Diversity Adversarial Training against Adversarial Attack on Deep Neural Networks

Hyun Kwon 1 and Jun Lee 2,*

1 Department of Electrical Engineering, Korea Military Academy, Seoul 01805, Korea; hkwon.cs@gmail.com or khhk@kaist.ac.kr
2 Division of Computer Information and Science, Hoseo University, Asan-si 31499, Korea
* Correspondence: junlee@game.hoseo.edu; Tel.: +82-10-2852-6450

Abstract: This paper presents research focusing on visualization and pattern recognition based on computer science. Although deep neural networks demonstrate satisfactory performance regarding image and voice recognition, as well as pattern analysis and intrusion detection, they exhibit inferior performance towards adversarial examples. Noise introduction, to some degree, to the original data could lead adversarial examples to be misclassified by deep neural networks, even though they can still be deemed as normal by humans. In this paper, a robust diversity adversarial training method against adversarial attacks was demonstrated. In this approach, the target model is more robust to unknown adversarial examples, as it trains various adversarial samples. During the experiment, Tensorflow was employed as our deep learning framework, while MNIST and Fashion-MNIST were used as experimental datasets. Results revealed that the diversity training method has lowered the attack success rate by an average of 27.2 and 24.3% for various adversarial examples, while maintaining the 98.7 and 91.5% accuracy rates regarding the original data of MNIST and Fashion-MNIST.

Keywords: machine learning; adversarial example; defense technology; deep neural network (DNN)

1. Introduction

Recently, visualization and pattern recognition based on computer science has received extensive attention. Particularly, deep neural networks [1] demonstrated good performance for voice recognition, image classification, medical diagnosis [2,3], and pattern analysis. However, these networks suffer from some security vulnerabilities. According to Barreno, Marco et al. [4], such security issues can be classified into two main groups: the causative attack and the exploratory attack. A causative attack degrades the performance of the deep neural network by intentionally adding malicious samples to the training data during the network learning phase. The poisoning [5] and backdoor attacks [6,7] are representative examples of a causative attack. However, the exploratory attack is a method of the evasive attack by manipulating test data on a model that has already been trained; it is a realistic attack because there exists no assumption that the learning data can be accessed, as that in a causative attack. An adversarial attack [5,8] is a representative example of an exploratory attack. This study focused on proposing a defense method against adversarial example attacks. An adversarial example [9] that adds some specific noise to the original data can be correctly recognized by humans, while incorrectly classified by deep neural networks. Such examples could deceive some deep neural networks, such as those used in autonomous vehicles or medical businesses, and may lead to unexpected outcomes.

There exist various defensive approaches against adversarial attacks. These methods can be divided into two major categories: those of manipulating data and of making deep neural networks become more robust. The data manipulation method [10–15] reduces the attack effect of the noise of the adversarial example by filtering out the noise or by resizing the input data. In contrast, approaches of making deep neural networks to
be more robust include methods of distillation [16] and adversarial training [9,17]. The distillation method uses two neural networks to prevent the generation of adversarial example. However, the adversarial training method becomes robust to adversarial attacks by additionally training the target model on adversarial data that are generated from a local neural network.

Among them, the adversarial training method is simple and effective. According to previous studies [18,19], the adversarial training is evaluated to be more efficient than other methods in terms of practical defense performance. However, the existing adversarial training trains the target model with adversarial examples generated by one single attack approach. However, if the target model can be trained with adversarial examples that are generated from multiple attack methods rather than a single one, it would become more robust against unknown adversarial attacks.

In this paper, we demonstrate a diversity adversarial training method, in which the target model is trained with additional adversarial examples that are generated by various attack approaches, such as the fast gradient descent method (FGSM) [20], iterative-FGSM (I-FGSM) [21], DeepFool [22], and Carlini and Wagner (CW) [23]. Our contributions can be summarized as follows: First, we demonstrate a diversity adversarial training approach that trains the target model with adversarial examples generated by various methods. In addition, detailed explanations regarding the construction principle and structure of the method were presented. Moreover, we analyzed images of the adversarial examples generated by different methods, corresponding attack success rate, and accuracy of the diversity training method. In addition, we verified the performance of our method using the MNIST [24] and Fashion-MNIST [25] datasets.

