Attack as Defense: Characterizing Adversarial Examples using Robustness

Zhe Zhao
ShanghaiTech University
Shanghai, China
zhaozhe1@shanghaitech.edu.cn

Guangke Chen
ShanghaiTech University
Shanghai, China
chengk@shanghaitech.edu.cn

Jingyi Wang
Zhejiang University
Hangzhou, China
wangjyee@zju.edu.cn

Yiwei Yang
ShanghaiTech University
Shanghai, China
yangyw@shanghaitech.edu.cn

Fu Song*
ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
Shanghai, China
songfu@shanghaitech.edu.cn

Jun Sun
Singapore Management University
Singapore
junsun@smu.edu.sg

ABSTRACT

As a new programming paradigm, deep learning has expanded its application to many real-world problems. At the same time, deep learning based software are found to be vulnerable to adversarial attacks. Though various defense mechanisms have been proposed to improve robustness of deep learning software, many of them are ineffective against adaptive attacks. In this work, we propose a novel characterization to distinguish adversarial examples from benign ones based on the observation that adversarial examples are significantly less robust than benign ones. As existing robustness measurement does not scale to large networks, we propose a novel defense framework, named attack as defense (A^{2}D), to detect adversarial examples by effectively evaluating an example's robustness. A^{2}D uses the cost of attacking an input for robustness evaluation and identifies those less robust examples as adversarial since less robust examples are easier to attack. Extensive experiment results on MNIST, CIFAR10 and ImageNet show that A^{2}D is more effective than recent promising approaches. We also evaluate our defense against potential adaptive attacks and show that A^{2}D is effective in defending carefully designed adaptive attacks, e.g., the attack success rate drops to 0% on CIFAR10.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Software and its engineering** → *Software testing and debugging*.

KEYWORDS

Deep learning, neural networks, defense, adversarial examples

*Fu Song is the corresponding author.

ACM Reference Format:
Zhe Zhao, Guangke Chen, Jingyi Wang, Yiwei Yang, Fu Song, and Jun Sun. 2021. Attack as Defense: Characterizing Adversarial Examples using Robustness. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21), July 11–17, 2021, Virtual, Denmark.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3460319.3464822

1 INTRODUCTION

Deep learning (DL) has arguably become a new programming paradigm which takes over traditional software programs in many areas. For instance, it has achieved state-of-the-art performance in real-world tasks such as autonomous driving [1], medical diagnostics [61] and cyber-security [65]. Despite the success, DL software are still far from dependable (especially for safety- and security-critical systems) and, like traditional software, they must be properly tested, and defended in the presence of malicious inputs. In particular, DL software are known to be brittle to adversarial examples [10, 11, 22, 37, 69], i.e., by adding a slight perturbation on an input, a well-trained DL model could be easily fooled.

There is a huge body of work proposing various attack and defense mechanisms with regard to adversarial examples, from both the security [6, 22, 32, 52] and software engineering community [41, 68, 71, 75, 76]. Since the adversarial attack L-BFGS was introduced [69], many sophisticated adversarial attacks have been proposed, such as Fast Gradient Sign Method (FGSM) [22], Iterative Gradient Sign Method (BIM) [32], Jacobian-based Saliency Map Attack (JSMA) [52], Local Search Attack (LSA) [49], Decision-Based Attack (DBA) [6], DeepFool [47], and Carlini and Wagner's attack (C&W) [10]. As countermeasures, defense mechanisms are proposed to improve robustness in the presence of adversarial attacks. Examples include adversarial training [22, 45, 59], defensive distillation [53] and feature squeezing [84]. These works constitute important steps in exploring the defense mechanisms, yet have various limitations and are shown to be insufficient [2, 9, 10, 25, 73].

Some efforts also have been made to distinguish adversarial examples from benign ones and reject those potentially adversarial ones [75, 76]. A usual underlying assumption is that adversarial and benign examples differ in a certain subspace distribution, e.g., kernel density [20], local intrinsic dimensionality [44], manifold [46], logits values [57], etc. Although these characterizations provide some insights on the adversarial subspace, they are far from reliably discriminating adversarial examples alone. Worse yet, attackers can easily break existing defense mechanisms by specifically designed adaptive attacks [8, 9, 73].

In this work, we propose a novel characterization, i.e., robustness, to distinguish adversarial examples from benign ones. Our main observation is that adversarial examples (crafted by most existing attack methods) are significantly less robust than benign ones. Different from previous statistical characterizations [20, 44, 46, 57], the difference in robustness comes from two inherent characteristics of model training and adversarial example generation. First, the model training progressively minimizes the loss of each example with respect to the model. Thus, a benign example is often trained to be robust against a model with good generalization. As a result, benign examples are in general relatively far away from the decision boundary. Second, most adversarial attacks aim to generate human-imperceptible perturbations which often result in much less robust just-cross-boundary adversarial examples. The contrasting robustness characteristics make it suitable to distinguish adversarial examples from benign ones.

However, it is still an open research problem on how to effectively measure an input's robustness with respect to a DL model. Existing robustness measurement methods (e.g., CLEVER [81] which calculates a minimum perturbation required to change an input's label) are often too computationally expensive. In this work, we propose to utilize the robustness difference from a reversed angle. Our intuition is that it is easier to employ a successful adversarial attack on an input that is less robust. Thus, we propose a novel defense, named *attack as defense* (A^{2}D), to effectively detect adversarial examples from benign ones. Given an input example, we apply different kinds of attacks to attack the input and measure how 'easy' it is to employ a successful attack. An input example is considered less robust and thus more likely to be adversarial, if the attacks are easier to succeed. Note that the effectiveness of our defense (A^{2}D) relies on a set of attacks whose attack cost (easiness) can be quantitatively measured, which will guide the selection of attacks from a large number of adversarial attacks in the literature.

Compared to existing adversarial example detection approaches (which are mostly proven to be ineffective [2, 8–10, 25, 73]), our A^{2}D framework has the following advantages. First, A^{2}D utilizes the inherent robustness difference caused by the contrasting characteristics of model training and adversarial example generation. To circumvent the defense, an attack needs to generate robust adversarial examples which in general might induce human-perceptible large distortion. Second, A^{2}D is essentially an ensemble approach using the robustness information obtained from different kinds of attacks which is hard to bypass at once. We evaluated A^{2}D with a carefully selected set of attacks on three popular datasets MNIST [35], CIFAR10 [31] and ImageNet [73]. Experimental results show that A^{2}D can be more effective than recently proposed detection algorithms [20, 44, 75, 76] and remain effective even in the white-box adversarial setting. We further show that A^{2}D is effective against adaptive attacks. That is, we show that A^{2}D (combined with a complementary detection approach for detecting large-distortion adversarial examples [46] or adversarial training) is very effective against specifically designed adaptive attacks, e.g., the attack success rate (ASR) drops from 72% to 0% on CIFAR10 using our defense with adversarial training, and the ASR drops from 100% to 0% on MNIST using our defense with autoencoder [46]. We remark that many existing defenses combined with adversarial training result in lower robustness than adversarial training on its own [73].

In a nutshell, we make the following contributions:

- We propose a novel characterization to distinguish adversarial examples from benign ones via robustness.
- We present detection approaches based on our novel characterization, which can utilize existing attacks and do not need to modify or retrain the protected model.
- We conduct extensive experiments to test our observations and our defense, which outperforms recent promising detection algorithms.
- We thoroughly discuss possible adaptive attacks to our defense and evaluate them to our defense integrated with a complementary detection approach and adversarial training. The integrated defense is very promising.

The remainder of this paper is organized as follows. Section 2 presents preliminaries. Section 3 introduces our characterization accompanied with experiments for validating. Section 4 demonstrates various detection approaches using our characterization. Experimental results are reported in Section 5. Section 6 discusses and evaluates adaptive attacks against our defense. Section 7 discusses related works. Finally, we conclude this work in Section 8.

2 BACKGROUND

2.1 Adversarial Attacks

In this work, we target deep neural network (DNN) for classification tasks. We denote a DNN by $f : X \rightarrow C$, which maps each input $x \in X$ to a certain label $c \in C$. We denote the ground-truth label of an input x by c_x. Given a DNN f and a benign input x (which means $f(x) = c_x$), the attacker's goal is to craft a perturbation Δx (measured in different L_n norms [10]) for the input x such that the DNN f classifies the example $\hat{x} = x + \Delta x$ as a different label $f(\hat{x})$, i.e., $f(\hat{x}) \neq f(x)$. Such $\hat{x}$ is called an *adversarial example*.

In the literature, an extensive number of adversarial attacks have been proposed [6, 10–12, 15, 19, 22, 27, 32, 37, 47, 48, 50, 52, 69]. We briefly introduce some representative attacks that will be used for robustness evaluation in our work.

