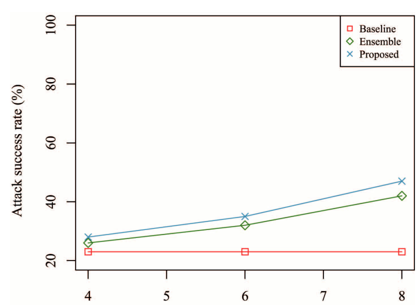

(a) Training time




(b) Average distortion




(c) Attack success rate for the unknown eight-layer network with dropout 0.5




(d) Attack success rate for the unknown Mishkin method

**Fig. 6** Results of comparison of baseline method, ensemble method, and proposed method for pretrained target models $D_{\text{differ-}i}$ $(1 \leq i \leq 8)$.

Figure 5 (c) shows the attack success rates of the baseline method, the ensemble method, and the proposed method when testing an unknown classifier that was an eight-layer neural network with six convolution layers. The baseline method, ensemble method, and proposed method showed a 32%, 44%, and 47% attack success rate, respectively. For the proposed method and the ensemble method, the attack success rate slightly increased as the number of local target models increased.

Figure 5 (d) shows the attack success rates of the baseline method, the ensemble method, and the proposed method when testing the unknown classifier of the Mishkin method. The baseline method, ensemble method, and proposed method showed a 23%, 35%, and 39% attack success rate, respectively. Similar to Fig. 5 (c), this result shows an attack success rate of the proposed method 4% higher than that of the ensemble method. However, the overall attack success rate was lower than that shown in Fig. 5 (c).

#### 6.2.2 Heterogeneous Architectures

Table 5 shows examples of original samples and adversarial examples generated by the baseline method, ensemble method, and proposed method when the number of target models was eight. The table shows that the adversarial examples generated by these three methods have images that are similar to the original sample derived by human perception, similar to Table 4.

For pretrained target models $D_{\text{differ-}i}$ ($1 \leq i \leq 8$), Fig. 6 shows the training time, average distortion, and attack success rate of the baseline method, the ensemble method, and the proposed method when testing unknown classifiers that were an eight-layer neural network with six convolution layers, and the Mishkin method.

Figure 6 (a) shows the training times for the baseline method, the ensemble method, and the proposed method. Unlike the case in Fig. 5 (a), the training time of the ensemble method is higher than the training time of the proposed method. This is because the ensemble method requires more time in a heterogeneous architecture to train different and complex target models simultaneously.

Figure 6 (b) shows the average distortion of the baseline method, the ensemble method, and the proposed method. Unlike the case in Fig. 5 (b), there is little difference between the ensemble method and the proposed method.

Figure 6 (c) and Fig. 6 (d) show that the proposed method achieved a better attack success rate than the other methods, similar to Fig. 5 (c) and Fig. 5 (d). Unlike MNIST, however, the attack success rate of the proposed method is slightly higher than that of the ensemble method in Fig. 6 (c) and Fig. 6 (d), by 4% and 5%, respectively.

## 7. Discussion

In this section, we discuss several aspects and findings of our proposed scheme.

***Transferability.*** Transferability is a transfer attack method in which an adversarial example generated by a known target model can also cause the misclassification of an unknown target model. Such an attack is possible because the unknown classifier basically has high accuracy for the original sample. Thus, it is likely that the unknown classifier has a classification boundary similar to that of the other models that demonstrate a high accuracy for the original sample.

The experimental results show that the proposed method can perform a transfer attack to achieve a reasonable attack success rate for the unknown classifier. The baseline method showed an average attack success rate of 49.75% and 27.5% with MNIST and CIFAR10, respectively. The ensemble method showed an average attack success rate of 59.44% and 36.4% with MNIST and CIFAR10, respectively. By comparison, the proposed method achieved an average attack success rate of 68.69% and 40.5% with MNIST and CIFAR10, respectively.

***Scalability.*** We investigated the variations in distortion and the attack success rates of each classifier by increasing several models in terms of scalability. It was observed that the attack success rates for the unknown classifiers increase as the number of target models increases.

***Distortion.*** It was shown that even if the number of target models increases, the distortion does not change significantly. This is because the change in the decision boundary, which is the union of that of several target models, is not significant. Basically, target models are pretrained models that recognize the original sample as the original class with high accuracy. Therefore, it is highly probable that the decision boundaries of several target models are close to each other, which ensures high accuracy with the original sample. The ensemble method that attacks multiple target models finds an adversarial example at a point outside the decision boundary of the union of the target models. Thus, even if the number of target models increases, the variation in the distortion with the ensemble method is not large.