The remainder of this paper is organized as follows: Section 2 describes related research on adversarial examples. Section 3 explains the diversity training method. Experiments and analysis are presented in Section 4, while the diversity training scheme is discussed in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

The objective of an adversarial attack is to deceive the deep neural network to misclassify, while minimizing the distortion from the original sample. The first research specifically focused on this topic was introduced by Szegedy et al. [9]. In this section, we describe previous researches [26,27] on adversarial attacks and counter attack methods.

2.1. Adversarial Attacks

Adversarial examples are generated through the feedback of the results of the input values to the trained victim model. The transformer adds some noise to the input value and sends it to the victim model, calculates a probability based on the model output, and transforms it by adding noise again so that the classification probability of the target class chosen by the attacker is high. While repeating this process, adding minimal noise to the original data creates adversarial examples that the victim model misrecognizes. There exist FGSM [20], I-FGSM [21], DeepFool [22], and CW [23] methods to generate such adversarial examples, which may vary based on the way they are generated.

2.2. Defense Methods against Adversarial Attacks

Methods to defend against adversarial attacks can be divided into two categories: manipulating data and enhancing deep neural networks. The data manipulation method is to reduce the noise attack effect of adversarial examples based on filtering or resizing the input data [10–13]. However, approaches of making deep neural networks robust include adversarial training [9,24] and distillation methods [16]. The distillation method uses two neural networks to prevent adversarial examples. Initially, the class probability for the input value is calculated by the first network, and then is provided to the second network as a label. The second neural network prevents the generation of adversarial examples through complicating the gradient descent calculation among adversarial examples. The adversarial
training method additionally trains the target model with adversarial examples that are generated by a local neural network, so as to enhance defense against adversarial attacks. In this study, we demonstrate a dynamic adversarial training approach, in which the target model is trained by adversarial examples that are obtained with various generation methods, which is more robust against adversarial attacks than other existing methods.

3. Methodology

The diversity training method consists of two stages: generation of various adversarial examples and a process in which the model learns these samples. That is, it first generates various adversarial examples, and provides them as additional training data to the model so as to increase its robustness against unknown adversarial attacks. First, the diversity training method creates various adversarial examples using FGSM, I-FGSM, DeepFool, and CW methods using the local model known to the attacker. Subsequently, the target model is additionally trained with these samples. Through this process, the robustness of the target model against adversarial attacks can be increased, as shown in Figure 1.

![Figure 1](image-url)  
**Figure 1.** Overview of the diversity training scheme: the target model $M_T$ is trained with adversarial examples obtained from various generation methods using the local model $M_L$. A, B, C, and D indicate fast gradient descent method (FGSM), iterative-FGSM (I-FGSM), DeepFool, and Carlini and Wagner (CW) methods, respectively.

The diversity training method can be mathematically expressed as follows. The operation function of the local model $M_L$ is denoted as $f_L(x)$. The local model $M_L$ is trained with the original training dataset. Given the pretrained local model $M_L$, the original training data $x \in X$, their corresponding class labels $y \in Y$, and target class labels $y^* \in Y$, we solve the optimization problem of creating a targeted adversarial example $x^*$:

$$x^* : \arg\min_{x^*} L(x, x^*) \text{ s.t. } f_L(x^*) = y^*,$$

where $L(\cdot)$ is a distance matrix between the normal sample $x$ and transformed example $x^*$. $\arg\min_x F(x)$ indicates that $F(x)$ becomes minimal regarding the value of $x$. $f_L(\cdot)$ is a local model function that classifies the input value.

To generate these $x^*$, each adversarial example is obtained by the FGSM, I-FGSM, DeepFool, and CW methods.
**FGSM:** The FGSM method can create $x^*$ using $L_\infty$:

$$
x^* = x + \varepsilon \cdot \text{sign}\left(\nabla \text{loss}_{F,T}(x)\right),
$$

where $t$ and $F$ indicate the target class and model operation function, respectively. Here, the gradient descent is updated with the normal sample $x$, based on the $\varepsilon$ value, while $x^*$ is created through optimization. This method is simple yet exhibits good performance.