FGSM. Fast Gradient Sign Method (FGSM) [22] uses a loss function $J(x, c_x)$ (e.g. the cross-entropy loss) to describe the cost of classifying x as label c_x, and maximizes the loss to implement an untargeted attack by performing one step gradient ascend from the input x with a L_∞ distance threshold ϵ. Formally, a potential adversarial example $\hat{x}$ is crafted as follows:

$$\hat{x} = x + \epsilon \times \mathbf{sign}(\nabla_x J(x, c_x))$$

where ∇_x is the partial derivative of x, and $\mathbf{sign}(\cdot)$ is a sign function such that $\mathbf{sign}(c)$ is +1 if $c > 0$, −1 if $c < 0$ and 0 if $c = 0$.

BIM. Basic Iterative gradient Method (BIM) [32] is an iterative version of FGSM. For each iteration, BIM performs FGSM with a small step size α and clips the result so that it stays in the ϵ-neighbourhood of the input sample. The ith iteration is updated by as follows:

$$x^{i+1} = \mathtt{clip}_{\epsilon,x}(x^i + \alpha \times \mathbf{sign}(\nabla_x J(x^i, c_x)))$$

where $x^0 = x$, and the iterative process can repeat several times.

The perturbation of FGSM and BIM is restricted by the L_∞ norm, measuring the largest change between $\hat{x}$ and x (i.e. $\|x - \hat{x}\|_\infty \leq \epsilon$). We could derive L_2 norm (i.e. $\|x - \hat{x}\|_2 \leq \epsilon$) attacks by,

$$\hat{x} = x + \epsilon \times \frac{\nabla_x J(x, c_x)}{\nabla_x \|J(x, c_x)\|_2}$$

Similarly, FGSM and BIM can be adapted from untargeted attacks to target ones which specify the target label of an adversarial example.

Compared to fixed step size, there are some optimization-based attack methods that seek to find adversarial examples with the minimal perturbation, such as L-BFGS [69] and C&W [10]. In addition, JSMA [52] seeks to modify the smallest number of pixels, which is an attack method with the L_0 norm.

2.2 Robustness

A DNN f is (locally) *robust* with respect to an input x and an L_p norm distance threshold ϵ if for every example $\hat{x}$ such that $\|x - \hat{x}\|_p \leq \epsilon$, $f(x) = f(\hat{x})$ holds. Several approaches have been proposed to certify robustness, based on SAT/SMT/MILP solving [17, 29], abstraction refinement [18, 77, 78], and abstract interpretation [21, 63, 64]. Though these approaches feature theoretical guarantees, they are limited in scalability and efficiency, hence fail to work for large models in practice. To improve scalability, a few approaches were proposed to achieve statistical guarantees, by claiming robustness with certain probability [4, 58, 81]. Among them, CLEVER [81] score is an effective attack-independent metric to estimate robustness by sampling the norm of gradients and fitting a limit distribution using extreme value theory. In this work, we use the CLEVER score to compare the robustness of adversarial and benign examples.

2.3 Problem Formulation

We focus on the detection of adversarial examples as motivated by many relevant works [20, 44, 75, 76]. The problem is: *given an input example x to a DNN model f, how to effectively decide whether x is benign or adversarial?* The fundamental problem is how to better characterize adversarial examples. Our solution is to use robustness.

Threat Model. We consider a challenging defense scenario which assumes that the adversary knows all the information of the model under attack, namely, white-box attacks. Besides, we assume the detector has access to a set of benign examples, but knows nothing about how the adversary generates adversarial examples. We also assume the detector can use various attacks (for robustness evaluation). These assumptions are reasonable in practice, as there are many publicly available datasets and source of attacks.

3 CHARACTERIZATION

3.1 Robustness: Adversarial vs. Benign

Our detection approach is based on the observation that adversarial examples are much less robust than benign ones. To understand the underlying reason, we briefly recap the processes of DL model training and adversarial example generation. Training a DL model typically takes multiple epochs. For each epoch, the training dataset is partitioned into multiple batches and each batch of input examples is trained once. After each batch, the parameters are updated,

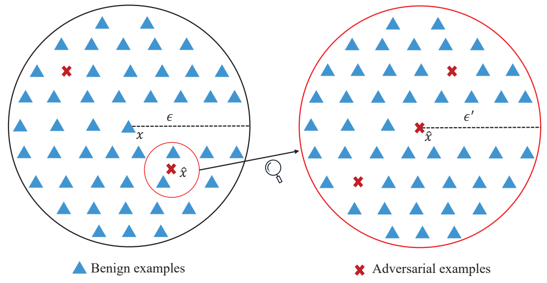


Figure 1: An illustration of robustness of adversarial and benign examples. The left-part depicts ϵ-neighbourhood of a benign example x and the right-part depicts ϵ-neighbourhood of the adversarial example $\hat{x}$, where each triangle denotes an example that can be correctly classified into the label c_x and each cross denotes an example that cannot be correctly classified into the label c_x.

Table 1: Parameters of attacks, the accuracy of the target MNIST model on training/testing dataset is 99.6%/99.1% for MNIST and 87.3%/80.3% for CIFAR10

Dataset	Attack Method	Parameters
MNIST	FGSM	$\epsilon = 0.3$
	BIM	$\epsilon = 0.3, \alpha = 0.01$
	JSMA	$\theta = 1, \gamma = 0.1$
	C&W	$\kappa = 0, c = 0.02$
CIFAR10	FGSM	$\epsilon = 0.05$
	BIM	$\epsilon = 0.05, \alpha = 0.005$
	JSMA	$\theta = 1, \gamma = 0.1$
	C&W	$\kappa = 0, c = 0.02$

e.g., via stochastic gradient descent. Once all the epochs finish, the DL model is ready for testing. During training, each example in the dataset goes through a number of epochs. Consequently, it is often the case that the trained DL model achieves good generalization results. Therefore, as illustrated in Figure 1 (left-part), under a reasonable distance threshold ϵ, most examples in the ϵ-neighborhood of a benign example x are also benign while adversarial examples are relatively far away from the benign example x.

In the process of adversarial example generation, the attacker crafts a perturbation Δx for a benign example x such that the resulting example $\hat{x} = x + \Delta x$ is adversarial. During the generation of adversarial examples, attackers often neglect robustness due to the pursuit of other attributes, such as minimal perturbation, invisibility, target label classification and query efficiency. These attributes and robustness are often incompatible, and thus difficult to achieve simultaneously. Consequently, adversarial examples with small distortion are very close to the decision boundary [76] and most examples in the ϵ'-neighbourhood of the adversarial example $\hat{x}$ are benign (with respect to the original example x). This is illustrated in Figure 1 (right-part), which is the zoom in of the ϵ'-neighbourhood of the adversarial example $\hat{x}$ in Figure 1 (left-part).

The above observation can be quantified using robustness. We conduct a quantitative robustness comparison of benign and adversarial examples using the CLEVER score on MNIST and CIFAR10

Table 2: CLEVER scores with confidence interval of 90% significance level. Column *Label for Evaluate* shows the different results under untargeted and targeted attacks, where LLC denotes the least likely class. Column *Benign examples* shows the results of the benign examples, and other columns show the results under the corresponding attack methods. Columns λ show the ratio of the CLEVER scores of benign examples to that of the adversarial ones for each attack.

Dataset	Label for Evaluate	Benign examples	Adversarial examples								Avg.
			FGSM	λ	BIM	λ	JSMA	λ	C&W	λ	λ
MNIST	Untarget	3.5572 ± 0.3342	0.1093 ± 0.0506	32.55	0.0256 ± 0.0031	138.95	0.0550 ± 0.0060	64.68	0.0004 ± 0.0001	8893	74.77
	Target-2	3.6711 ± 0.3296	0.1148 ± 0.0427	31.98	0.0258 ± 0.0031	142.29	0.0558 ± 0.0063	65.79	0.0004 ± 0.0001	9178	74.62
	Target-5	3.8303 ± 0.3113	0.2047 ± 0.0431	18.71	0.1582 ± 0.0084	24.21	0.1898 ± 0.0096	20.18	0.1384 ± 0.0043	27.68	22.17
	LLC	3.8372 ± 0.3097	0.2390 ± 0.0421	16.06	0.1647 ± 0.0071	23.30	0.2120 ± 0.0076	18.10	0.1406 ± 0.0045	27.29	20.29
CIFAR10	Untarget	0.3851 ± 0.1850	0.2743 ± 0.1627	1.40	0.0329 ± 0.0033	11.71	0.0128 ± 0.0021	30.09	0.0005 ± 0.0002	770	4.81
	Target-2	0.4141 ± 0.1806	0.2971 ± 0.1675	1.39	0.0380 ± 0.0044	10.90	0.0129 ± 0.0021	32.10	0.0005 ± 0.0002	828	4.75
	Target-5	0.4657 ± 0.1913	0.3389 ± 0.1675	1.37	0.0971 ± 0.0117	4.80	0.0610 ± 0.0061	7.63	0.0925 ± 0.0168	5.03	3.16
	LLC	0.4829 ± 0.1913	0.3572 ± 0.1713	1.35	0.1091 ± 0.0132	4.43	0.0918 ± 0.0095	5.26	0.1035 ± 0.0180	4.67	2.92

datasets. For each dataset, we choose the first 100 images from the test dataset as subjects. Adversarial examples are generated by applying four representative attacks: FGSM, BIM, JSMA, and C&W. The models and attack tools are taken from [20], where the parameters are presented in Table 1. The CLEVER scores, in the form of a confidence interval of 90% significance level [13], are shown in Table 2.

