applied sciences

MDPI

Article

On the Effectiveness of Adversarial Training in Defending against Adversarial Example Attacks for Image Classification

Sanglee Park and Jungmin So *

Department of Computer Science and Engineering, Sogang University, Seoul 04107, Korea; lp0523@sogang.ac.kr
* Correspondence: jso1@sogang.ac.kr

Received: 18 September 2020; Accepted: 11 November 2020; Published: 14 November 2020

Abstract: State-of-the-art neural network models are actively used in various fields, but it is well-known that they are vulnerable to adversarial example attacks. Throughout the efforts to make the models robust against adversarial example attacks, it has been found to be a very difficult task. While many defense approaches were shown to be not effective, adversarial training remains as one of the promising methods. In adversarial training, the training data are augmented by "adversarial" samples generated using an attack algorithm. If the attacker uses a similar attack algorithm to generate adversarial examples, the adversarially trained network can be quite robust to the attack. However, there are numerous ways of creating adversarial examples, and the defender does not know what algorithm the attacker may use. A natural question is: Can we use adversarial training to train a model robust to multiple types of attack? Previous work have shown that, when a network is trained with adversarial examples generated from multiple attack methods, the network is still vulnerable to white-box attacks where the attacker has complete access to the model parameters. In this paper, we study this question in the context of black-box attacks, which can be a more realistic assumption for practical applications. Experiments with the MNIST dataset show that adversarially training a network with an attack method helps defending against that particular attack method, but has limited effect for other attack methods. In addition, even if the defender trains a network with multiple types of adversarial examples and the attacker attacks with one of the methods, the network could lose accuracy to the attack if the attacker uses a different data augmentation strategy on the target network. These results show that it is very difficult to make a robust network using adversarial training, even for black-box settings where the attacker has restricted information on the target network.

Keywords: image classification; adversarial example attacks; adversarial training; MNIST

1. Introduction

Deep neural networks (DNN) have recently been showing high performance for various applications such as computer vision [1,2], natural language processing [3] and speech recognition [4]. However, a major vulnerability of existing neural networks has been pointed out, which is called vulnerability against adversarial examples . For an image classification task, an adversarial example is an image that is slightly modified from the original image, with the purpose of fooling a classifier network [5]. While the distorted image looks very similar to the original image and looks natural to the human eye, the classifier network misclassifies the image and outputs a wrong result. There are many different methods proposed for creating an adversarial image from an original image, such as Fast Gradient Sign Method (FGSM) [6], Carlini and Wagner Attack (C&W) [7], Deepfool [8] and Jacobian Saliency Map Attack (JSMA) [9].

Since the vulnerability has been found, researchers have put a large effort to make the neural networks robust to adversarial example attacks. Various solutions have been proposed, but many of them were found to be less powerful because new attack methods or small modifications to existing attack methods could circumvent the defense mechanisms and succeed in fooling the networks [7]. Among defense methods, *adversarial training* is a method that has shown some good results in terms of robustness. Specifically, Madry et al. used adversarial training to train a network that is mostly robust to adversarial images where the perturbations are L_∞ norm bounded by $\epsilon = 0.3$ [10]. However, to increase robustness by adversarial training, we need a large number of training data [11]. In addition, the network is still vulnerable to other attacks that uses a different distance metric such as L_2 norm or L_0 norm [12].

Since there are plenty of methods that create adversarial examples, a natural question is whether we can use adversarial training to train a network so that it is robust to multiple types of adversarial examples, such as ones created using L_0-, L_1-, L_2- and L_∞-perturbation. Tramer et al. [13] tried to answer this question in the context of a white-box attack, where the attacker has complete access to the parameters of the trained model. They found that different types of adversarial examples, such as examples created using L_∞- and L_1-perturbation, are mutually exclusive, which means adversarially training with one type of adversarial examples does not help defending against the other types of adversarial examples. Moreover, adversarially training the model with one type of adversarial examples may increase vulnerability to the other type of adversarial examples.

In this paper, we try to answer the question in the context of a black-box attack, in which the attacker does not have access to the parameters of the trained model. In contrast to a white-box attack, an attacker performing a black-box attack cannot generate optimized adversarial examples to fool the target network. We also assume that the attacker cannot reverse engineer the model by repeatedly testing the model with his own inputs [14]. In many real-world settings, a black-box attack is the only choice for the attacker, because the service provider does not open the model parameters to the public. In addition, repeatedly testing the model would incur time and monetary cost [15]. Under these assumptions, the attacker trains his own target network and uses the target network to apply an attack algorithm and create adversarial examples. These examples are fed into the defender's network, hoping that they are misclassified by the target network.

To find out if adversarial training is effective in defending against multiple types of adversarial examples, we train a neural network model using adversarial examples created from various well-known attack methods. After obtaining various networks trained using different adversarial images, we perform black-box attacks on the networks with various attack methods and measure how well the network classifies the perturbed images. We use the MNIST handwritten digit classification dataset [16] as the target and train neural networks with well-known structures such as Lenet5 [16] and ResNet [2]. Each network is trained with and without data augmentation such as rotation and translation, in order to see the effect of using data augmentation on the adversarial robustness. First, the experiment results show that, while adversarially training a network with a certain attack method could defend well against that particular attack method, the effect is limited for other attack methods. For example, a network trained with FGSM samples could successfully classify FGSM adversarial examples, but its performance on other adversarial examples such as JSMA and C&W examples are limited. Second, even if the network is trained with multiple types of adversarial examples, the network could lose accuracy to the attack if the attacker uses a different data augmentation strategy on the target network. For example, a network trained using FGSM, JSMA and C&W samples can be vulnerable to FGSM attacks, if the defender used data augmentation in training the network while the attacker did not use data augmentation. The results show that it is very difficult to train a robust network using adversarial training, because not only the attack methods but also other training configurations such as data augmentation strategy can affect the robustness of the network against black-box attacks.

2. Preliminaries

2.1. Creating Adversarial Examples

Although it is possible to create an adversarial example from scratch, a typical way is to take an original sample and modify the sample so that it could be misclassified by the target network. Figure 1 shows images from the MNIST dataset plus adversarial examples created by three different methods. We can observe from the images that the three methods, while they have the same goal, creates different images.