In general, the proposed method results in more distortion than the other methods while maintaining human perception. This is because more processes are required to generate hierarchical adversarial examples using the divide-and-conquer method than with the ensemble method.

***Dataset.*** We evaluated the performance of the proposed method using the MNIST and CIFAR10 datasets. The experiment results show that the training time, average distortion rate, and attack success rate depend on the dataset. In terms of the average distortion, although the distortion on CIFAR10 is higher than that on MNIST, human recognition with CIFAR10 is more similar to that of the original sample than that with MNIST. This result shows that with an adversarial example generated for a 3D image such as in CIFAR10, no problem can be detected by human eye. In terms of attack success rate, the proposed method shows better performance on MNIST than on CIFAR10. In terms of training time, CIFAR10 needs to train longer than MNIST because CIFAR10 is 3D and has larger image sizes.

***Type of pretrained model.*** We studied the transferability between homogeneous architectures, which create different target models by learning different data from the same models. Moreover, we studied the transferability between heterogeneous architectures, which create different target models by learning training data from different models. Therefore, this study provides an experimental analysis of both the homogeneous architecture and the heterogeneous architecture.

***Target.*** For the unknown classifier scenario, the proposed method is effective. For example, if an attacker is attempting to attack an unknown enemy tank or an armored fighting vehicle or is attempting to attack any unknown classifier for a new car that has not yet been released, the attacker will have a higher attack success rate using the proposed method.

***Training time.*** We evaluated the training time for each method. The experiment results show that the training time required for the proposed method is generally higher when the multiple target models are the same simple model. The proposed method divides multiple target models hierarchically and processes them step by step. However, in the case of heterogeneous architectures, the ensemble method requires more time than the proposed method. With the ensemble method, when the multiple target models are different and complex, more time is required to train them at the same time.

***Application.*** With traffic signs, adversarial example attacks against unknown classifiers are more practical because it is difficult to change the signs after they are distributed on the road. From this practical point of view, it is considered that a sign generated using the proposed method would be effective.

***Limitation.*** In terms of scalability, we only tested up to twelve models and eight models in MNIST and CIFAR10, respectively.

## 8. Conclusion

In this study, we proposed the hierarchical adversarial example, which achieves a reasonable attack success rate against unknown classifiers. The hierarchical adversarial example is generated by step-by-step generation and an adversarial training method. The experimental results show that our proposed method can achieve an attack success rate for an unknown classifier of up to 9.25% and 18.94% higher with MNIST, and 4.1% and 13% higher with CIFAR10, compared with the previous ensemble method and the conventional baseline method, respectively. We also observed the scalability aspect. Even though there is less change in distortion, the attack success rate increases as the number of target models increases.

In future research, we will extend our experiments to other standard image datasets, such as ImageNet. As the transformer, we will use a generative adversarial network [28] instead of the Carlini attack method [9]. From an application point of view, the proposed method will extend the range of voice [29], [30], music [31], face [32], and CAPTCHA systems [33], [34] to attack unknown classifiers. The evaluation and analysis of the higher scalability of the target model will also be a part of future studies. Finally, developing a countermeasure for the proposed scheme will be another challenge.

## Acknowledgments

We thank the editors and the anonymous reviewers, who gave us very helpful comments that improved this paper. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2016R1A4A1011761) and (MSIT) (2017R1A2B4006026) and Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00173, Security Technologies for Financial Fraud Prevention on Fintech).

### References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol.61, pp.85–117, 2015.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," International Conference on Learning Representations, 2015.
[3] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal Process. Mag., vol.29, no.6, pp.82–97, 2012.
[4] I. Arel, D.C. Rose, and R. Coop, "Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition," AAAI Fall Symposium: Biologically Inspired Cognitive Architectures, 2009.
[5] G.L. Oliveira, A. Valada, C. Bollen, W. Burgard, and T. Brox, "Deep learning for human part discovery in images," 2016 IEEE International Conference on Robotics and Automation (ICRA), pp.1634–1641, IEEE, 2016.
[6] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," Proc. 25th international conference on Machine learning, pp.160–167, ACM, 2008.
[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," International Conference on Learning Representations, 2014.
[8] F. Zhang, P.P.K. Chan, B. Biggio, D.S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," IEEE Trans. Cybern., vol.46, no.3, pp.766–777, 2016.
[9] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017 IEEE Symposium on Security and Privacy (SP), pp.39–57, IEEE, 2017.
[10] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.2574–2582, 2016.
[11] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," 2016 IEEE Symposium on Security and Privacy (SP), pp.582–597, IEEE, 2016.
[12] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, pp.135–147, ACM, 2017.
[13] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, and A. Swami, "Practical black-box attacks against machine learning," Proc. 2017 ACM on Asia Conference on Computer and Communications Security, pp.506–519, ACM, 2017.
[14] Y. LeCun, C. Cortes, and C.J.C. Burges, Mnist handwritten digit database, AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2, 2010.
[15] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," International Conference on Learning Representations, 2015.
[16] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable ad-