**I-FGSM:** I-FGSM is an extension of the FGSM. In this method, instead of changing the amount of $\varepsilon$ at each step, a smaller amount, $\alpha$, is changed and eventually clipped by $\varepsilon$:

$$
x_i^* = x_i^{*}_{t-1} - \text{clip}_\varepsilon(\alpha \cdot \text{sign}(\nabla \text{loss}_{F,T}(x_i^{*}_{t-1}))).
$$

I-FGSM obtains an adversarial example during a given iteration on a target model. Compared to FGSM, it demonstrates a higher attack prevention rate in terms of white box attacks.

**DeepFool:** The DeepFool approach creates an adversarial example with less distortion from the original sample; it generates $x^*$ through the linearization approximation. However, because the neural network is nonlinear, this method is more complicated than FGSM.

**CW:** The fourth method is the Carlini attack that can generate an adversarial example with 100% attack success rate, which uses a different objective function:

$$
D(x, x^*) + c \cdot f(x^*).
$$

This method estimates an appropriate binary $c$ value to obtain a high success rate of attack. In addition, it can control the attack success rate even at the cost of some increase in distortion by adjusting the confidence value as follows:

$$
f(x^*) = \max\{Z_i(x^*) \mid i \neq y\} - Z_i(x^*)_y,
$$

where $y$ is an original class, while $Z(\cdot)$ [28] represents the pre-softmax classification result vector.

The adversarial examples generated by each method are added to the training set that is used to train the target model $M_T$. This process can be described mathematically as follows.

The operation function of the target model $M_T$ is denoted as $f_i(x)$. The target model $M_T$ is first trained with the original training dataset. Given the adversarial example $x^* \in X$, original class $y \in Y$, and target classes $y^* \in Y$, the pre-trained target model $M_T$ is trained with $x^*$ with its corresponding label as the original class $y$, as follows:

$$
f_i(x^*) = y.
$$

In this manner, the target model is trained with various adversarial examples, and thus, its robustness against unknown adversarial examples is increased. The details of the diversity training scheme are illustrated in Algorithm 1.
Algorithm 1 Diversity adversarial training

Input:

\[ x_j \in X \quad \text{▷ Original training dataset} \]
\[ y \in Y \quad \text{▷ original class} \]
\[ t \quad \text{▷ validation data} \]
\[ M_L \quad \text{▷ local model} \]
\[ M_T \in X \quad \text{▷ Original training dataset} \]
\[ y \in Y \quad \text{▷ original class} \]
\[ t \quad \text{▷ validation data} \]
\[ M_L \quad \text{▷ local model} \]
\[ \text{fgs, ifgs, dp, cw} \quad \text{▷ FGSM, I-FGSM, DeepFool, CW methods} \]

Diversity adversarial training: \((M_L, x, \text{fgs}, \text{ifgs}, \text{dp}, \text{cw})\)

\[ X^* \leftarrow \text{Generation of adversarial example } (M_L, x, \text{fgs}) \]
\[ X^* \leftarrow \text{Generation of adversarial example } (M_L, x, \text{ifgs}) \]
\[ X^* \leftarrow \text{Generation of adversarial example } (M_L, x, \text{dp}) \]
\[ X^* \leftarrow \text{Generation of adversarial example } (M_L, x, \text{cw}) \]

Train the target model \( M_T \leftarrow (X, Y) + (X^*, Y) \)

Record accuracy of the target model \( M_T(t) \)

\text{return } M_T

4. Experiment Setup

Through experiments, we demonstrate that the diversity training method can effectively resist adversarial attacks. This section presents the experimental setup for evaluating the diversity training method.

4.1. Datasets

The MNIST [24] and Fashion-MNIST [25] are used as our evaluation datasets. MNIST is a representative handwritten dataset, which includes digits from 0 to 9. It consists of 60,000 training and 10,000 test grayscale images, each of which is of size \(28 \times 28\). In contrast, Fashion-MNIST is composed of 10 different types of fashion images, such as sandals, t-shirts, bags, and pullovers. Similarly, it also consists of 60,000 training and 10,000 test samples, each of which is grayscale image of size \(28 \times 28\).