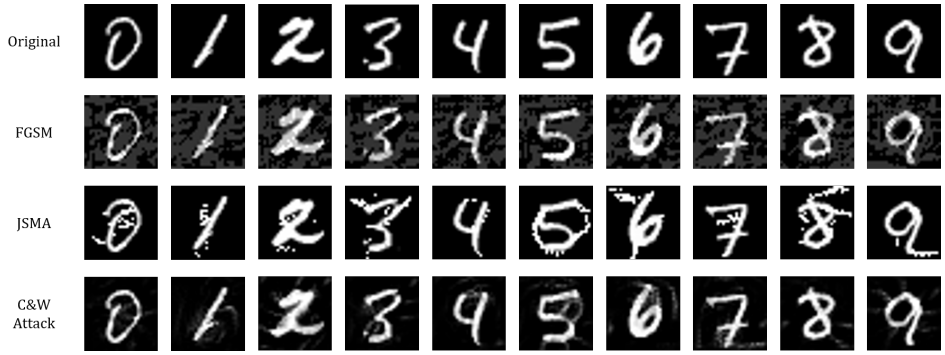

Figure 1. Image of original MNIST and adversarial attacked examples generated by FGSM, JSMA and C&W L_2 attacks.

A "good" adversarial example of an image is the one that looks very similar to the original image while being misclassified by the network. While there are many different metrics to measure image similarity, norm-based distance metrics are frequently used. For example, FGSM [6] uses the L_∞ distance by limiting the amount of noise added to each input pixel. On the other hand, JSMA [9] uses the L_0 distance by limiting the number of pixels that can be modified. Carlini et al. [7] generalized the problem of creating adversarial examples, stating that the attacker tries to minimize the L_p distance between the original image x and the adversarial image x', while x' is misclassified by the target network. The L_p distance can be defined as Equation (1), where p is a non-negative number and n is the dimension of vector v.

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}} \tag{1}$$

When using L_p distance, we can define p to be zero or infinity. The L_0 distance is the same as number of dimensions where the value is different between x and x'. The L_∞ distance, on the other hand, is the maximum of distances between each dimension of x and x'. For MNIST images, each dimension of input corresponds to a pixel. Thus, L_0 distance between two images is the number of pixels where the color is different, whereas L_∞ distance between two images is the maximum of color differences among pixels of x and x'. For other p values, we can calculate the L_p distance by calculating $\|x - x'\|_p$. Among different values of p, the L_2 distance which corresponds to the Euclidean distance is the most widely used distance metric. In the paper, we ch0ose three representative methods for creating adversarial examples: FGSM, JSMA and C&W L_2. These methods use L_∞, L_0 and L_2 distance when creating adversarial examples, respectively. While many other attack methods also use these distance metrics, it should be noted that there are methods that use other p values such as L_1 distance [17].

Suppose we are allowed to modify an original image x to an adversarial image x' so that the distance between the two is less than some threshold under a particular distance metric. Among the infinite possible ways of modifying x under the given condition, we should choose the direction that will give us the best chance that x' will be misclassified. If the attacker knows the architecture of the target network and has access to its parameters, he can calculate the gradient of a loss function

(such as cross-entropy between the prediction and the true class) with respect to the input. Using the gradient, the attacker can modify x in the direction that will increase the loss function. This is called a *white-box* attack, meaning that the information of the target network is open to the attacker. However, often times the attacker does not have information on the architecture of the neural network or its parameters. Attacking the network without this information is called a *black-box* attack. One can also define a *gray-box* attack when the attacker has partial information on the target network. For example, the attacker may know the architecture of the target network, but not its parameters. In a black-box or gray-box attack, the attacker cannot calculate gradient of the target network with respect to an input. Thus, the attacker has to use another model as a substitute based on which the adversarial examples are created. In this case, the attacker may just use a different network to create adversarial examples, or try to make the substitute network close to the actual target network by extracting information from the target network. It is shown that, although the power of attack is weakened, the black-box attack can still generate adversarial examples that fool the target network [18].

Depending the purpose of attack, the attacker may want to modify the original input so that it is misclassified as a specific class. For example, the attacker may want to change the number '3' slightly so that it is recognized as '8' by the target network. This is called a targeted attack. On the other hand, if the goal of the attacker is to have the target network produce a wrong result on an input and does not care about what the result is, then it is an untargeted attack. Intuitively, the required amount of modification is larger for a targeted attack compared to an untargeted attack.

2.2. Attack Algorithms

2.2.1. Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) [6] is an algorithm that uses L_∞ distance to create an adversarial example. FGSM modifies all pixels in the input image by a given amount. The amount of modification for each pixel is determined in the way that the value of loss function is increased. Specifically, the adversarial example is created using Equation (2). In the equation, x is the input data and y is its label, which indicates the true class of x. J is the loss function and θ is the set of parameters in the target neural network. Since the loss function basically calculates the difference between the true class y and the predicted class of x using parameters θ, choosing x' that increases J will also increase the chance that x' is misclassified by the network.

$$x' = x + \epsilon \cdot sign\left(\bigtriangledown_x J_\theta\left(x, y\right)\right) \tag{2}$$

FGSM first calculates the gradient of J with respect to the input x, and then modifies each dimension of the input based on the sign of the gradient. For example, if the gradient is positive for some pixel, its value is increased. If the gradient is negative, its value is decreased. The amount of modification is determined by the parameter ϵ. A high ϵ will increase the chance of misclassification, but the resulting image will look very different from the original image. A low ϵ will produce an output image that is very similar to the input image, but the chance of misclassification becomes low as well.

The second row of Figure 1 shows the images created using FGSM. We can observe that the background pixels that used to be black are now somewhat gray, and some parts of the digit that used to be white have also turned gray. There are other attack algorithms that use the same approach as FGSM, such as Basic Iterative Method (BIM) [19], Projected Gradient Descent (PGD) [10] and Momentum Iterative Attack [20]. Instead of changing the input image once, these methods modify the input multiple times iteratively, so that the distance between the input image and the adversarial image is reduced.