versarial examples and black-box attacks," ICLR, abs/1611.02770, 2017.
[17] A. Krizhevsky, V. Nair, and G. Hinton, The cifar-10 dataset, online: http://www.cs.toronto.edu/kriz/cifar.html, 2014.
[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," ICLR Workshop, 2017.
[19] S.M.M. Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), number EPFL-CONF-226156, 2017.
[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," IEEE Conference on Computer Vision and Pattern Recognition, 2009, CVPR 2009, pp.248–255, IEEE, 2009.
[21] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," arXiv preprint arXiv:1703.09387, 2017.
[22] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: A system for large-scale machine learning," OSDI, vol.16, pp.265–283, 2016.
[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol.86, no.11, pp.2278–2324, 1998.
[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," ICLR 2015, 2015.
[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," The International Conference on Learning Representations (ICLR), 2015.
[26] K. Sopyta, https://github.com/ksopyla, Github.
[27] D. Mishkin and J. Matas, "All you need is a good init," arXiv preprint arXiv:1511.06422, 2015.
[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," Advances in Neural Information Processing Systems, pp.2672–2680, 2014.
[29] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," USENIX Security Symposium, pp.513–530, 2016.
[30] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, pp.103–117, ACM, 2017.
[31] C. Kereliuk, B.L. Sturm, and J. Larsen, "Deep learning and music adversaries," IEEE Trans. Multimedia, vol.17, no.11, pp.2059–2071, 2015.
[32] A. Rozsa, M. Günther, E.M. Rudd, and T.E. Boult, "Facial attributes: Accuracy and adversarial robustness," Pattern Recognition Letters, 2017.
[33] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo, "No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation," IEEE Trans. Inf. Forensics Security, vol.12, no.11, pp.2640–2653, 2017.
[34] H. Kwon, Y. Kim, H. Yoon, and D. Choi, "Captcha image generation systems using generative adversarial networks," IEICE Trans. Inf. & Syst., vol.E101-D, no.2, pp.543–546, 2018.

## Appendix

**Table A·1** $D_{same-i}$ model architecture.

| Layter type | MNIST shape | CIFAR10 shape |
|---|---|---|
| Convolution+ReLU | [3, 3, 32] | [3, 3, 64] |
| Convolution+ReLU | [3, 3, 32] | [3, 3, 64] |
| Max pooling | [2, 2] | [2, 2] |
| Convolution+ReLU | [3, 3, 64] | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 64] | [3, 3, 128] |
| Max pooling | [2, 2] | [2, 2] |
| Fully connected+ReLU | [200] | [200] |
| Fully connected+ReLU | [200] | [200] |
| Softmax | [10] | [10] |

**Table A·2** $D_{same-i}$ model parameter.

| Parameter | MNIST | CIFAR10 |
|---|---|---|
| Learning rate | 0.1 | 0.001 |
| Momentum | 0.9 | 0.9 |
| Dealy rate | - | 10 |
| Dropout | 0.5 | 0.5 |
| Batch size | 128 | 128 |
| Epochs | 50 | 50 |

**Table A·3** $D_{same-i}$ pre-trained target models.

| Model | MNIST | | CIFAR10 | |
|---|---|---|---|---|
| | Training data | Rate | Training data | Rate |
| $D_{same-1}$ | 0~5000 | 97.75% | 5000~50000 | 80.8% |
| $D_{same-2}$ | 5000~10000 | 97.74% | 10000~50000 | 80.2% |
| $D_{same-3}$ | 10000~15000 | 97.85% | 15000~50000 | 80.1% |
| $D_{same-4}$ | 15000~20000 | 97.44% | 5000~45000 | 80.3% |
| $D_{same-5}$ | 20000~25000 | 97.7% | 5000~40000 | 80.0% |
| $D_{same-6}$ | 25000~30000 | 97.56% | 0~45000 | 80.9% |
| $D_{same-7}$ | 30000~35000 | 97.89% | 0~40000 | 80.3% |
| $D_{same-8}$ | 35000~40000 | 97.56% | 0~50000 | 81.3% |
| $D_{same-9}$ | 40000~45000 | 97.62% | | |
| $D_{same-10}$ | 45000~50000 | 97.90% | | |
| $D_{same-11}$ | 50000~55000 | 97.48% | | |
| $D_{same-12}$ | 55000~60000 | 97.09% | | |