4.2. Model Configuration

In our presented approach, there exists a local model for the diversity adversarial training of the target model, a target model to be attacked, and a holdout model used by an attacker to perform a transfer attack. As a black-box attack, the attacker performs a transfer attack on the target model using adversarial examples created by the holdout model.
4.2.1. Local Model

The local model is a model known to the defenders of the system, which generates various adversarial examples. It is basically a convolutional neural network (CNN) [29], whose architecture and parameters are shown in Tables 1 and 2, respectively. The local model is trained with the original training data, which exhibits accuracies on the test images of MNIST and Fashion-MNIST as 99.43 and 92.13%, respectively.

Table 1. Local model architecture for MNIST and Fashion-MNIST. Conv. means convolutional layer. FC indicates fully connected layer.

| Layer Type       | Shape              |
|------------------|--------------------|
| Conv. with ReLU  | [3, 3, 32]         |
| Conv. with ReLU  | [3, 3, 32]         |
| Max pooling      | [2, 2]             |
| Conv. with ReLU  | [3, 3, 64]         |
| Conv. with ReLU  | [3, 3, 64]         |
| Max pooling      | [2, 2]             |
| FC with ReLU     | [200]              |
| FC with ReLU     | [200]              |
| Softmax          | [10]               |

4.2.2. Holdout Model

The holdout model is used by an attacker to generate an unknown adversarial example to carry out a black-box transfer attack. Its model structure and parameters are illustrated in Tables 2 and 3, respectively. After training with the original training samples, its accuracies on the test images of MNIST and Fashion-MNIST are 99.32 and 92.21%, respectively.

Table 2. Model parameters.

| Parameter       | MNIST     | Fashion-MNIST |
|-----------------|-----------|---------------|
| Optimizer       | Adam [31] | Adam          |
| Learning rate   | 0.1       | 0.1           |
| Momentum        | 0.9       | 0.85          |
| Delay rate      | -         | -             |
| Dropout [32]    | 0.5       | 0.5           |
| Batch size      | 128       | 128           |
| Epochs          | 20        | 20            |

Table 3. Holdout model architecture for MNIST and Fashion-MNIST. Conv. means convolutional layer. FC indicates fully connected layer.

| Layer Type       | Shape   |
|------------------|---------|
| Conv. + ReLU     | [3, 3, 128] |
| Conv. + ReLU     | [3, 3, 64]  |
| Max pooling      | [2, 2]    |
| Conv. + ReLU     | [3, 3, 64]  |
| Conv. + ReLU     | [3, 3, 64]  |
| Max pooling      | [2, 2]    |
| FC + ReLU        | [128]    |
| Softmax          | [10]     |
Table 4. Target model architecture for MNIST and Fashion-MNIST. Conv. means convolutional layer. FC indicates the fully connected layer.

| Layer Type          | Shape      |
|---------------------|------------|
| Conv. with ReLU     | [5, 5, 64] |
| Conv. with ReLU     | [5, 5, 64] |
| Max pooling         | [2, 2]     |
| Conv. with ReLU     | [5, 5, 64] |
| Max pooling         | [2, 2]     |
| FC with ReLU        | [128]      |
| Softmax             | [10]       |

4.2.3. Target Model

The target model is the model to be attacked, and is trained with various adversarial examples generated using a local model. By training with diverse adversarial samples, the target model becomes more robust against unknown adversarial attacks generated from the holdout model. Its model structure and parameters are shown in Tables 2 and 4, respectively. After training with the original training data, the accuracy of the target model on the test images of MNIST and Fashion-MNIST are 99.26 and 92.04%, respectively.

4.3. Generation on Adversarial Examples from Local Model

To generate various adversarial examples targeting the local model, each of the FGSM, I-FGSM, DeepFool, and CW methods generated 1000 adversarial samples, respectively. The reason 1000 adversarial examples were chosen randomly is that experimenting with 1000 adversarial examples yields reliable results. Many studies [33,34] related to adversarial examples have mostly evaluated 1000 adversarial examples. Subsequently, the target model is additionally trained with these obtained images.