2.2.2. Jacobian Saliency Map Attack

The Jacobian Saliency Map Attack (JSMA) [9] perturbs the input image using L_0 distance metric. Among pixels in the input image, JSMA iteratively selects the pixel that can will increase the chance of misclassification by the highest amount. To do that, the attacker first calculates gradient of the output layer neurons with respect to each pixel of the input image. The result is the Jacobian matrix that is shown in Equation (3). In the equation, $l(x)$ is the values of output layer neurons before applying softmax, also called *logits*. i is the index of the input pixels and j is the index of the logits.

$$\bigtriangledown l\left(x\right) = \frac{\partial l\left(x\right)}{\partial x} = \left[\frac{\partial l_j\left(x\right)}{\partial x_i}\right]_{i \times j} \tag{3}$$

If the attack is a targeted attack, JSMA selects i which has the maximum absolute value among the elements in the column j of the Jacobian matrix, where j is the target class. Then, the pixel is changed in the direction that will decrease $l_j(x)$. Once the pixel is changed, a new Jacobian matrix is calculated with respect to the modified input. This process is repeated until the adversarial example x' is misclassified as the target class by the target network, or we reach the maximum number of pixels allowed to be modified. For an untargeted attack, a target class can be selected randomly or the target class which requires the minimum amount of modified pixels can be searched. The third row of Figure 1 shows the images created using JSMA. In contrast with FGSM, images created using JSMA has dots that were not in the original input while most of the background stays black.

2.2.3. Carlini and Wagner Attack

The Carlini and Wagner (C&W) attack [7] solves the optimization problem in Equation (4) to obtain the change applied to the input image. In the equation, $D(x, x+\delta)$ is the distance between the original input x and the adversarial image $x+\delta$. $C(x+\delta)$ is the class determined by the target network and t is the target class. In the problem, each pixel of an image has a value in the range $[0, 1]$. Thus, the pixel values of the adversarial image should also fall into this range. Therefore, we have the condition $x + \delta \in [0,1]^n$.

$$\begin{aligned} \text{minimize} \quad & D(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x + \delta \in [0,1]^n \end{aligned} \tag{4}$$

The optimization cannot be directly solved because of the constraint $C(x+\delta) = t$. Thus, the authors substituted C using an approximation function f, which is designed so that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$. One example of f is Equation (5), where Z is the output of the network for input x', before applying the final softmax function.

$$f(x') = (\max_{i \neq t}(Z(x')_i) - Z(x')t)^+ \tag{5}$$

In the optimization problem, the distance metric D could be any metric including L_0, L_2 and L_∞ distance. The final row of Figure 1 shows the images created using C&W with L_2 distance as the distance metric. Compared to images created by FGSM and JSMA, the images look more similar to the original inputs, with some light strokes added near the digit. The C&W attack is shown to be effective in creating adversarial images that looks very similar to original inputs [21,22] and is successful at attacking networks armed with defensive methods such as defensive distillation [23].

2.3. Defensive Schemes

2.3.1. Adversarial Training

Adversarial examples often have features that are not seen in the original training samples. For example, in Figure 1, the cloudy background in samples created using FGSM is something that cannot be seen in the training set images. Because of that, the neural network trained from clean images can be misled by the unseen features. A natural counter measure is to include adversarial images in the training stage. Goodfellow et al. [6] created adversarial samples from training images using FGSM and included both clean and adversarial images when training the network. The adversarially trained network could achieve much higher accuracy compared to a normal network for adversarial samples created from the test set.

Madry et al. [10] used PGD, an iterative and stronger version of FGSM, for adversarial training. This scheme is found to be effective for images created from similar attack algorithms. Specifically, the network can accurately classify most of adversarial samples, if the L_∞ distance between the original and the adversarial pair is bounded by some parameter ϵ. However, Schott et al. [12] showed that, while the adversarially trained network is resistant to attacks using L_∞ distance, it is still vulnerable to other adversarial samples created using other distance metrics.

2.3.2. Test-Time Processing

While adversarial training tries to make the neural network robust at the training stage, the test-time processing does not change the training data and process. Instead, the input data or the model is manipulated at test-time to mitigate the effect of adversarial perturbation. In Figure 1, we can observe that in the adversarial images, noise is added either at the background or near the digit. Thus, an intuitive defense scheme is to try to remove the noise, by applying blurring or quantization [24]. Randomly transforming the input image could also help [25]. Especially for white-box attacks, it is possible that an adversarial sample is crafted so that it barely crosses the class boundary in the neural network. Random transformation could bring back the image to its original class. Generative Adversarial Networks (GANs) and autoencoders can be used in transforming the inputs. For example, a GAN is first trained to learn the distribution of clean images. At test time, a sample close to the image is searched in the learned distribution. Then, instead of the input, the image in the learned distribution is input to the network [26].

Instead of manipulating the input image, it is also possible to process the neural network at test time. For example, some neurons from the intermediate layers of the network can be randomly pruned [27] or random noises can be added to the output of intermediate layers [28]. When adding randomness to the network, a typical technique is to combine results from an ensemble of multiple networks in order to normalize the effect of randomness. By applying randomization, we can investigate multiple points in the neighborhood of the input, instead of a single point. It may be helpful in defending against adversarial images, because the attacker now has to add more perturbation to the original input so that the adversarial sample and its random neighbors are all misclassified.

2.3.3. Adversarial Example Detection

A robust classifier is expected to accurately classify images with adversarial perturbations. Building a robust classifier is a very challenging task, because there are so many ways of creating adversarial examples. Depending on the application, it may be sufficient to detect whether an input is suspected to be an adversarial sample. For example, it is possible to train a neural network that does binary classification on clean and adversarial examples [29,30]. This network could be used at test time along with the classifier network to determine whether an input is an adversarial example. Instead of training an additional network, it is also possible to add an additional class for the outliers in the classifier network [31].

Instead of training a network for adversarial example detection, other methods try to find characteristics of adversarial samples distinct from clean samples. Xu et al. [24] compared network output for an input image and a feature squeezed version of the image. If their outputs are significantly different, it is a sign that the image is an adversarial sample. Feinman et al. [32] showed that the uncertainty of adversarial examples is higher than clean samples. Hendrycks et al. [33] pointed out that adversarial examples have different coefficients in the result of principal coefficient analysis (PCA). Song et al. [34] detected adversarial examples by comparing with the distribution of training data. Once an adversarial example is detected, it is often rejected from the system or notified to the (human) operator to make the final decision.