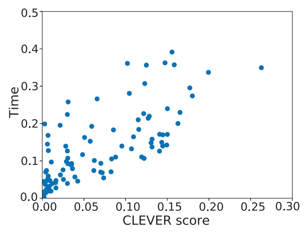
We can observe that the CLEVER scores of benign examples are much larger than that of adversarial ones for both MNIST and CIFAR10 datasets, though the difference varies from attacks and datasets. This indicates that the difference of robustness between adversarial and benign examples is significant, thus confirming our observation. We also observe that the ratios λ using untarget/target-2 label for computing the CLEVER scores are larger than the ones using other labels. This is because that the label of untarget or target-2 for each adversarial example is often the label of its benign counterpart, which also confirms our observation that most examples in ϵ-neighbourhood of each adversarial example have the same label of the benign counterpart.

3.2 Attack Cost: Adversarial vs. Benign

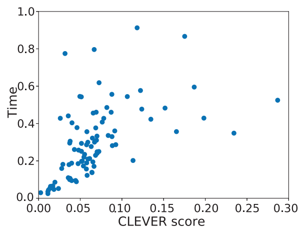
Based on the above observation, one could design adversarial example detection approaches similar to other characterizations like label change rate [76]. However, existing techniques for robustness certification (with statistical guarantees) still have limited scalability, and hence are not able to handle large models efficiently. For instance, on a single GTX 1080 GPU, the cost of computing the CLEVER score is near: 450 seconds for each MNIST example and 1150 seconds for each CIFAR10 example using untarget, 50 seconds for each MNIST example and 128 seconds for each CIFAR10 example using target-2/5.

To effectively and efficiently detect adversarial examples, we propose a novel detection approach, named *attack as defense* (A^{2}D for short), which uses the cost of attacking an example to test its robustness. The underlying assumption is that the more robust the example is, the more difficult (a larger attack cost) it is to attack. The implication is that we can decide whether an input example is likely to be adversarial by utilizing off-the-shelf attacks.

To leverage attack cost to detect adversarial examples, the first problem needs to be tackled is *how to select attacks for defense*. In general, the attack cost should be able to be quantified and reflect inputs' robustness. As a result, FGSM is not suitable since it simply performs one-step perturbation. In contrast, iterative attacks (such as BIM, JSMA, and C&W) that iteratively search for adversarial examples with least distortion could be leveraged, as the costs of such attacks can be quantified and are relevant to inputs' robustness.



(a) Score vs. time on MNIST

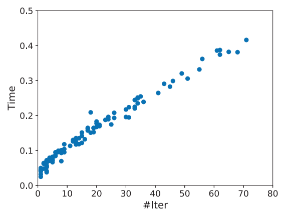


(b) Score vs. time on CIFAR10

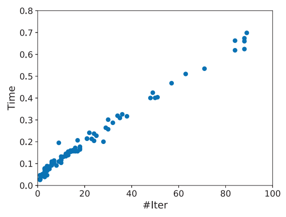
Figure 2: CLEVER score vs. attack time using JSMA

We illustrate this observation using JSMA. JSMA calculates a saliency map based on the Jacobian matrix to model the impact that each pixel imposes on the classification result. During each iteration, JSMA uses a greedy algorithm that modifies certain pixels to increase the probability of the target label. The process is repeated until finding an adversarial example or reaching the termination criteria. The attack cost (time and iteration) of JSMA depends on the robustness of each example. For an example x that is less robust than another one x', an adversarial example of x can be quickly constructed using less time/iteration than the one of x'. To further test this observation, we compare the attack time of JSMA on 100 MNIST and 100 CIFAR10 examples whose CLEVER scores range from 0 to 0.3. The results are reported as scatter plots in Figure 2(a) and Figure 2(b), which confirm our observation.

The next problem is then *how to characterize attack costs for different kinds of attacks*. The most direct indicator of attack costs is the attack time as demonstrated in Figure 2. The attack time of different examples can reflect their robustness. However, the attack time is easily affected by the real-time performance of computing devices in a physical environment, which makes the variance of attack time intolerable. For an iterative attack, the number of iterations is positively correlated with attack time. This is justified by the scatter plots shown in Figure 3(a) and Figure 3(b) using the same experiments as above. Therefore, in this work, we propose to



(a) Time vs. ♯iter on MNIST



(b) Time vs. ♯iter on CIFAR10

Figure 3: Attack time vs. iterations (♯iter) using JSMA

use the number of iterations of the attacks as the indicator of the attack costs.

To demonstrate the effectiveness of the attack costs for characterizing adversarial examples, we choose 5 types of images (Benign, FGSM, BIM, DeepFool and C&W), each of which randomly select 1,000 samples, and divided each type of images into two independent sets. Then we use JSMA to attack these images and record the number of iterations required. To show the difference in attack costs, we calculate the average Euclidean distance of the number of iterations between each pair of sets of examples, the results are presented in Figure 4. We can see that for different types of examples (adversarial vs. benign), the distance is enormous. While for the same types of examples (adversarial vs. adversarial or benign vs. benign), the distance is close to zero. It is worth mentioning that for the examples generated by different attacks, the distance is also very similar, meaning that even if the adversarial examples are generated by different attacks, they are also "cognate" examples and have similar attack costs.

Utilizing the diversity of attack methods, an ensemble detection method can be constructed jointly. As we mentioned before, iterative attacks have the potential to be used as defense, so we can integrate multiple iterative attacks to derive a more robust defense. Different attacks have the ability to capture different characterizations. For example, JSMA crafts adversarial examples based on the L_0 norm, while BIM is based on the L_∞ norm, thus they measure the robustness of inputs under different distance metrics. Thanks to various types of attacks, ensemble multiple attacks will make the defense more reliable and difficult to bypass.

4 DETECTION APPROACH

In this section, we consider how to detect adversarial examples by leveraging attack costs. In this work, we propose two effective detection approaches that are based on k-nearest neighbors (K-NN) and standard score (Z-score), respectively. The former requires both benign and adversarial examples, while the latter requires only benign examples.

Hereafter, we sometimes denote by attack$_d$ the attack that is used as defense, i.e., to generate attack costs.

4.1 K-NN Based Detection Approach

Assume that we have two disjoint sets: B the set of benign examples and A the set of adversarial examples.

Single detector. Let us consider the attack$_d$ o. The attack cost α_y of attacking an example y using the attack$_d$ o is regarded as the

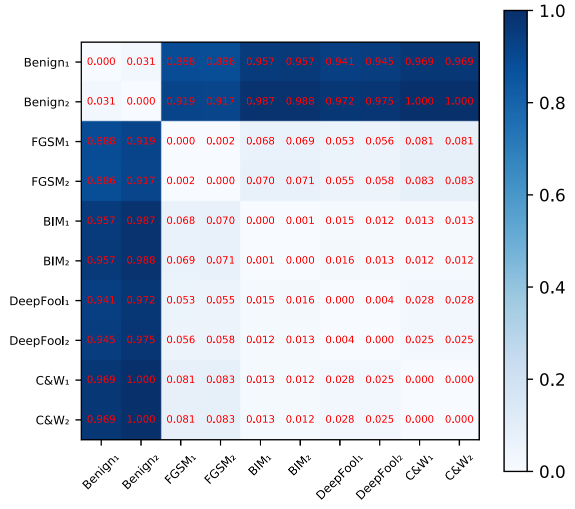


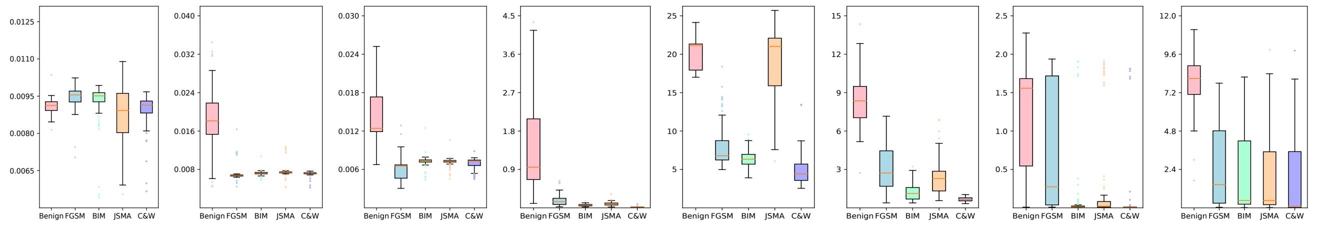
Figure 4: Euclidean distances of the average number of iterations between each pair of sets of examples

fingerprint of y. We can generate a set of fingerprints $\{\alpha_y \mid y \in A \cup B\}$ from the examples $y \in A \cup B$ by utilizing the attack$_d$ o. For each unknown input x and parameter K, we first compute the attack cost α_x of the input x using the attack$_d$ o and then identify the K-nearest neighbors $N_K = \{\alpha_{y_i} \mid 1 \leq i \leq K\}$ of α_x from the set $\{\alpha_y \mid y \in A \cup B\}$. The set N_k is partitioned into two subsets: $A_x = \{y \in A \mid \alpha_y \in N_K\}$ and $B_x = \{y \in B \mid \alpha_y \in N_K\}$. The input x is classified as adversarial if $|A_x| > |B_x|$, namely, the number of adversarial examples is larger than that of benign ones in K-neighbourhood of the input x.