**Table A·4** In MNIST, $D_{differ-i}$ $(1 \leq i \leq 12)$ pre-trained target models and accuracy of the original data; FC is fully-connected neural network and CNN is a convolution neural network. 1-FC: [200], 2-FC: [512, 200], 3-FC: [512, 512, 200], 4-FC: [512, 512, 200, 100], 1-CNN: [3, 3, 32] with max pooling [2, 2], 2-CNN: [[3, 3, 32], [3, 3, 32]] with max pooling [2, 2], and Softmax [10].

| Model | MNIST Architecture | Accuracy |
|---|---|---|
| $D_{differ-1}$ | LeNet: 2-CNN+1-FC | 98.86% |
| $D_{differ-2}$ | 1-FC (Dropout 0.5) | 93.83% |
| $D_{differ-3}$ | 2-CNN+2-FC | 98.65% |
| $D_{differ-4}$ | 1-CNN+1-FC | 98.29% |
| $D_{differ-5}$ | 3-FC (Dropout 0.2) | 98.29% |
| $D_{differ-6}$ | RCNN: 1-CNN+1FC (Dropout 0.5) | 98.39% |
| $D_{differ-7}$ | 2-CNN+2-FC (Dropout 0.5) | 98.76% |
| $D_{differ-8}$ | 2-FC (Dropout 0.2) | 98.35% |
| $D_{differ-9}$ | 1-CNN+2-FC (Dropout 0.5) | 98.31% |
| $D_{differ-10}$ | 2-CNN+3-FC (Dropout 0.5) | 98.62% |
| $D_{differ-11}$ | 4-FC (Dropout 0.2) | 98.76% |
| $D_{differ-12}$ | 1-CNN+3-FC (Dropout 0.5) | 98.34% |

**Table A·5** In CIFAR10, $D_{differ-i}$ ($1 \leq i \leq 8$) pre-trained target models and accuracy of the original data; $D_{differ-2}$ is VGG-19 model [24], F is fully-connected neural network, and C is a convolution neural network. 2F: [4096, 4096], 3F: [4096, 4096, 4096], 5F: [4096, 4096, 4096, 4096, 4096], 2F(256): [256, 256], 5F(256): [256, 256, 256, 256, 256], 2C(64): [[3, 3, 64]. [3, 3, 64] with max pooling [2, 2], 3C(64): [[3, 3, 64], [3, 3, 64], [3, 3, 64]] with max pooling [2, 2], 4C(64): [[3, 3, 64], [3, 3, 64], [3, 3, 64], [3, 3, 64]] with max pooling [2, 2], 5C(64): [[3, 3, 64], [3, 3, 64], [3, 3, 64], [3, 3, 64], [3, 3, 64]] with max pooling [2, 2], 2C(128): [[3, 3, 128]. [3, 3, 128] with max pooling [2, 2] and Softmax [10].

| Model | CIFAR10 Architecture | Accuracy |
|---|---|---|
| $D_{differ-1}$ | 2C(64)+2C(128)+2F(256) | 81.3% |
| $D_{differ-2}$ | 2C(64)+2C(128)+4C(256)+4C(512)+2F | 91.3% |
| $D_{differ-3}$ | 2C(64)+2C(128)+4C(256)+4C(512)+3F | 91.5% |
| $D_{differ-4}$ | 3C(64)+2C(128)+5F(256) | 80.1% |
| $D_{differ-5}$ | 4C(64)+2C(128)+2F(256) | 80.3% |
| $D_{differ-6}$ | 2C(64)+2C(128)+4C(256)+4C(512)+5F | 91.2% |
| $D_{differ-7}$ | 5C(64)+2C(128)+2F(256) | 80.5% |
| $D_{differ-8}$ | 3C(64)+2C(128)+4C(256)+4C(512)+2F | 91.0% |

**Table A·6** $D_{differ-i}$ model parameter.

| Parameter | MNIST | CIFAR10 |
|---|---|---|
| Learning rate | 0.1 | 0.001 |
| Momentum | 0.9 | 0.9 |
| Dealy rate | - | 10 |
| Dropout | - | 0.5 |
| Batch size | 128 | 128 |
| Epochs | 50 | 50 |

**Table A·7** In MNIST, the unknown classifier of 5-layer fully-connected network; $w_i$ is a weight of $i^{th}$ layer. $b_i$ is a constant of $i^{th}$ layer.