3. Effect of Adversarial Training on Network Performance

3.1. Attack Model and Experiment Setup

Suppose an attacker tries to create adversarial examples that fools a neural network classifier, for malicious purposes such as bypassing authentication or detection. In practical settings, it is often not possible for an attacker to lay hands on the network parameters. In addition, it may be difficult for the attacker to test the target network by trying out lots of input data and observing outcome of the network. Thus, a more "practical" attack is a black-box (or gray-box) attack which assumes the attacker has no knowledge on the parameters of the target network. The attack model we consider in this paper is the case when the attacker exactly or approximately knows the architecture of the network, but does not have information on the parameters. To attack a target network, the attacker first trains his own network that has the same (or similar) structure as the target network. Then, the attacker creates adversarial examples based on the network, using an attack algorithm of choice.

Under this attack model, we study the effect of adversarial training on defending against the attack. We use the MNIST handwritten digit dataset [16], which is a simple and one of the most widely used datasets for testing performance of machine learning techniques. We use two well-known neural network models, LeNet5 [16] and ResNet [2], for the target networks as well as the networks used by the attacker to create adversarial examples. For attack algorithms, we use the ones explained in the previous section: FGSM, JSMA and C&W with L_2 distance as the metric. These three algorithms represent the attack algorithms that use L_∞, L_0 and L_2 distance for adversarial sample generation.

For each type of neural network models, we train multiple networks with different input data. One of the network is trained with only the clean data from the training set, and the other networks are trained with adversarial samples along with the clean samples. For example, to adversarially train a network with FGSM, we augment the training set with FGSM samples created from each clean data in the training set. Thus, the number of samples in the training set becomes twice the original set. Table 1 shows the parameters used for the attack algorithms. In FGSM, ϵ is the maximum difference between the original sample and its adversarial pair for each pixel. In JSMA, θ is a parameter similar to ϵ in FGSM and γ is the maximum percentage of pixels that can be modified from the original sample. In C&W, a confidence level can be adjusted which determines amount of modification. Setting a high confidence value will result in a high probability of getting the adversarial sample misclassified, but the resulting image will look more distorted to the human eye. The images in Figure 1 were created using the parameters in Table 1. For both LeNet5 and ResNet models, we use the Adam optimizer [35] with learning rate 0.0001, and the training is done after 50 epochs for LeNet5 and 100 epochs for ResNet. The time taken for generating adversarial examples is also shown in Table 1, which was measured using a computer with Intel Core i7-9700F CPU, 16GB memory and a single Nvidia GeForce RTX 2080Ti GPU. For FGSM, it takes approximately 1 min to create adversarial samples from the whole training set. On the other hand, C&W takes about 13.7 h to generate FGSM adversarial examples for a ResNet model.

Table 1. Parameters for the attack algorithms and time for generating adversarial examples.

Attack Algorithm	Parameters	Time for Generating 60,000 Adversarial Samples (s)	
		LeNet5	ResNet
FGSM (L_∞)	$\epsilon = 0.2$	48	66
JSMA (L_0)	$\theta = 1.0$, $\gamma = 0.2$	2226	9180
C&W (L_2)	confidence: 5	8142	49,542

When training, it is often a common practice to augment training data with random transformations to compensate for the small number of data. We also augment data with using random rotation, random translation and random erasing, as shown in Figure 2. Specifically, at each epoch, an image in the training set is randomly rotated with a degree in the range of $[-20, 20]$ and also randomly shifted in four directions by a maximum of 6 pixels (approximately 20% of width and height). For random erasing, a rectangular region of random size up to 30% of the pixels is probabilistically erased. When doing adversarial training, we only apply rotation and translation to clean images and not the adversarial examples.

Figure 2. Random translation, random rotation and random erasing applied to a sample in the training set.

3.2. Performance of an Adversarially Trained LeNet5 Network

Figure 3 shows the accuracy of the test set while the LeNet5 network is being trained. The network in Figure 3a is trained with only the clean images, while the networks in Figure 3b–d are trained with clean images plus adversarial images generated from FGSM, JSMA and C&W, respectively. The lines in each graph indicate the test sets. 'Clean' means the test set consist of original images from the MNIST test set. 'FGSM', 'JSMA' and 'C&W' means the test set consists of adversarial examples created from the original test set samples using FGSM, JSMA and C&W attack, respectively.

In Figure 3a, we can see that the test accuracy of FGSM, JSMA and C&W set is much lower than the clean set, achieving between 40% and 60% accuracy. This means that the attack is successful roughly half of the time, even when the attacker does not know the parameters of the target network. Although the exact accuracy will depend on what parameters the attacker uses to create adversarial examples, this result shows that it is quite easy to break down the performance of a neural network even with limited information. Figure 3b shows the test accuracy when the network is trained with FGSM samples along with their clean pairs. We can see that the network accurately classifies most of FGSM adversarial samples, achieving an accuracy comparable to the clean samples. This means that the network has successfully learned the distribution of images created by FGSM. On the other hand, the accuracy of JSMA has not improved, suggesting that the JSMA images have a different distribution with original and FGSM images. For C&W, adversarial training with FGSM was beneficial, showing a 20% improvement in the accuracy. This result suggests the features of FGSM images have some correlation with features of C&W images, while there is little correlation between FGSM and JSMA images. In other words, perturbing samples using L_∞ distance generates images closer to those generated using L_2 distance, compared to those generated using L_0 distance. A similar observation can be made in Figure 3c as well, which shows performance of the network trained using JSMA samples along with their original pairs. Training with JSMA is most helpful in classifying JSMA adversarial examples, while the accuracy improvements are limited for FGSM and C&W samples. Figure 3d shows

performance of the network trained using C&W samples. The network successfully classifies C&W adversarial samples, whereas achieves 60–70% accuracy on FGSM and JSMA samples.