5. Experimental Results

The attack success rate [35,36] refers to the rate at which the target model misclassifies the adversarial examples as target class chosen by the attacker. For example, if 97 out of 100 samples are misidentified by the target model as belonging to the class that the attacker wants them to be classified into, the attack success rate is 97%. The opposite of the attack success rate is the failure rate. Accuracy refers to the match rate of the target model between the input data and their true class labels.

Examples of adversarial images generated by various methods for the local model based on the MNIST dataset is illustrated in Figure 2. In the figure, each adversarial example has a different amount of noise added to the original sample. Specifically, FGSM and I-FGSM added more, while CW and DeepFool introduced relatively less noise to the original sample. As a result that CW and DeepFool generate noise optimized for the local model, they can generate adversarial examples with a smaller amount of noise than the original sample.
Examples of adversarial images generated by various adversarial example generation methods for the local model based on the Fashion-MNIST are presented in Figure 3. Similar to Figure 2, it can be observed from Figure 3 that some noise has been added to the original fashion images. Likewise, in this figure, CW and DeepFool generated adversarial examples with less noise added to the original sample than FGSM and I-FGSM.

The attack success rates for adversarial examples generated with the holdout model using the without, baseline, and diversity training methods based on MNIST dataset are shown in Figure 4. Here, the without method indicates a target model that does not use any adversarial training defense approach. The baseline method [24] refers to a method of training a target model by applying one adversarial training approach, such as FGSM. The diversity training method refers to a method of training the target model with adversarial examples generated by the FGSM, I-FGSM, DeepFool, and CW methods. Inspection of attack success rate in the figure reveals that the without model misrecognizes more than 89.9% of the adversarial examples. However, the diversity training method has reduced the attack success rate to less than 32.9%, and that the average performance is observed to be improved by more than 44.8% compared to the baseline method. In addition, the analysis of the failure rate is shown in Figure A1 of the Appendix A. Therefore, the diversity training method is more robust against adversarial attacks.
The attack success rates for adversarial examples generated with the holdout model using the without, baseline, and diversity training methods based on the Fashion-MNIST are demonstrated in Figure 5. Here, the without method means a target model that does not use any adversarial training defense approach. The baseline method [24] refers to a method of training a target model by applying one adversarial training approach, such as FGSM. The diversity training method refers to a method of training the target model with adversarial examples generated by the FGSM, I-FGSM, DeepFool, and CW methods. As shown in Figure 4, it can be seen that the without model misrecognizes more than 85.9% of the adversarial examples. However, the diversity training method is observed to reduce the attack success rate of adversarial examples to below 35.2%, and that the average performance is improved by more than 40.1% compared to the baseline method. In addition, the analysis of the failure rate is shown in Figure A2 of the Appendix A. Therefore, the diversity training method is more robust against adversarial attacks.

The accuracies of the without method, baseline method, and diversity training method on the test images of MNIST and Fashion-MNIST datasets are presented in Figure 6. Although trained with additional adversarial samples, the target model still maintains the accuracy obtained by the original data. In the figure, the diversity training method has almost the same accuracy as the without method and baseline method on the test data. Comparison between the accuracies of MNIST and Fashion-MNIST reveals that the accuracy obtained with Fashion-MNIST is lower due to the data characteristics.

In the experimental section, examples of adversarial samples generated by various methods were shown to demonstrate the performance of the diversity training method. In addition, an experimental analysis was carried out regarding the attack success rate and accuracy to determine the robustness against unknown adversarial example attacks when trained with various adversarial examples. Our demonstrated approach is an improved adversarial training method, whose performance was analyzed against the baseline method. The experimental result confirmed that the diversity training approach is more robust.
against unknown adversarial attacks than the existing adversarial training method. We believe our method can be adopted to deep neural network-based image recognition. However, the experimental analysis of our demonstrated method was limited only to MNIST and Fashion-MNIST datasets, and thus, evaluation with other datasets can serve as our future research topic.

Figure 6. The accuracy of original test data for the without method, baseline method, and diversity training method in MNIST and Fashion-MNIST.

6. Discussion

Assumption: The assumption of the diversity training method is that the attack is a black box, that is, the attacker does not have any information about the target model. In this study, the attacker generates various adversarial examples using the holdout model, which is known to the attacker, then transfers the black-box attack to the target model.