| Layer type | MNIST shape |
|---|---|
| Input layer | [batch, 784] |
| 1 layer+ReLU | $w_1$[784, 200], $b_1$[200] |
| 2 layer+ReLU | $w_2$[200, 100], $b_2$[100] |
| 3 layer+ReLU | $w_3$[100, 60], $b_3$[60] |
| 4 layer+ReLU | $w_4$[60, 30], $b_4$[30] |
| 5 layer+ReLU | $w_5$[30, 10], $b_5$[10] |
| One-hot encoded labels | [batch, 10] |

**Table A·8** In MNIST, the unknown classifier of convnet; convnet is 5-layer neural network with 3 convolution layers. $w_i$ is a weight of $i^{th}$ layer. $b_i$ is a constant of $i^{th}$ layer.

| Layer type | MNIST shape |
|---|---|
| Input layer | [batch, 784] |
| 1 convolution layer+ReLU | $w_1$[5,5, ,1,C1], $b_1$[C1] |
| 2 convolution layer+ReLU | $w_2$[3,3,C1,C2], $b_2$[C2] |
| Max pooling | [2, 2] |
| 3 convolution layer+ReLU | $w_3$[3,3,C2, C3], $b_3$[C3] |
| Max pooling | [2, 2] |
| 4 fully connected layer+ReLU | $w_4$[7*7*C3, FC4], $b_4$[FC4] |
| 5 output layer | $w_5$[FC4, 10], $b_5$[10] |
| One-hot encoded labels | [batch, 10] |

**Table A·9** In CIFAR10, the unknown classifier of 8 layer neural network with 6 convolution layers with dropout 0.5.

| Layer type | CIFAR10 shape |
|---|---|
| Convolution+ReLU | [3, 3, 64] |
| Convolution+ReLU | [3, 3, 64] |
| Max pooling | [2, 2] |
| Convolution+ReLU | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 128] |
| Max pooling | [2, 2] |
| Fully connected+ReLU | [500] |
| Fully connected+ReLU | [200] |
| Softmax | [10] |

**Table A·10** In CIFAR10, the unknown classifier of Mishkin method [27] with dropout 0.2.

| Layer type | CIFAR10 shape |
|---|---|
| Convolution+ReLU | [3, 3, 32] |
| Convolution+ReLU | [3, 3, 32] |
| Convolution+ReLU | [3, 3, 32] |
| Convolution+ReLU | [3, 3, 48] |
| Convolution+ReLU | [3, 3, 48] |
| Max pooling | [2, 2] |
| Convolution+ReLU | [3, 3, 80] |
| Convolution+ReLU | [3, 3, 80] |
| Convolution+ReLU | [3, 3, 80] |
| Convolution+ReLU | [3, 3, 80] |
| Max pooling | [2, 2] |
| Convolution+ReLU | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 128] |
| Convolution+ReLU | [3, 3, 128] |
| Max pooling | [8, 8] |
| Fully connected+ReLU | [500] |
| Softmax | [10] |

**Hyun Kwon** received the B.S degree in mathematics from Korea Military Academy, South Korea, in 2010. He also received the M.S. degree in School of Computing from Korea Advanced Institute of Science and Technology (KAIST) in 2015. He is currently working toward the Ph.D. degree at School of Computing, KAIST. His research interests include machine learning, information security, computer security, and intrusion tolerant system.

**Yongchul Kim** received the B.E. degree in electrical engineering from the Korea Military Academy, South Korea, in 1998, the M.S. degree in electrical engineering from the University of Surrey, U.K., in 2001, and the Ph.D. degree in electrical and computer engineering with the department of electrical and computer engineering, North Carolina State University, in 2007. He has been a professor in the department of electrical engineering at the Korea Military Academy. His research interests include WiMAX and wireless relay networks.

**Ki-Woong Park** received the B.S. degree in computer science from Yonsei University, South Korea, in 2005, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2007, and the Ph.D. degree in electrical engineering from KAIST in 2012. He received a 2009-2010 Microsoft Graduate Research Fellowship. He worked for National Security Research Institute as a senior researcher. He has been a professor in the department of computer and information security at Sejong University. His research interests include security issues for cloud and mobile computing systems as well as the actual system implementation and subsequent evaluation in a real computing system.

**Hyunsoo Yoon** received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1979, the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1981, and the Ph.D. degree in computer and information science from The Ohio State University, Columbus, OH, in 1988. He is currently a professor at School of Computing in KAIST. His research interests include mobile ad hoc networks, wireless networks, and network security.

**Daeseon Choi** received the B.S. degree in computer science from Dongguk University, Korea, in 1995, the M.S. degree in computer science from Pohang Institute of Science and Technology (POSTECH), Korea, in 1997, and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Korea, in 2009. He is currently a professor at Department of Medical Information, Kongju National University, Korea. His research interests include information security and identity management.