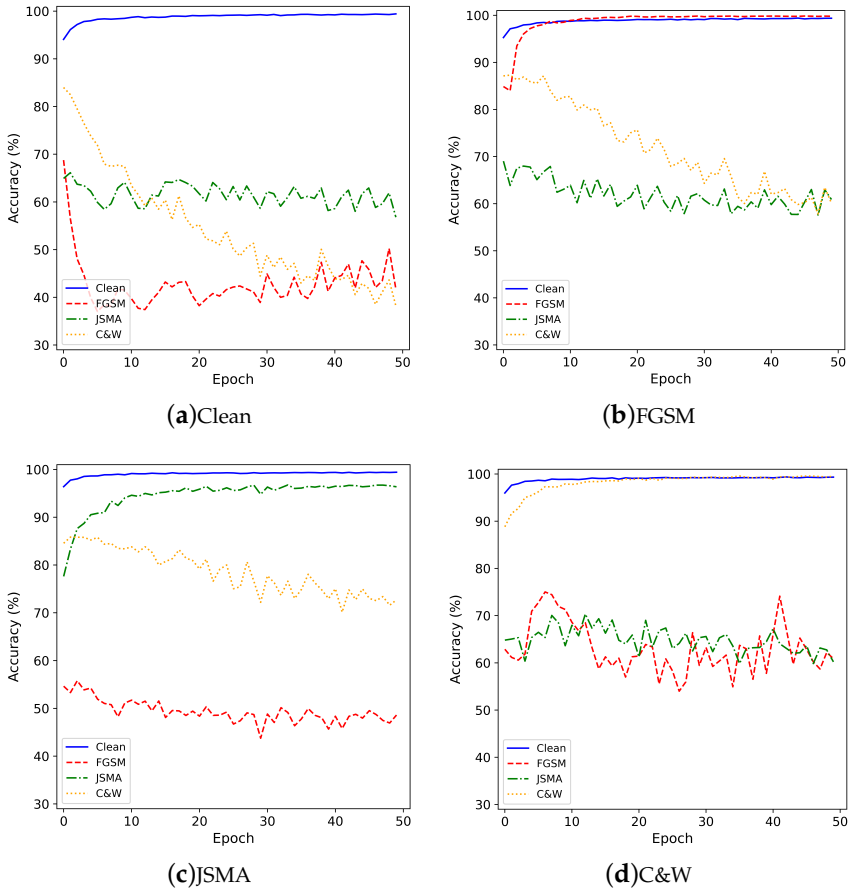

Figure 3. Accuracy of adversarially trained LeNet5 network on the test set with adversarial examples. Each models are trained with: (**a**) Clean; (**b**) FGSM; (**c**) JSMA; and (**d**) C&W.

Table 2 shows the test accuracy of LeNet5 networks trained using different combinations of clean and adversarial examples. For each set of test samples, the medium and maximum accuracy obtained between 20 and 50 epochs is shown in the table. For example, for a network trained using only the clean images, the maximum accuracy achieved on clean test images is 99.42%, while the maximum accuracy of FGSM, JSMA and C&W images is 50.23%, 64.07% and 55.28%, respectively. We can see that, without adversarial training, the network can be easily attacked by a variety of attack methods, with its accuracy considerably lowered. When the network is adversarially trained with a single set of adversarial examples, it was able to accurately classify adversarial example created using the same method, but not for other types of adversarial samples. From this observation, a natural insight is to augment the training data with multiple types of adversarial examples. In the table, we measured accuracy of networks trained with multiple set of adversarial images. When the network is trained with two sets of adversarial examples (such as FGSM and C&W), it is robust to the test images created from these two methods, but not robust for adversarial examples generated from other methods (such as JSMA). Finally, the network trained with clean, FGSM, JSMA and C&W samples achieves high accuracy of all three types of adversarial examples, while maintaining high accuracy of the clean samples. At the cost of creating more adversarial examples and training with more data, the network can achieve higher robustness.

Table 2. Test accuracy of LeNet5 measured between 20 and 50 epochs. The network is trained with data augmentation including random rotation, random translation and random erasing.

		Maximum				Median			
		Clean	**FGSM**	**JSMA**	**C&W**	**Clean**	**FGSM**	**JSMA**	**C&W**
Trained model	Clean	99.42	50.23	64.07	55.28	99.23	42.01	61.06	46.07
	FGSM	99.40	99.31	63.90	75.67	99.24	99.27	60.34	65.08
	JSMA	99.42	50.35	96.75	81.20	99.30	48.46	96.35	75.02
	C&W	99.37	74.12	69.00	99.57	99.22	61.17	63.30	99.31
	FGSM + JSMA	99.40	99.32	96.29	90.05	99.28	99.26	95.15	84.94
	FGSM + C&W	99.43	99.32	65.27	99.65	99.24	99.26	62.96	99.42
	JSMA + C&W	99.43	73.98	96.70	98.99	99.35	70.71	95.83	98.58
	FGSM + JSMA + C&W	99.36	99.30	95.82	98.81	99.27	99.21	94.92	98.39

3.3. Performance of an Adversarially Trained ResNet18 Network

Since the result can be dependent on the neural network model, we conducted the same experiment on the ResNet18 network. While the LeNet5 model has 648,226 trainable parameters, the ResNet18 model is a much larger size with 11,175,370 parameters. In general, the ResNet18 model is known to perform better in learning complex features from images. In this experiment, both the defender and the attacker create adversarial examples from their own ResNet18 networks, and the same data augmentation strategy is applied as in the LeNet5 case. Since ResNet18 takes more epochs to converge compared to LeNet5, the network is trained for 100 epochs and the result is taken from test accuracy measured between 40 to 100 epochs of training. The result is in Table 3. We can observe similar patterns observed in the LeNet5 result. When an undefended network is tested using FGSM, JSMA and C&W samples, the accuracy is significantly lower compared to testing with clean images. When the network is trained with adversarial samples created using a particular algorithm, the network becomes robust to the adversarial examples generated using the same method. Multiple combinations of adversarial samples can be used for adversarial training, which provides robustness to more adversarial samples while maintaining the accuracy for clean test data. Compared with LeNet5, the ResNet18 model achieves higher robustness when the network is undefended. For example, the accuracy of FGSM samples tested on the LeNet5 model is approximately 50%, while the result on the ResNet18 is 70%. The accuracy of adversarial examples is also increased for other methods as well. Thus, we can say that, when adversarial training is not used, model structure and number of parameters could affect robustness of the network. However, when adversarial training is used, the accuracy of adversarial examples is comparable for LeNet5 and ResNet18 networks. In many cases, adversarial training is more effective than the model architecture. For example, when a network is adversarially trained with FGSM images, LeNet5 achieves higher accuracy than ResNet18 in classifying FGSM adversarial samples.

Table 3. Test accuracy of ResNet18 measured between 40 and 100 epochs. The network is trained with data augmentation including random rotation, random translation and random erasing.