Generation of diverse adversarial examples: Unlike the conventional adversarial training method, the diversity training approach demonstrates a method of training the target model with various adversarial examples. It generates various adversarial samples targeting the local model known to the defender. Different adversarial examples are generated using the FGSM, I-FGSM, DeepFool, and CW methods, each of which has different degrees of distortion. Detailed comparison of FGSM, I-FGSM, DeepFool, and CW methods revealed that the CW method generates adversarial examples with the smallest distortion. The adversarial samples generated from these FGSM, I-FGSM, DeepFool, and CW methods are used to additionally train the target model. Through this process, a more robust model is obtained against an unknown adversarial attack.

Model configuration: In this paper, there exist three different types of models: target model, local model, and holdout model. These models are constructed with different architectures, as shown in Tables 1–3, respectively. In the local model, various adversarial examples are generated based on FGSM, I-FGSM, DeepFool, and CW methods from the local model known to the defender. The target model is the final inference model, which becomes a black-box model for the attacker. The target model is trained with various adversarial examples to become robust against unknown adversarial attacks. However, the attacker uses an adversarial example generated from the holdout model to make the target model misclassify, which is known as a transfer attack.

In terms of the optimizer, we used the Adam optimizer. After obtaining various adversarial examples, the Adam optimizer is used by the target model to learn these samples. The cross-entropy loss is used as our objective function. During loss minimization, the target model has a training process to try to classify various adversarial samples into correct classes. Instead of the Adam optimizer, the stochastic gradient descent (SGD) [37] or other optimization algorithms [38] can also be employed.
Dataset: Our experiments were conducted using the MNIST and Fashion-MNIST datasets, both of which contain grayscale images of size $28 \times 28$. However, there was a performance difference between MNIST and Fashion-MNIST. As a result of the characteristics of the Fashion-MNIST, the similarity between T-shirt and shirt images is higher than that of the numeric images, and thus, the test images of Fashion-MNIST tend to demonstrate lower accuracy than those of the MNIST. However, for both datasets, even after being trained with various adversarial samples, the diversity training method maintained almost the same accuracy as the model trained with the original training data.

Defense considerations: While a separate module for the target model is not required, the adversarial training approach is a simple and effective method to defend against adversarial attacks. Unlike existing adversarial training methods, the diversity training method is robust against unknown adversarial attacks by generating various adversarial samples and including them into the training data of the model. Regarding local models, our approach does not require a large number of such models because it generates several adversarial examples by using one local model in the same way as those existing adversarial learning methods. Regarding the generated adversarial examples, the attacker can produce various adversarial samples, and even under such conditions, the presented approach is observed to be robust on various adversarial attacks. In terms of accuracy obtained by training with the original data, the diversity training method maintains an accuracy similar to that of the existing adversarial training method.

Applications: One potential application field of the diversity training method is in autonomous vehicles. For an autonomous vehicle, the attacker intentionally deceives its classification mechanism to misclassify the road sign, which could be a modified adversarial example. To defend against such an adversarial attack, our demonstrated method can be used as a defense method to correctly identify the modified road sign. In addition, in the case of a medical business, there exists a risk of treatment misjudgment for a patient due to revised adversarial examples. Therefore, the diversity training method can be used in such systems to increase the correct classification rate of the medical treatment.

Limitation and future work In the diversity training method, the generation approaches used for adversarial samples include FGSM, I-FGSM, DeepFool, and CW methods. However, there exist some other methods to generate such samples. In addition, the diversity training method uses one local model similar to the basic adversarial example method. However, if the number of local models increases, research on various ensemble adversarial training by constructing several local models instead of a single one will be an interesting topic. In terms of the evaluation, the method of analyzing the distribution of the recognized adversarial training for the original sample and unknown adversarial example as a decision boundary will be an interesting topic.

7. Conclusions

In this paper, we demonstrate a diversity adversarial training method. In this method, adversarial examples were first generated using methods such as FGSM, I-FGSM, DeepFool, and CW, then used to train the target model to make it become more robust against an unknown adversarial attack. The experimental results revealed that our approach lowers the average attack success rates by 27.2 and 24.3% for various adversarial examples, while maintaining 98.7 and 91.5% accuracies for the original data of MNIST and Fashion-MNIST datasets.