Ensemble detector. The K-NN based detection approach can be easily generalized from one attack$_d$ to multiply attacks$_d$ $o_1, \cdots, o_n$, leading to a more robust detector. Under this setting, the fingerprint of an example y is a vector of attack costs, $\vec{\alpha}_y = (\alpha_y^1, \cdots, \alpha_y^n)$, where for every $1 \leq j \leq n$, α_y^j is the attack cost of the example y by utilizing the attack$_d$ o_j. Consequently, we can generate a set of fingerprints $\{\vec{\alpha}_y \mid y \in A \cup B\}$ from the examples $y \in A \cup B$ by utilizing the attacks$_d$ $o_1, \cdots, o_n$. Similar to the single attack setting, for each unknown input x and parameter K, we identify the K-nearest neighbors $N_K = \{\vec{\alpha}_{y_i} \mid 1 \leq i \leq K\}$ of the fingerprint $\vec{\alpha}_x$ of the input x and partition N_k into two subsets: $A_x = \{y \in A \mid \vec{\alpha}_y \in N_K\}$ and $B_x = \{y \in B \mid \vec{\alpha}_y \in N_K\}$. The input x is classified as adversarial if $|A_x| > |B_x|$.

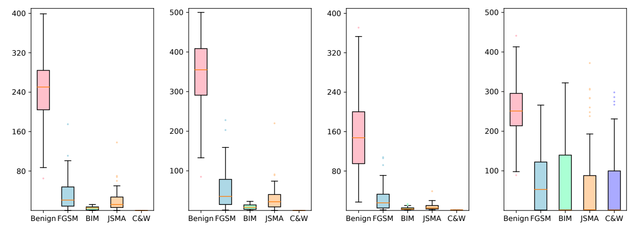
4.2 Z-Score Based Detection Approach

Z-score is a well-known concept in statistics for measuring a sample in terms of its relationship to the mean and standard deviation of a dataset [34]. The Z-score of a sample i is defined by: $z = \frac{i-\mu}{\sigma}$, where μ is the sample mean and σ is the sample standard deviation. Intuitively, the score z indicates how many standard deviations that the sample i is far away from the sample mean. Our Z-Score based detection approach leverages the distribution of attack costs



(a) FGSM_d (b) BIM_d (c) BIM2_d (d) JSMA_d (e) $\text{C\&W}_d$ (f) L-BFGS_d (g) LSA_d (h) DBA_d

Figure 5: Attack time of benign and adversarial examples, where y-axis means seconds



(a) BIM_d (b) BIM2_d (c) JSMA_d (d) DBA_d

Figure 6: Attack iterations comparison

of benign examples to check whether an example y is adversarial or not. Thus, it is likely more robust with respect to unseen attacks.

Single detector. Let us consider a set B of benign examples and an attack$_d$ o as defense. We can compute the distribution of attack costs of the examples in B. Assume that the distribution is an approximately normal distribution $N(\mu, \sigma^2)$. Otherwise, we can transform it by applying the Box-Cox power transformation [5]. Thus, the Z-score z_y of an example y is defined as: $z_y = \frac{\alpha_y - \mu}{\sigma}$, where α_y denotes the cost of attacking y using the attack$_d$ o. For a given ratio h of the sample standard deviation as the threshold, based on our observation that adversarial examples are less robust than benign ones, an input x is classified to adversarial if $z_x < h$, i.e., x is h standard deviations away from the sample mean.

Ensemble detector. We can also generalize this approach from one attack$_d$ to multiply attacks$_d$ $o_1, \cdots, o_n$. For each attack$_d$ o_j, we can construct a Z-Score based detector d_j, resulting in detectors $d_1, \cdots, d_n$. The ensemble detector determines whether an input is adversarial or not by taking into account the results of all the detectors $d_1, \cdots, d_n$. Consider $k \leq n$, the ensemble detector classifies an input to adversarial if k detectors classify the input to adversarial, otherwise benign. The ensemble detector would have high true positive rates when $k = 1$, high true negative rates when $k = n$.

5 EVALUATION

In this section, we evaluate our approach on three widely used datasets MNIST, CIFAR10 and ImageNet. The experiments are designed to answer the following research questions:

RQ1. How to select effective attacks for defense?
RQ2. How effective are the selected attacks for defense?
RQ3. How effective and efficient is A^2D (i.e., detection)?

Experiment setups. For reproductivity of this work, all the information of the target models and attack parameters used in our experiments are given below. In total, we compared with 4 baselines, involving 3 different environments (Env_1 on the Keras platform, Env_2 and Env_3 on the PyTorch platform). Since the performance of these four defenses may vary due to platforms, CNN models and attack settings. For a fair comparison, we conduct comparison directly using the same target models and attacks provided by each of them, and implement our approach in all environments. It is worth mentioning that the full source code of some baselines is not available. Thus, for the missing attacks, we use alternative implementations in Foolbox [56], recommended by [73].

The first baseline [20], denoted by BL_1, uses a Gaussian Mixture Model to represent outputs from the final hidden layer of a CNN, which was considered to be *the most effective defense on MNIST* among ten detections in [9]. The second baseline [44], denoted by BL_2, uses local intrinsic dimension to represent and distinguish adversarial subspace and claims to be better than BL_1. BL_3 [76] uses label change rate through model mutation testing to distinguish adversarial examples. BL_4 [75] dissects the outputs of intermediate layers to construct a fault tolerance approach. We emphasize that BL_3 and BL_4, published respectively in ICSE'19 and ICSE'20, are arguably two state-of-the-art defense approaches.

Env_1 contains models and attack methods provided by BL_1 [20]. The DL model for MNIST is LeNet, the DL model for CIFAR10 is a deep 12-layer convnet. The accuracy of the target model on training/testing dataset is 99.6%/99.1% for MNIST and 87.3%/80.3% for CIFAR10. BL_2 [44] uses the models and attack code segments provided by BL_1, so Env_1 is the environment used by these two baselines. It should be noted that there are two slightly different BIM implementations in Env_1, and we use the 'bim-a'.

Env_2 is provided by BL_3 [76], which includes LeNet5 model for MNIST and GooglLeNet model for CIFAR10. The accuracy of the target model on training/testing dataset is 98.5%/98.3% for MNIST and 99.7%/90.5% for CIFAR10. BL_3 stated that the BB is ineffective on CIFAR10 as the authors could not train a good substitute model, so we omit BB on CIFAR10.

Env_3 is provided by BL_4 [75], which includes LeNet4 for MNIST, WRN for CIFAR10 and ResNet101 for ImageNet. The accuracy of the target model on training/testing dataset is 90.0%/98.4% for MNIST and 100.0%/96.2% for CIFAR10. In addition, ResNet101 has a top-1 accuracy rate 77.36% on the validation set.

Tables 1 and 3 respectively show the attack parameters of Env_1 and Env_2. Env_3 only provides one attack that is L_2 norm adoption

Table 3: Parameters of attacks for Env_2

Dataset	Attack Method	Parameters
MNIST	FGSM	$\epsilon = 0.35$
	JSMA	$\theta = 1, \gamma = 0.12$
	DeepFool	overshoot=0.02
	C&W	$\kappa = 0, c = 0.6$
	BB	Sub model+FGSM, $\epsilon = 0.35$
CIFAR10	FGSM	$\epsilon = 0.05$
	JSMA	$\theta = 1, \gamma = 0.12$
	DeepFool	overshoot=0.02
	C&W	$\kappa = 0, c = 0.6$

of FGSM with 0.016 as the attack step. In order to make the results more comprehensive, we used the three attacks JSMA, DeepFool and C&W, implemented in Foolbox, to generate adversarial examples with default parameters.

5.1 RQ1: Attack Selection

We answer **RQ1** by comparing attack costs of adversarial and benign examples. Adversarial examples are crafted in the same setting as in Section 3.1, using the first 1,000 images from MNIST.

For defense, we choose eight attacks: FGSM, BIM, BIM2 (BIM under L_2 norm), JSMA, C&W, L-BFGS, LSA, DBA from Foolbox [56]. For ease of reading, an attack used as a defense is marked by a subscript d, e.g., FGSM_d. According to the results of Table 2, we use untargeted attack as defense, unless the attack only supports targeted attack for which we use target-2 attack. We use the default parameters of Foolbox except that BIM_d and BIM2_d immediately terminate when an adversarial example is found, and DBA_d terminates when the MSE between the adversarial example and its original version is less than 0.02, otherwise the number of iterations is fixed, namely, the attack costs of all the examples will be similar.