		Maximum				Median			
		Clean	**FGSM**	**JSMA**	**C&W**	**Clean**	**FGSM**	**JSMA**	**C&W**
Trained model	Clean	99.68	70.69	80.70	93.27	99.50	65.30	76.82	91.54
	FGSM	99.65	95.92	78.40	97.51	99.56	92.24	73.53	96.69
	JSMA	99.66	78.71	97.78	96.13	99.56	73.85	96.80	95.42
	C&W	99.62	88.99	81.37	98.60	99.52	85.71	76.29	98.11
	FGSM + JSMA	99.65	95.34	96.34	98.05	99.54	93.80	95.34	97.53
	FGSM + C&W	99.67	97.54	77.87	98.79	99.57	96.59	75.14	98.38
	JSMA + C&W	99.69	89.80	97.00	98.63	99.56	86.57	95.57	98.25
	FGSM + JSMA + C&W	99.63	97.57	96.68	98.84	99.51	96.58	95.37	98.44

4. Impact of Data Augmentation on Robustness against Adversarial Examples

In the previous section, the experiment results show that a neural network model trained using multiple types of adversarial examples could achieve high accuracy on test samples generated using each of the attack methods. Although not tested for all existing attacks, at least for the three attacks a network could defend itself to a particular attack by adversarially training with a similar type of attack (such as using the same L_p-perturbation) in a black-box setting. In the experiments, the defender and the attacker use the same data augmentation techniques when training their networks, which are random rotation, random translation and random erasing. Since the defender and the attacker do not know what augmentation strategy the opponent will use, it is probable that they use a different augmentation strategy when training their respective networks. For example, the defender may use data augmentation when training the network, while the attacker generates adversarial examples based on a network trained without data augmentation.

In this section, we study the impact of data augmentation on robustness of adversarially trained networks. Specifically, we try to see if a mismatch in data augmentation strategy could make the network vulnerable, similar to the case where a mismatch in the attack method could make the network vulnerable to attacks. The result in the previous section was obtained for the case where both the defender and the attacker use data augmentation to train their networks. In this section, we first describe the result when both the defender and the attacker do not use data augmentation, and then discuss the result when the defender and the attacker use different strategies in data augmentation.

4.1. Performance of a Network Trained without Data Augmentation

We conducted an experiment where both the defender and the attacker train LeNet5 networks without data augmentation and create adversarial examples from the networks. Once generating adversarial examples from the training set, the defender trains LeNet5 again with original samples and the adversarial samples, also without data augmentation. The result is shown in Table 4. In the perspective of effectiveness of adversarial training, similar patterns can be observed with other experiments: adversarial training is effective in classifying adversarial samples created from the same method as samples used in training. Similar results can be observed if the defender and the attacker use the same strategy, regardless of whether they use or do not use data augmentation.

Another interesting observation when comparing Tables 2 and 4 is the trade-off between accuracy of the clean test set and accuracy of the adversarial examples. Without data augmentation, a LeNet5 network achieves 99.16% accuracy of the clean test set. When data augmentation is used, the accuracy can be improved to 99.42%. However, an undefended network trained without data augmentation achieves much higher accuracy for adversarial examples for all three methods.

Table 4. Test accuracy of LeNet5 measured between 20 and 50 epochs. The network is trained without data augmentation techniques such as random rotation and random translation.

		Maximum				Median			
		Clean	**FGSM**	**JSMA**	**C&W**	**Clean**	**FGSM**	**JSMA**	**C&W**
Trained model	Clean	99.16	66.58	70.03	88.22	99.02	63.27	66.99	83.02
	FGSM	99.11	99.11	74.41	95.66	99.00	99.01	73.05	94.19
	JSMA	99.15	72.97	98.24	95.80	99.03	69.45	98.09	94.06
	C&W	99.19	85.58	75.11	99.38	99.08	79.18	72.93	99.21
	FGSM + JSMA	99.16	99.00	98.15	98.02	99.05	98.89	97.88	97.52
	FGSM + C&W	99.14	99.12	78.55	99.37	99.01	99.03	77.06	99.16
	JSMA + C&W	99.25	86.68	98.36	99.23	99.15	81.08	98.12	99.03
	FGSM + JSMA + C&W	99.20	99.03	98.17	99.22	99.10	98.94	97.82	99.09

Performance comparison of the LeNet5 network with and without data augmentation is shown in Figure 4. In Figure 4a, we can see that the accuracy is slightly improved for the clean test set, when data augmentation is used. However, for Figure 4b–d, the performance of LeNet network trained with data augmentation achieves significantly lower accuracy on adversarial examples compared to a network trained without data augmentation. It is well known that data augmentation improves classification performance of a network by increasing generalization capacity [36], and thus data augmentation is often heavily used for most of the tasks. However, it is possible that data augmentation strategies such as rotation and translation could increase vulnerability of the network against adversarial examples. Although not covered in this paper, we believe this trade-off is an important problem that needs to be looked into in the future.