Future research includes evaluating our demonstrated approach with other image datasets [39]. Moreover, training various adversarial examples using the generative adversarial network [40] algorithm will be an interesting topic.

Author Contributions: Conceptualization, H.K.; methodology, H.K.; software, H.K.; validation, H.K.; writing—original draft preparation, H.K.; writing—review and editing, H.K. and J.L.; visualization, H.K.; supervision, J.L. All authors have read and agreed to the published version of the manuscript.
Funding: This work was supported by 2021 (21-center-1) research fund of Korea Military Academy (Cyber Warfare Research Center) and was supported by Hwarang-Dae Research Institute of Korea Military Academy.

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

Appendix A

Table A1. Target model architecture for MNIST.

| Layer Type                  | MNIST Shape |
|-----------------------------|-------------|
| Convolution + ReLU          | [3, 3, 32]  |
| Convolution + ReLU          | [3, 3, 32]  |
| Max pooling                 | [2, 2]      |
| Convolution + ReLU          | [3, 3, 64]  |
| Convolution + ReLU          | [3, 3, 64]  |
| Max pooling                 | [2, 2]      |
| Fully connected + ReLU      | [200]       |
| Fully connected + ReLU      | [200]       |
| Softmax                     | [10]        |

Table A2. Target Model Parameters.

| Parameter     | MNIST Model | CIFAR10 Model       |
|---------------|-------------|---------------------|
| Learning rate | 0.1         | 0.1                 |
| Momentum      | 0.9         | 0.9                 |
| Delay rate    | -           | 10 (decay 0.0001)   |
| Dropout       | 0.5         | 0.5                 |
| Batch size    | 128         | 128                 |
| Epochs        | 50          | 200                 |
Table A3. Target model architecture [41] for CIFAR10.

| 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]   |
| Max pooling          | [2, 2]        |
| Convolution + ReLU   | [3, 3, 256]   |
| Convolution + ReLU   | [3, 3, 256]   |
| Max pooling          | [2, 2]        |
| Convolution + ReLU   | [3, 3, 512]   |
| Convolution + ReLU   | [3, 3, 512]   |
| Max pooling          | [2, 2]        |
| Convolution + ReLU   | [3, 3, 512]   |
| Convolution + ReLU   | [3, 3, 512]   |
| Max pooling          | [2, 2]        |
| Fully connected + ReLU | [4096]     |
| Fully connected + ReLU | [4096]     |
| Softmax              | [10]          |

Figure A1. For MNIST, the failure rate for adversarial examples generated with the holdout model using the without method, baseline method, and diversity training method.

Figure A2. For Fashion-MNIST, the failure rate for adversarial examples generated with the holdout model using the without method, baseline method, and diversity training method.
References

1. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* 2015, 61, 85–117.

2. Tanzi, L.; Vezzetti, E.; Moreno, R.; Aprato, A.; Audisio, A.; Massé, A. Hierarchical fracture classification of proximal femur X-ray images using a multistage Deep Learning approach. *Eur. J. Radiol.* 2020, 133, 109373.

3. El Asnaoui, K.; Chawki, Y. Using X-ray images and deep learning for automated detection of coronavirus disease. *J. Biomed. Struct. Dyn.* 2020, 1–12, doi:10.1080/07391102.2020.1767212.

4. Barreno, M.; Nelson, B.; Joseph, A.D.; Tygar, J. The security of machine learning. *Mach. Learn.* 2010, 81, 121–148.

5. Kwon, H.; Yoon, H.; Park, K.W. Selective Poisoning Attack on Deep Neural Networks. *Symmetry* 2019, 11, 892.

6. Kwon, H. Detecting Backdoor Attacks via Class Difference in Deep Neural Networks. *IEEE Access* 2020, 8, 191049–191056.

7. Kwon, H.; Yoon, H.; Park, K.W. Multi-targeted backdoor: Indentifying backdoor attack for multiple deep neural networks. *IEICE Trans. Inf. Syst.* 2020, 103, 883–887.