The results in terms of attack time are reported as boxplots in Figures 5. It is not surprising that the attack time of adversarial and benign examples using FGMS_d are similar, as it is a one-step attack. Though the results show the variation between different attacks and defenses, the differences are often significant when an iterative attack is used as defense, e.g., BIM_d, BIM2_d, JSMA_d, L-BFGS_d and DBA_d, while L-BFGS_d is less efficient than the others. We find that the differences are not stable when $\text{C\&W}_d$ and LSA_d are used. This is because that $\text{C\&W}_d$ implements a binary search to minimize distortion and may stop searching when there is a bottleneck, while LSA_d suffers from a low attack success rate, which causes it to have many outlier attack times.

Based on the above results, considering the efficiency and discrepancy in attack time between benign and adversarial examples, BIM_d, BIM2_d, JSMA_d, and DBA_d will be used as defense in the follow-up experiments. These four methods cover both white-box and black-box attacks, as well as different distance metrics L_0, L_2 and L_∞. It should be noted that with an amount of adversarial attacks being proposed (more than 2,000 papers in 2020), it is impossible to evaluate all of them. In this work, we only analyze the pros and cons of the above methods and choose suitable attacks.

The above experiments demonstrate how to quickly select a suitable attack as defense. In order to ensure the reliability of BIM_d,

Table 4: AUROC comparison of our approach over baselines

Env_1	Attack	JSMA_d	BIM_d	BIM2_d	DBA_d	BL_1	BL_2
MNIST	FGSM	0.9653	**0.9922**	0.9883	0.9504	0.8267	0.9161
	BIM	0.9986	**0.9996**	0.9995	0.9625	0.9786	0.9695
	JSMA	**0.9923**	0.9922	0.9914	0.9497	0.9855	0.9656
	C&W	**1.0**	**1.0**	**1.0**	0.9672	0.9794	0.9502
CIFAR10	FGSM	0.6537	0.712	0.6474	0.6977	0.7015	**0.7891**
	BIM	0.8558	**0.8636**	0.861	0.8276	0.8255	0.8496
	JSMA	0.9459	**0.955**	0.9526	0.9452	0.8421	0.9475
	C&W	0.9905	0.9984	**0.9988**	0.9833	0.9217	0.9799

Env_2	Attack	JSMA_d	BIM_d	BIM2_d	DBA_d	BL_3
MNIST	FGSM	0.9665	**0.9883**	0.9846	0.9595	0.9617
	JSMA	0.9971	**0.9984**	0.9974	0.984	0.9941
	DeepFool	0.9918	**0.9971**	0.9951	0.9587	0.9817
	C&W	0.9456	**0.9870**	0.9769	0.8672	0.9576
	BB	0.9746	**0.9895**	0.9852	0.9535	0.9677
CIFAR10	FGSM	0.8808	0.8994	**0.8998**	0.8746	0.8617
	JSMA	0.9774	**0.9890**	0.9873	0.9566	0.9682
	DeepFool	0.9832	0.9898	**0.9902**	0.9769	0.9614
	C&W	0.8842	**0.9176**	0.9175	0.9004	0.9063

Env_3	Attack	JSMA_d	BIM_d	BIM2_d	DBA_d	BL_4
MNIST	FGSM	0.9985	0.9999	**1.0**	0.9674	0.9993
	JSMA	0.9972	0.9998	**0.9999**	0.9113	0.9993
	DeepFool	0.9702	0.9877	0.9874	0.9255	**0.9892**
	C&W	0.9985	**1.0**	**1.0**	0.9623	0.9996
CIFAR10	FGSM	0.9945	0.9979	**0.9983**	0.9629	0.9981
	JSMA	0.9934	0.9962	0.9961	0.976	**0.9966**
	DeepFool	**0.9713**	0.9703	0.9692	0.9604	0.9618
	C&W	0.9951	0.9981	**0.9985**	0.9928	0.9968
ImageNet	FGSM	0.973	0.9763	**0.9782**	0.9625	0.9617
	JSMA	**0.9962**	0.9805	0.99	0.9937	0.9695
	DeepFool	**0.9958**	0.9793	0.9892	0.9891	0.9924
	C&W	0.9873	0.9731	0.9801	**0.9924**	0.9636

BIM2_d, JSMA_d, and DBA_d, we also analyze the numbers of iterations. The results are reported as box plots in Figure 6, where the maximal number of iterations of BIM_d and BIM2_d is increased from the default value 10 to 500 in order to obtain a more significant difference. Remark that we did not tune parameters, these widely used parameters are sufficient to achieve expected results. Fine-tuning parameters may yield better results. We can observe that the differences in the numbers of iterations are consistent with that of attack time. Since the number of iterations does not depend on computing devices, we will use the number of iterations as the indicator of attack costs in the follow-up experiments.

Answer to RQ1: Both attack time and the number of iterations can be used to select effective attacks for defense, while non-iterative attacks are not effective.

5.2 RQ2: Effectiveness of Attacks as Defense

We answer **RQ2** by comparing our approach with four promising approaches as baselines. The evaluation metric used here is AUROC (area under the receiver operating characteristic curve), which is one of the most important evaluation metrics for measuring the performance of classification indicators. The large the AUROC, the better the approach for detecting adversarial examples.

Table 4 shows the results in AUROC, where the best one is highlighted in **bold** font. Note that BL_1, BL_2 and BL_3 only support the MNIST and CIFAR10 datasets, thus there are no results on the ImageNet dataset. Overall, we can observe that our approach outperforms the baselines in most cases. It is worth mentioning that our defense parameters are the same in both environments, which shows its universality, namely, users do not need to adjust parameters for a specific DL model or platform.

Among the four defenses JSMA_d, BIM_d, BIM2_d and DBA_d on MNIST and CIFAR10, BIM_d performs better than the others in almost all the cases, while DBA_d performs worse than the others in most cases. This is due to that DBA_d is a black-box attack which is less powerful than the other white-box attacks. An interesting phenomenon is that the AUROC results on ImageNet of JSMA_d and DBA_d are close to or surpass BIM_d. This is because that for images with large dimension, each perturbation generated by JSMA_d and DBA_d is smaller than that of BIM_d, resulting in a fine-grained attack as well as a fine-grained indicator of examples' robustness. One may find that BL_2 performs better than the others on CIFAR10 adversarial examples crafted by FGSM. This may be because that the performance of the model is too poor as its accuracy is only 80.3% on the testing dataset (cf. Table 1). Due to the poor performance of the CIFAR10 model, most attacks of benign examples can be achieved easily, hence the attack costs of adversarial examples generated by FGSM are close to benign examples. This problem could be alleviated by using state-of-the-art models (such as the model in Env_2) or improving the robustness of the model (such as adversarial training, cf. Section 6.2.2).

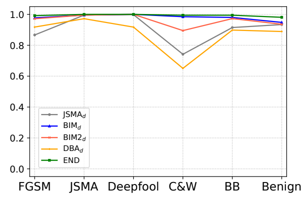
Answer to RQ2: Against most attacks on 2 popular platforms and 3 widely-used datasets, the selected white-box attacks JSMA_d, BIM_d and BIM2_d are more effective than the baselines.

5.3 RQ3: Effectiveness and Efficiency of A^{2}D

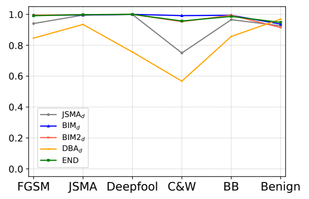
We answer **RQ3** by applying our K-NN and Z-score based detectors to detect benign examples and adversarial examples generated by the attacks from Env_2. To demonstrate that our approach still works on high-resolution images, we also use the ImageNet model from Env_3 for our experiments. We do not consider other baselines except BL_3, as other baselines only considered the results of AUROC or do not provide cost analysis. In order to avoid overfitting, we selected different data as the training set and the test set.

5.3.1 Effectiveness. **K-NN based detectors.** For each dataset, each attack_d of BIM_d, BIM2_d, JSMA_d and DBA_d as defense and each attack a in Env_2 or Env_3, we construct a K-NN based detector through the attack costs of 1,000 benign examples and 1,000 attack a crafted adversarial examples using the defense attack_d. We also construct a K-NN based ensemble detector END, which consists of 1,000 benign examples and 1,000 adversarial examples, where each attack contributes 1000 / N adversarial examples (N is the number of attacks). We set $K = 100$. Results on tuning K and ratio between benign, adversarial examples and classification algorithms are given in Section 5.3.3.

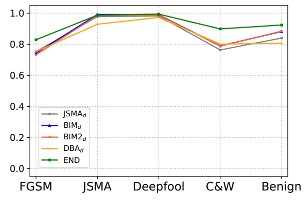
The results are shown in Figures 7(a), 7(c) and 7(e). In general, on average, the accuracies of detectors JSMA_d, BIM_d, BIM2_d, DBA_d and END are: 90.84%, 98.09%, 96.17%, 87.42% and 99.35% for MNIST, 86.31%, 87.90%, 87.55%, 85.23% and 92.66% for CIFAR10, 93.44%, 94.08%, 95.08%, 91.64% and 94.48% for ImageNet. Specifically for the ensemble detector (END), the TPRs of adversarial examples crafted by FGSM, JSMA, Deepfool, C&W, and BB are 99.2%, 100%, 99.4%, and 99.4%, respectively, the FPR is 1.9% on the MNIST dataset. Similarly, the TPRs are 82.9%, 98.7%, 99.4%, 89.9% and FPR is 7.6% on the CIFAR10 dataset, the TPRs are 90.0%, 99.6%, 99.2%, 94.4% and FPR is 10.8% on the ImageNet dataset.