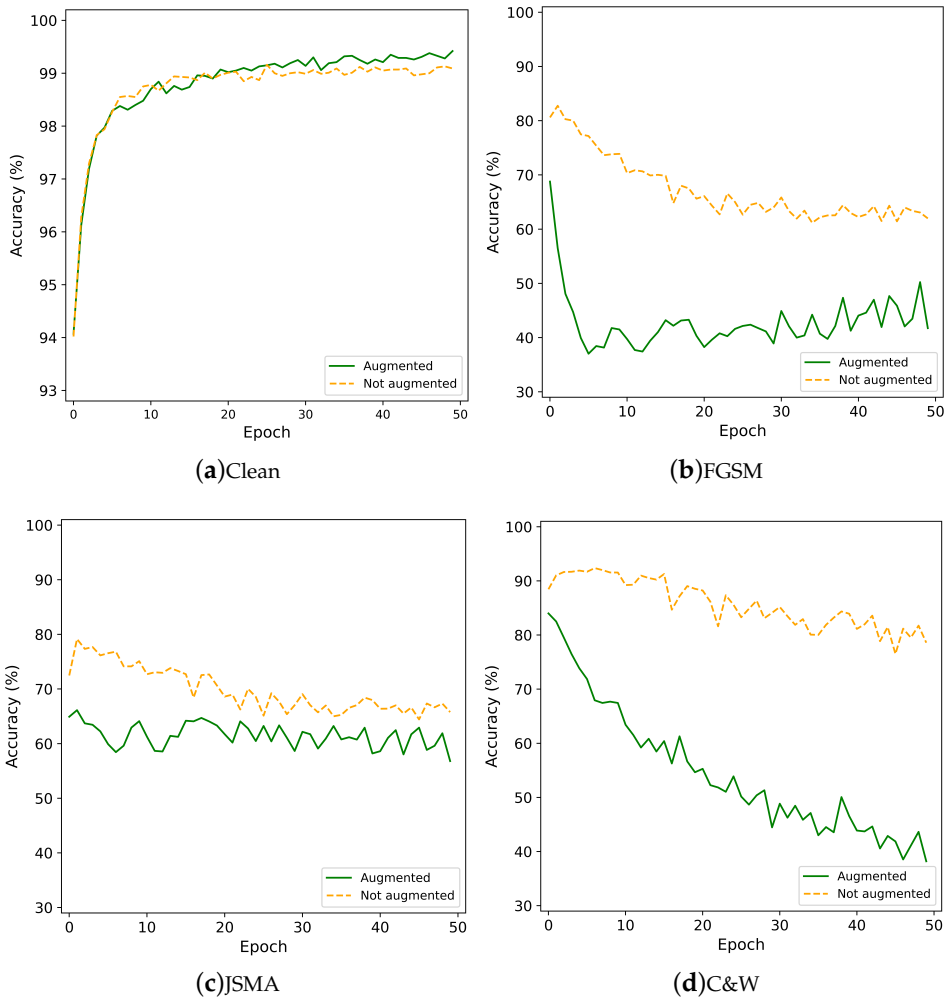

Figure 4. Test accuracy of LeNet5 model on each adversarial example attack. Each model is trained with: (**a**) Clean; (**b**) FGSM; (**c**) JSMA; and (**d**) C&W.

4.2. Robustness of Adversarially Trained Network with Mismatching Data Augmentation Strategy

In the previous experiments, the same augmentation strategy was used by the attacker and the defender. If the defender obtained adversarial samples for training from a network trained with data augmentation, the attacker also created adversarial examples using a network trained with the same data augmentation strategy. On the other hand, if the defender trained a network without data augmentation, the attacker also used a network trained without data augmentation as the target network for the black-box attack. In such cases, the adversarial training was successful defending against a particular attack algorithm. However, what if the attacker and the defender used a different

augmentation strategy? Consider the case where the defender and the attacker both train a LeNet5 model and also use the same attack algorithm such as FGSM to create adversarial examples. However, the defender's network is trained with data augmentation while the attacker's network is not. Will the adversarial training still be successful against the attack? To answer the question, we conducted an experiment for cases where the defender uses data augmentation and the attacker does not, and vice versa. The result is shown in Table 5. In the table, M denotes a network model trained using data augmentation and M' is a network trained without data augmentation. For the attack, A is the set of adversarial examples created from a target network trained with data augmentation and A' is the set created from a network trained without data augmentation.

Table 5. Test accuracy of LeNet5 networks on adversarial examples. M denotes a network model trained using data augmentation, while M' is a network trained without data augmentation. A is the set of adversarial examples generated from a target network trained with data augmentation, while A' is the set created from a network trained without data augmentation.

		Attack / Model	FGSM		JSMA		C&W		Average
			A	A'	A	A'	A	A'	
Trained model	Clean	M	41.74	71.05	56.80	65.61	38.17	92.20	60.93
		M'	70.28	62.67	72.29	65.12	81.23	83.27	72.48
	FGSM	M	99.25	81.65	57.71	66.03	59.80	92.41	76.14
		M'	81.05	99.10	73.72	72.32	84.38	93.35	83.99
	JSMA	M	48.60	69.61	96.37	95.95	72.71	94.92	79.69
		M'	70.88	72.25	95.87	98.20	92.06	95.27	87.42
	C&W	M	66.96	70.38	63.05	68.56	99.40	98.94	77.88
		M'	55.09	71.68	71.74	72.00	93.09	99.31	77.15
	FGSM + JSMA	M	99.27	82.25	95.24	94.09	81.75	91.67	90.71
		M'	82.29	99.00	95.79	98.15	95.57	98.03	94.81
	FGSM + C&W	M	99.32	88.34	63.17	69.91	99.63	98.52	86.48
		M'	85.78	99.12	76.29	77.69	94.86	99.32	88.84
	JSMA + C&W	M	69.31	76.85	96.25	95.52	98.71	98.78	89.24
		M'	68.94	84.26	95.75	98.29	96.89	99.16	90.55
	FGSM + JSMA + C&W	M	99.21	86.66	95.33	94.86	98.59	97.25	95.32
		M'	85.12	98.96	95.15	97.84	97.30	99.19	95.59
	Average		76.41	82.11	81.28	83.13	86.51	95.72	

For an undefended model, the attack is more successful if the augmentation strategy matches between the defender and the attacker. The accuracy of the FGSM samples is 41.74% when both the defender and the attacker uses data augmentation when training their networks. If both do not use data augmentation, the accuracy of the FGSM samples is 62.67%. If the augmentation strategy is different between the defender and the attacker, the accuracy of the FGSM samples is approximately 70%. However, when the defender is using adversarial training, the robustness is higher when the defender and the attacker use the same augmentation strategy. In the table, if the defender uses FGSM, JSMA and C&W samples for adversarial training, the accuracy of FGSM samples is above 99% when the augmentation strategy matches between the defender and the attacker. However, the accuracy drops down to 85% if they use different augmentation strategies.

This result shows that the effectiveness of adversarial training depends not only on what algorithm is used for creating adversarial samples, but other factors such as data augmentation strategies for training the target network. It means if the network model needs to be robust against all possible attacks, the number of adversarial examples needed for adversarial training could become very large in order to cover all possible combinations. Overall, training a robust network that guarantees a certain level of robustness against all types of adversarial examples is very difficult, even for black-box attacks.

5. Conclusions

Adversarial example attacks can be a serious threat to deep learning based systems and applications such as autonomous vehicles. With limited training data, a neural network cannot learn distributions that are not present in the training samples. The attack makes use of this limitation and creates samples that look obvious to the human eye but are misclassified by the network. Adversarial training is a method that is shown to be effective if the defender and the attacker use similar methods to generate adversarial samples. However, it is also shown that adversarially training a network with one attack method does not make the model robust to other attack methods, and a network trained with multiple types of adversarial examples is still vulnerable to the white-box attacks.