8. Kwon, H.; Lee, J. AdvGuard: Fortifying Deep Neural Networks against Optimized Adversarial Example Attack. *IEEE Access* 2020, doi:10.1109/ACCESS.2020.3042839.

9. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.

10. Fawzi, A.; Fawzi, O.; Frossard, P. Analysis of classifiers’ robustness to adversarial perturbations. *Mach. Learn.* 2018, 107, 481–508.

11. Shen, S.; Jin, G.; Gao, K.; Zhang, Y. Ape-gan: Adversarial perturbation elimination with gan. *arXiv* 2017, arXiv:1707.05474.

12. Meng, D.; Chen, H. Magnet: A two-pronged defense against adversarial examples. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 135–147.

13. Xu, W.; Evans, D.; Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. In Proceedings of the Network and Distributed System Security Symposium (NDSS 2018), San Diego, CA, USA, 18–21 February 2018.

14. Kwon, H.; Yoon, H.; Park, K.W. Acoustic-decay: Detection of adversarial examples through audio modification on speech recognition system. *Neurocomputing* 2020, 417, 357–370.

15. Kwon, H.; Yoon, H.; Park, K.W. POSTER: Detecting audio adversarial example through audio modification. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 2521–2523.

16. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 23–25 May 2016; pp. 582–597.

17. Kwon, H.; Kim, Y.; Park, K.W.; Yoon, H.; Choi, D. Advanced ensemble adversarial example on unknown deep neural network classifiers. *IEICE Trans. Inf. Syst.* 2018, 101, 2485–2500.

18. He, W.; Wei, J.; Chen, X.; Carlini, N.; Song, D. Adversarial example defense: Ensembles of weak defenses are not strong. In Proceedings of the 11th USENIX Workshop on Offensive Technologies (WOOT 17), Vancouver, BC, Canada, 14–15 August 2017.

19. Carlini, N.; Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017; pp. 3–14.

20. Goodfellow, I.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.

21. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv* 2016, arXiv:1607.02533.

22. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. Deepfool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2574–2582.

23. Papernot, P.; Papernot, N.; Celik, Z.B. Machine learning in adversarial settings. *IEEE Secur. Priv.* 2016, 14, 68–72.

24. LeCun, Y.; Cortes, C.; Burges, C.J. *MNIST Handwritten Digit Database*; AT&T Labs: Florham Park, NJ, USA, 2010; Volume 2. Available online: http://yann.Lecun.Com/exdb/mnist (accessed on 5 March 2021).

25. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* 2017, arXiv:1708.07747.

26. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* 1998, 86, 2278–2324.

27. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* 2018, arXiv:1803.08375.

28. Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, Saarbrucken, Germany, 21–24 March 2016; pp. 372–387.

29. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* 1998, 86, 2278–2324.
33. Athalye, A.; Engstrom, L.; Ilyas, A.; Kwok, K. Synthesizing robust adversarial examples. In Proceedings of the International Conference on Machine Learning, PMLR, Vienna, Austria, 25–31 July 2018; pp. 284–293.
34. Ilyas, A.; Engstrom, L.; Athalye, A.; Lin, J. Query-efficient black-box adversarial examples (superceded). arXiv 2017, arXiv:1712.07113.
35. Carlini, N.; Wagner, D. Towards evaluating the robustness of neural networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 39–57.
36. Xiao, C.; Li, B.; Zhu, J.Y.; He, W.; Liu, M.; Song, D. Generating adversarial examples with adversarial networks. arXiv 2018, arXiv:1801.02610.
37. Bottou, L. Stochastic gradient descent tricks. In Neural Networks: Tricks of the Trade; Springer: Berlin/Heidelberg, Germany, 2012; pp. 421–436.
38. Hamdia, K.M.; Zhuang, X.; Rabczuk, T. An efficient optimization approach for designing machine learning models based on genetic algorithm. Neural Comput. Appl. 2020, 1–11, doi:10.1007/s00521-020-05035-x.
39. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
40. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Advances in Neural Information Processing Systems; MIT Press: Cambridge, MA, USA, 2014; pp. 2672–2680.
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.