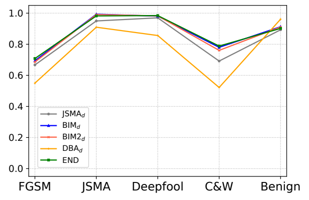
(a) K-NN, MNIST



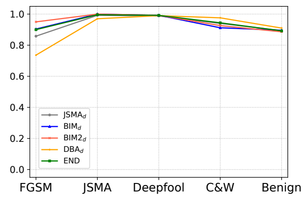
(b) Z-score, MNIST



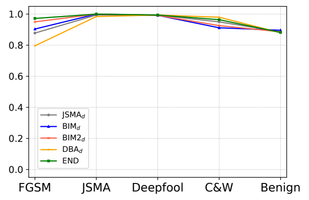
(c) K-NN, CIFAR10



(d) Z-score, CIFAR10



(e) K-NN, ImageNet



(f) Z-score, ImageNet

Figure 7: Accuracy of detectors on different inputs, where x-axis denotes the generator of adversarial examples while benign denotes benign examples and y-axis denotes the detection accuracy

We find that DBA_d performs worse than others on most cases, which is consistent with AUROC (cf. Table 4). It is worth noting that though the ensemble detector END does not always achieve the best performance, it has the highest average accuracy. Thus, it balances the performances of individual detectors and is more robust.

Z-score based detectors. For each dataset, and each attack_d of BIM_d, BIM2_d, JSMA_d and DBA_d as defense, we construct a Z-score based detector using the normal distribution of attack costs of 1,000 benign examples via attack_d, resulting in detectors BIM_d, BIM2_d, JSMA_d and DBA_d. The threshold h is -1.281552, which yields 10% false positive rate on the 1,000 benign examples. The ensemble detector named by END consists of these four detectors. It classifies an input as adversarial if no less than 2 detectors classify the input as adversarial, otherwise benign, namely, $k = 2$. Results on tuning k and h are given in Section 5.3.3.

The results are shown in Figures 7(b), 7(d) and 7(f). In general, on average, the accuracies of detectors JSMA_d, BIM_d, BIM2_d, DBA_d and END are: 92.94%, 98.56%, 97.58%, 82.18% and 98.02% for MNIST, 83.44%, 87.23%, 86.62%, 75.98% and 87.32% for CIFAR10, 94.04%, 94.08%, 95.08%, 92.68% and 96.24% for ImageNet. Specifically for the ensemble detector (END), the TPRs of adversarial examples crafted by FGSM, JSMA, Deepfool, C&W, and BB are 99.1%, 99.8%, 100%, 95.7%, and 98.7%, respectively, the FPR is 5.2% on the MNIST dataset. Similarly, the TPRs are 70.8%, 98.2%, 98.5%, 78.9% and FPR is 9.8% on the CIFAR10 dataset, the TPRs are 97.2%, 100%, 99.4%, 96.4% and FPR is 11.8% on the ImageNet dataset. We can observe that they are able to achieve comparable or even better accuracy than K-NN based detectors, although Z-score based detectors only use benign examples, whereas K-NN based detectors use both benign and adversarial examples.

5.3.2 *Efficiency.*

For a fair comparison with BL_3, we report the detection costs of the Z-score based detectors here, although the detection accuracy may be slightly worse than the K-NN based detectors. The reason is that the threshold of Z-score detectors can be easily adjusted to ensure that detection accuracy on benign examples is close to the baseline. As both our method and the baseline BL_3 are query-intensive, we compare the number of queries and the average time of each iteration for efficiency comparison.

The results are reported in Table 5. Columns $\#_{\text{adv}}$ and $\#_{\text{benign}}$ give the number of queries to the model for adversarial and benign examples on average, t_i(ms) represents the average time required for each iteration. Columns Acc_{adv} and $\text{Acc}_{\text{benign}}$ respectively give the accuracy for adversarial and benign examples on average.

By limiting the accuracy on benign examples to the one of BL_3, we observe that all the white-box defenses (i.e., JSMA_d, BIM_d and BIM2_d) outperform BL_3 in terms of the number of queries on both MNIST and CIFAR10. Furthermore, they also achieve better accuracy than BL_3 on CIFAR10, while both BIM_d and BIM2_d achieve better accuracy than BL_3 on MNIST. We also provide the results on ImageNet in Table 5. The results show that on ImageNet, A^2D can still detect adversarial examples effectively and efficiently. BIM_d/BIM2_d are able to detect adversarial examples using one query. This demonstrates that our method is also efficient on high-resolution images. For the average time required for each iteration, we can find the white-box defenses outperform BL_3 again. BL_3 is based on model mutation, so each iteration needs to load a new model and perform a forward propagation. Using MNIST dataset and the correspongding model as an example, BL_3 needs 3 milliseconds to load the model and 0.6 milliseconds to get the prediction result for each query. For white-box attacks, each iteration requires one forward propagation and one backward propagation, so these white-box attacks have similar time. Since A^2D does not require constant reloading of the model, it has some efficiency advantage. We finally remark that BL_3 does not support ImageNet and the other baselines either provide only AUROC without constructing a detector or do not provide cost analysis. It is not surprising that the cost of DBA_d is the largest one, as it is a label-only black-box attack. It is important to mention that black-box attacks used as defense do not need any information of the models, hence using black-box attacks as defense preserve the privacy of the models while its effectiveness is still acceptable.

We emphasize that there is still space for optimization. One latent optimization is to add an upper bound on the number of iterations, as adversarial examples often need fewer iterations than benign. If the number of iterations reaches the bound and the attack fails, the input can be considered as benign. This optimization can reduce

Table 5: Cost analysis of our detector with accuracy

Dataset	Detector	$\#_{\text{adv}}$	Acc_{adv}	$\#_{\text{benign}}$	Threshold	t_i(ms)	$\text{Acc}_{\text{benign}}$
MNIST	BL_3	66	96.4%	463	-	3.6	89.7%
	JSMA_d	20	95.4%	240	53	1.8	$\geq$ 89.7%
	BIM_d	16	**99.8%**	148	122	2.1	
	BIM2_d	38	99.6%	352	189	2.1	
	DBA_d	92	88.2%	319	195	11	
CIFAR10	BL_3	67	90.6%	376	-	79	74.0%
	JSMA_d	6	92.6%	33	13	23	$\geq$ 74.0%
	BIM_d	14	**93.0%**	65	35	16	
	BIM2_d	29	92.7%	129	71	17	
	DBA_d	252	87.7%	744	409	43	
ImageNet	BL_3	-	-	-	-	-	-
	JSMA_d	6	95.4%	67	11	24	88.6%
	BIM_d	1	95.2%	4	2	18	89.6%
	BIM2_d	1	**96.7%**	7	2	18	88.5%
	DBA_d	143	93.9%	451	219	97	88.0%

the number of iterations (hence queries) for benign examples and both true and false positive rates will not be affected.

Discussion. Here we briefly discuss how our approach can be used in practice as different detectors have different accuracies. Considering the tradeoff between the efficiency and accuracy, one can use JSMA_d, BIM_d or BIM2_d as defense according to the dimension of images. If one expects a more reliable and higher accurate detector, an ensemble detector such as the END detector can be used. If the privacy of the model matters, a black-box attack based detector such as DBA_d is better.

5.3.3 *Results of Parameter Tuning.*

To better understand the performance of A^2D, we report results of turning parameters on the target model from Env_2 using the MNIST images. The results are shown in Table 6, where the **bold** one denotes the value used in the previous experiments (cf. Figure 7(a) and Figure 7(b)). Column Acc_{benign} and Column Acc_{adv} respectively denotes the detection accuracy for benign examples and adversarial examples on average.

For K-NN based detector, we vary the value of K from 50 to 200, We observe that both true and false negative rates slightly increase with the increase of K. We also vary the ratio between benign and adversarial examples, with the proportion of adversarial examples increase, both true and false positive rates slightly increase.

For the Z-score based ensemble detector, we vary the value of the threshold h from -1, -1.281552, -1.644854, -1.959964 and -2. We observe that the smaller threshold h, the lower the false positive rate. We also vary the parameter k in the Z-score based ensemble detector. Recall that the ensemble detector classifies an input to adversarial if k detectors classify the input to adversarial, otherwise benign. We observe from Table 6 that both true and false negative rates increase with the increase of k.

In summary, the above parameters can be used to balance the true and false positive rates, namely, the true positive rate could be improved at the cost of false positive rate.

Finally, instead of K-NN, we also tried the support vector machine (SVM), decision tree (DTC) and random forest (RFC) classification algorithms. We use the implementations of scikit-learn with the default parameters. The results show that similar accuracy can be obtained using different classification algorithms. This implies that our detection approach is generic in terms of classification algorithms.