We conducted experiments using the MNIST dataset to find out whether adversarial training can be effective against black-box attacks in which the attacker cannot gain access to parameters of the target network. The results show that an adversarially trained network can be vulnerable to black-box attacks, as it is vulnerable to white-box attacks. First, a network adversarially trained with an attack method is mostly robust to that particular attack method, but it is not robust to other attack methods. Second, adversarially training a network with multiple types of adversarial examples could achieve high accuracy for each of the attacks, but other factors such as data augmentation strategy can negatively affect the robustness of the network. For example, if the defender trains a network with FGSM samples and the attacker also uses FGSM to attack the network, the accuracy is considerably degraded if the defender uses data augmentation on the network and the attacker does not. Overall, adversarial training cannot practically make a network fully robust to all types of adversarial example attacks, even for black-box attacks with limited access to the target network. Since this conclusion is based on empirical results, theoretical insights are needed to conclusively assess the capabilities of adversarial training against black-box attacks, which we plan to pursue as a future work.

Another observation from the experiments is that using data augmentation techniques such as rotation and translation generally improves accuracy of a clean test set, but the accuracy of adversarial examples can be degraded. Without knowing the attacker's strategy, training the network with no augmentation was beneficial in terms of robustness against adversarial examples, if the sacrifice in accuracy for clean samples is affordable. This suggests an interesting trade-off between data augmentation and network robustness, which is another subject for future work.

Author Contributions: Conceptualization, S.P. and J.S., methodology, S.P., validation, S.P., and writing—original draft preparation, S.P. and J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation (NRF) of Korea under grant No. 2019R1A2C1005881.

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

Abbreviations

The following abbreviations are used in this manuscript:

FGSM	Fast Gradient Sign Method
JSMA	Jacobian Saliency Map Attack
C&W	Carlini and Wagner attack

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* **2012**, *60*, 1–9. [CrossRef]
2. 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, Chengdu, China, 15–17 December 2016; pp. 770–778.
3. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers); Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186. [CrossRef]
4. Ravanelli, M.; Parcollet, T.; Bengio, Y. The Pytorch-kaldi speech recognition toolkit. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 6465–6469.
5. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2014**, arXiv:1312.6199.
6. Goodfellow, I.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572.
7. 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.
8. 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, 27–30 June 2016; pp. 2574–2582.
9. 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), Saarbrucken, Germany, 21–24 March 2016; pp. 372–387.
10. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv* **2018**, arXiv:1706.06083.
11. Schmidt, L.; Santurkar, S.; Tsipras, D.; Talwar, K.; Madry, A. Adversarially robust generalization requires more data. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 5014–5026.
12. Schott, L.; Rauber, J.; Bethge, M.; Brendel, W. Towards the first adversarially robust neural network model on MNIST. *arXiv* **2019**, arXiv:1805.09190.
13. Tramèr, F.; Boneh, D. Adversarial training and robustness for multiple perturbations. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5866–5876.
14. Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z.B.; Swami, A. Practical Black-Box Attacks against Machine Learning. In Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security (ASIACCS), Abu Dhabi, UAE, 2–6 April 2017.
15. Guo, C.; Gardner, J.R.; You, Y.; Wilson, A.G.; Weinberger, K.Q. Simple Black-box Adversarial Attacks. *arXiv* **2019**, arXiv:1905.07121.
16. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
17. Chen, P.Y.; Sharma, Y.; Zhang, H.; Yi, J.; Hsieh, C.J. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), New Orleans, LA, USA, 2–7 February 2018.
18. Papernot, N.; McDaniel, P.; Goodfellow, I. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv* **2016**, arXiv:cs.CR/1605.07277
19. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv* **2017**, arXiv:1607.02533.
20. Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; Li, J. Boosting Adversarial Attacks with Momentum. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 9185–9193. [CrossRef]
21. 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.
22. Carlini, N.; Wagner, D. MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples. *arXiv* **2017**, arXiv:cs.LG/1711.08478.
23. 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, 22–26 May 2016; pp. 582–597.
24. Xu, W.; Evans, D.; Qi, Y. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In Proceedings of the 2018 Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018. [CrossRef]

25. Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; Yuille, A. Mitigating Adversarial Effects Through Randomization. *arXiv* **2018**, arXiv:1711.01991.
26. Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv* **2018**, arXiv:1805.06605.
27. Dhillon, G.S.; Azizzadenesheli, K.; Bernstein, J.D.; Kossaifi, J.; Khanna, A.; Lipton, Z.C.; Anandkumar, A. Stochastic activation pruning for robust adversarial defense. *arXiv* **2018**, arXiv:1803.01442.
28. Liu, X.; Cheng, M.; Zhang, H.; Hsieh, C. Towards robust neural networks via random self-ensemble. In Proceedings of the 15th European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018. [CrossRef]
29. Lu, J.; Issaranon, T.; Forsyth, D. SafetyNet: Detecting and rejecting adversarial examples robustly. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017, pp. 1–9.
30. Metzen, J.; Genewein, T.; Fischer, V.; Bischoff, B. On detecting adversarial perturbations. In Proceedings of the Fifth International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
31. Grosse, K.; Manoharan, P.; Papernot, N.; Backes, M.; McDaniel, P. On the (Statistical) Detection of Adversarial Examples. *arXiv* **2017**, arXiv:1702.06280.
32. Feinman, R.; Curtin, R.; Shintre, S.; Gardner, A. Detecting adversarial samples from artifacts. *arXiv* **2017**, arXiv:stat.ML/1703.00410.
33. Hendrycks, D.; Gimpel, K. Early methods for detecting adversarial images. *arXiv* **2017**, arXiv:1608.00530.
34. Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; Kushman, N. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. *arXiv* **2018**, arXiv:1710.10766.
35. Kingma, D.P.; Ba, J. Adam: A Method for Stochasic Optimization. *arXiv* **2014**, arXiv:1412.6980.
36. He, Z.; Xie, L.; Chen, X.; Zhang, Y.; Wang, Y.; Tian, Q. Data augmentation revisited: Rethinking the distribution gap between clean and augmented data. *arXiv* **2019**, arXiv:cs.LG/1909.09148.