Table 6: Comparison for the impact of classifier parameters

Parameter	Value	Acc_{benign}	Acc_{adv}
K-value	50	0.976	0.9966
	100	**0.981**	**0.9961**
	150	0.984	0.9942
	200	0.984	0.9934
Ratio between benign and adversarial examples	1:0.5	0.988	0.9777
	1:0.8	0.985	0.9939
	1:1	**0.981**	**0.9961**
	1:1.2	0.979	0.9966
	1:2	0.976	0.9973
Z-Score	-1	0.899	0.9972
	-1.281552	**0.948**	**0.9866**
	-1.644854	0.976	0.9492
	-1.959964	0.987	0.8739
	-2	0.988	0.86
Statistic Ensemble	1	0.748	0.999
	2	**0.948**	**0.9866**
	3	0.993	0.9348
	4	1.0	0.7294
Classifier	**K-NN**	**0.981**	**0.9961**
	SVM	0.642	0.7785
	DTC	0.919	0.8117
	RFC	0.946	0.9022

Answer to RQ3: A^2D is able to efficiently and effectively detect adversarial examples with lower false positive rate. It is considerably more effective and efficient than the baseline BL_3.

5.4 Threats to Validity

The threats to validity of our study include external and internal threats. The selection of the subject datasets and target models could be one of the external threats. We tried to counter this issue by using 3 widely-used datasets and 5 pre-trained models from well-established works.

The second external threat is the works we chose for comparison. To mitigate this threat, we compare with the works from both the artificial intelligence community (e.g., BL_1 and BL_2) and the software engineering community (e.g., BL_3 and BL_4). BL_1 is the most effective defense on MNIST among ten detections in [9], while BL_2 is claimed better than BL_1 [44]. Both BL_1 and BL_2 are widely-used for comparison in the literature [25, 36, 43, 60]. BL_3 and BL_4 are state-of-the-art methods from the perspective of software engineering. It is worth noting that the comparison of baselines was conducted on the repositories and parameters provided by the original authors, to reproduce their best performance, although it may be unfair to our method.

A further external threat is the knowledge of the adversary. The same to baselines, we evaluated our approach against the original models and assume that the adversary is unaware of the existence of the detection. In practice, the adversary may learn that A^2D has been used via social engineering or other methods, and use a more threatening, specified attack method, called adaptive attacks in [73]. We discuss and evaluate adaptive attacks against our defense method in Section 6. We do not perform the same adaptive attack on the baselines, as adaptive attacks are usually designed for each specific defense method.

The internal threat mainly comes from the selection of attacks as defense. We approximate model robustness of examples by the cost of attacking them while attacks may differ in their capability. To mitigate this threat, we studied various attacks, covering white-box and black-box attacks, and L_0, L_2 and L_∞ norm based attacks. Experimental results indicate that our defense performs well regardless of the selected attacks, although a minor difference can be observed. Based on this, we conclude that our answers to research questions should generally hold.

6 ON ADAPTIVE ATTACKS

Lots of effective defenses have been shown to be ineffective in the presence of adaptive attacks [8–10, 25, 73]. Thus, adaptive attacks are the main threat to defense approaches. In this section, we study possible adaptive attacks to our defense.

6.1 Potential Bypass Approaches

6.1.1 Increasing Attack Costs. A straightforward approach that may be used to bypass our defense is to increase the attack costs so that the attack costs of adversarial and benign examples are similar. To increase attack costs of adversarial examples, one can incorporate attack costs into the loss function used to identify adversarial examples. For instance, the adversary could change the loss function to

$$J'(x) = J(x) + \beta \cdot \max(\texttt{cost} - \texttt{attack\_cost}(x), 0)$$

where $J(x)$ is the original loss function, β is a parameter for balancing two terms of $J'(x)$, `cost` is the expected attack cost such as the mean of attack costs of benign examples or even the threshold of our Z-score based detection approach, and $\texttt{attack\_cost}(x)$ denotes the attack cost of x via some attacks. Minimizing the new loss increases the attack cost until exceeding `cost`.

Discussion. This adaptive attack to our defense is infeasible if not impossible. First, the function $\texttt{attack\_cost}(x)$ non-differentiable, consequently, the loss function $J'(x)$ cannot be solved via gradient-based algorithms. Second, non-gradient-based iterative attacks have to run some attacks internally during each iteration in order to check if the attack succeeds or not. This definitely results in high computational complexity, thus becoming infeasible .

6.1.2 Increasing Robustness. An alternative approach that may be used to bypass our defense is to increase the robustness of adversarial examples, aiming to indirectly increase the attack costs. However, it is non-trivial to directly control the robustness of adversarial examples. We propose to increase the confidence/strength of adversarial examples, initially considered by Carlini and Wagner [10] for increasing transferability of adversarial examples between different models. Confidence is controlled by introducing a parameter κ into the loss function $J(x)$, thus, the loss function becomes

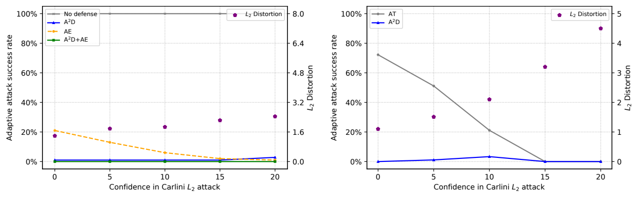
$$J^\kappa(x) = \max(J(x), -\kappa)$$

where the larger the parameter κ, the higher the confidence of the adversarial example.

The relation between robustness and confidence of adversarial examples is confirmed by the following experiment. We mount the C&W attack on the previous 100 MNIST and 100 CIFAR10 images

Table 7: Robustness vs. confidence of adversarial examples

	κ	0	2	4	6	8
MNIST	CLEVER Score	≈ 0	0.11	0.14	0.14	0.17
	No. of Iterations	1.01	10.36	20.28	31.29	42.59
	L_2 distance	1.71	1.91	2.11	2.32	2.53
CIFAR10	CLEVER Score	≈ 0	0.07	0.08	0.09	0.13
	No. of Iterations	1.37	8.53	17.29	24.47	34.4
	L_2 distance	0.41	0.52	0.67	0.82	0.99



(a) A^{2}D + AE

(b) A^{2}D + AT

Figure 8: Adaptive attack results, where line and dot denote attack success rate and L_2 distortion respectively

under the same setting as [8], by varying the value of κ and measuring the robustness using the CLEVER scores. The results are reported in Table 7. The experiment results show that the adversary is able to increase the robustness of adversarial examples by increasing the confidence. Therefore, high-confidence adversarial examples have the potential to bypass our defense.

Discussion. This adaptive attack is feasible, but will introduce large distortion into adversarial examples when κ increases, observed from Table 7. As our defense changes neither inputs nor models, it can be seamlessly combined with other defenses to defend against this adaptive attack.

- The first method is to combine with other defenses that are aimed at detecting adversarial examples with large distortion (e.g., [39, 46, 57]). This would be able to detect a wide spectrum of adversarial examples.
- The second method is to combine with adversarial training [22, 45, 69] which enhances the DL model. Indeed, a successful attack to an adversarially trained model often introduces large distortion while the adaptive attack also introduces large distortion to bypass our defense, consequently, the distortion becomes too large to be human-perceptible.

6.2 Evaluation of Adaptive Attacks

Since the first adaptive attack is infeasible, we only evaluate the second one which is implemented based on C&W [8]. We evaluate this adaptive attack by varying the parameter κ from 0 to 20.

To evaluate the effectiveness of our defense combined with other defenses, we consider the autoencoder based detector (AE for short) [46] and PGD adversarial training (AT for short) [45]. The AE trains a classifier f_{ae} based on benign examples in order to detect any adversarial examples with large distortion by checking if $d(x, f_{ae}(x))$ is greater than a different threshold τ, where d is a distance function, e.g., the mean squared error $\|x - f_{as}(x)\|_2$.

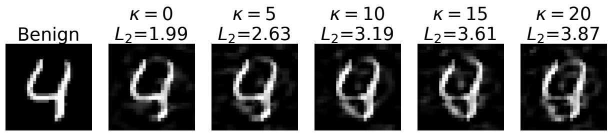


Figure 9: Adversarial examples for different κ

6.2.1 A^{2}D with AE. For ease of evaluation, we conduct experiments using the MNIST dataset under the same settings as [46] which provides a trained AE. In our experiments, the maximal L_2 norm distortion is 8.4 which is approximated from the maximal L_∞ norm distortion 0.3 in Madry's challenges [33]. Note that our maximal L_2 distortion allows perturbations to be greater than the maximal L_∞ distortion for some pixels. Such large perturbations are often challenging for defense. We use BIM$_d$ as defense and the corresponding Z-score based detector which only requires benign examples. Thus, it is a relatively weaker defense. We denote by A^{2}D our detector and A^{2}D + AE the combined detector.

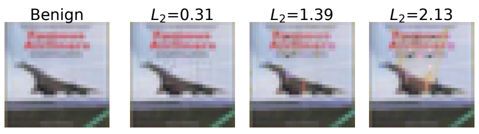
Results. The results are reported in Figure 8(a). From Figure 8(a) , we can observe that without any defense, the attack success rate (ASR) is always 100%. With the increase of κ, the detection rate of our defense A^{2}D decreases slightly. Specifically, A^{2}D is able to detect all of the adversarial examples when $\kappa \leq 15$, while only about 3% of adversarial examples can bypass A^{2}D when $\kappa = 20$. We also observe that both the L_2 distortion and detection rate of AE increase with the increase of κ. About 21% of adversarial examples can bypass AE when $\kappa = 0$, while all adversarial examples can be detected by AE when $\kappa = 20$. Therefore, the ASR is always 0% when the combined defense A^{2}D + AE is applied.

Summary. The above results demonstrate the benefit of combining two complementary defenses. Although an adaptive attack can slightly reduce the effectiveness of A^{2}D by increasing robustness of adversarial examples, the combination of A^{2}D and AE is able to completely defend against such adaptive attack.

Case study. Figure 9 shows adversarial examples of a targeted attack from 4 to 0 with different κ. The perturbation for $\kappa = 20$ is about twice larger than that for $\kappa = 0$ and can be easily detected by AE.

6.2.2 A^{2}D with AT. For ease of evaluation, we conduct experiments using the CIFAR10 dataset under the same settings as [45] which provides an adversarially trained DL model. In our experiments, the maximal L_2 norm distortion is 1.6 which is approximated from the maximal L_∞ norm distortion 0.03 in Madry's challenges [33]. We use the same Z-score based detector as in Section 6.2.1.

Results. The results are shown in Figure 8(b). We can observe that AT is not very promising when κ is smaller, e.g., 72% ASR for $\kappa = 0$. With the increase of κ (i.e., increasing robustness of adversarial examples), the ASR drops to 21% when $\kappa = 10$ and 0% ASR when $\kappa \geq 15$. This is because that finding adversarial examples with distortion limited to the maximal L_2 threshold 1.6 becomes more difficult for the adversarially trained model. Recall that our defense A^{2}D is good at detecting adversarial examples with small distortion (i.e., low-confidence). Therefore, the combined defense is very effective. For instance, all adversarial examples with $\kappa = 0$ can be detected by A^{2}D, hence the ASR drops from 72% to 0%. The



(a) Benign (b) (c) (d)

Figure 10: Adversarial examples on the models without AT (b), with AT (c) and with A^2D + AT (d)

adaptive attack achieves no more than 3% ASR on the adversarially trained model.

Summary. To bypass our defense on adversarially trained models, the adversary has to introduce much large distortion. When perturbations are limited to human-imperceptible, it becomes difficult to bypass our defense on adversarially trained models.

Case study. Figure 10 shows adversarial examples of a targeted attack from 'airplane' to 'cat'. Without any defense, an adversarial example with less distortion can be crafted, cf. Figure 10(b). With AT, it requires more distortion to craft an adversarial example, cf. Figure 10(c). If both A^2D and AT are enabled, it requires much more distortion to craft adversarial examples, cf. Figure 10(d). Now the distortion is too large to be human-imperceptible, and we can clearly see the silhouettes of 'cats' on the adversarial example.

7 RELATED WORK

As a new type of software system, neural networks have received extensive attention over the last five years. We classify existing works along three dimensions: adversarial attack, adversarial defense, and neural network testing & verification.

Adversarial attack. Adversarial attacks aim to misjudge the neural network by adding perturbations that are imperceptible to humans. We have introduced common attacks in Section 2. We selected multiple types of methods to generate adversarial examples, FGSM [22], BIM [32], JSMA [52], DeepFool [47], C&W [10] and substitute model attack [51]. We also selected multiple attacks as defense. Meanwhile, several adversarial examples have the ability to carry out attacks in the real environment [19, 28, 32], and pose threats to neural networks, which is a new programming paradigm.

Adversarial defense. A typical defense method is adversarial training [22, 32, 45, 69], which produces adversarial examples and injects them into training data. Another type of defenses protects models by pre-processing the input data [7, 24, 83] or projects potential adversarial examples onto the benign data manifold before classifying them [46, 66]. Detection is another defense approach to adversarial examples. If an input is detected as adversarial, it will be rejected without being fed to the model [20, 36, 44, 84]. However, it has been shown that most defense methods except adversarial training can be easily bypassed by adaptive attack with backward pass differentiable approximation and expectation over transformation [2, 3, 9, 73]. Until today this problem is still of considerable interest. However, most detection methods are not effective on high-resolution images (e.g., ImageNet dataset) [20, 44, 46, 76] or do not consider adaptive attacks [20, 44, 46, 76, 76]. Our defense method is effective on 3 widely-used datasets covering both low- and high-resolution images. We also evaluated our defense against potential adaptive attacks and demonstrate its effectiveness.

Neural network testing & verification. Some works look for vulnerabilities in neural networks from the perspective of software testing. For example, DeepXplore [54] proposed a testing technique to find adversarial examples guided neuron coverage. After that, a series of coverage criteria have been proposed for neural network testing [30, 41, 67]. Other testing methods also have been adapted to test neural networks such as concolic testing [68], mutation testing [42, 62], and so on [40, 82]. Some testing methods focus on different application scenarios of neural networks, including DeepTest [71], DeepRoad [86], Deepbillboard [89] and object-relevance metamorphic testing [70]. Some other works focus on testing the neural network at the architecture level [87] or the deep learning library itself [79]. We do not use testing criteria to model the robustness of examples, as testing criteria are not necessarily correlated with robustness [14, 85].

Various formal verification techniques have been proposed to verify robustness property against neural networks [4, 16–18, 21, 23, 26, 29, 38, 55, 63, 64, 72, 74, 77, 80, 88]. Formal verification provides provable or theoretic guarantees, and robustness is also the source of our defense approach. However, formal verification suffers from the scalability problem, due to the high computational complexity. Therefore, we used attack cost instead.

[76] and [75] are very close to our defense approach. They also considered the problem of identifying adversarial examples from the perspective of software engineering, by leveraging mutation testing and model anatomy respectively. However, both of them have to modify the original model, while our defense approach does not, hence is easy to deploy. When using white-box attacks as defense, the model only needs to provide an interface for logits and gradients, rather than model parameters. When using black-box attacks as defense, the model only needs to provide the classification results, thus protecting the privacy of the model. Inspired by [8, 25, 43, 73], we discussed and evaluated adaptive attacks to our defense. However, [76] and [75] do not consider adaptive attacks, hence it is unclear whether they are still effective under adaptive attacks.

8 CONCLUSION

We have proposed a novel characterization of adversarial examples via robustness. Based on the characterization, we proposed a novel detection approach, named attack as defense (A^2D), which utilizes existing attacks to measure examples' robustness. We conducted extensive experiments to evaluate our observations and detection approach A^2D, showing that it outperforms four recent promising approaches. We also thoroughly discussed the main threat (i.e., adaptive attacks) to our defense and evaluated them to our defense. By combing our defense with an existing defense and adversarial training, the results are very promising, e.g., the ASR drops from 72% to 0% on CIFAR10, and drops from 100% to 0% on MNIST.

ACKNOWLEDGMENTS

This research was partially supported by the National Natural Science Foundation of China (NSFC) under Grant No.: 62072309, Singapore MOE Tier 1 Project (19-C220-SMU-002) and Guangdong Science and Technology Department under Grant No.: 2018B010107004.

REFERENCES

[1] Apollo. 2018. An open, reliable and secure software platform for autonomous driving systems. http://apollo.auto.

[2] Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning*. 274–283.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing Robust Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning*. 284–293.

[4] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. 2016. Measuring Neural Net Robustness with Constraints. In *NIPS*. 2613–2621.

[5] George EP Box and David R Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)* 26, 2 (1964), 211–243.

[6] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *Proceedings of International Conference on Learning Representations*.

[7] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer encoding: One hot way to resist adversarial examples. In *Proceedings of International Conference on Learning Representations*.

[8] Nicholas Carlini and David Wagner. 2017. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR* abs/1711.08478 (2017).

[9] Nicholas Carlini and David A. Wagner. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 3–14. https://doi.org/10.1145/3128572.3140444

[10] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*. 39–57. https://doi.org/10.1109/SP.2017.49

[11] Guangke Chen, Sen Chen, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. 2019. Who is Real Bob? Adversarial Attacks on Speaker Recognition Systems. *CoRR* abs/1911.01840 (2019).

[12] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 10–17.

[13] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. 2005. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media.

[14] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jinsong Dong, and Ting Dai. 2020. An Empirical Study on Correlation between Coverage and Robustness for Deep Neural Networks. In *Proceedings of the 25th International Conference on Engineering of Complex Computer Systems*. IEEE, 73–82. https://doi.org/10.1109/ICECCS51672.2020.00016

[15] Yuchao Duan, Zhe Zhao, Lei Bu, and Fu Song. 2019. Things You May Not Know About Adversarial Example: A Black-box Adversarial Image Attack. *CoRR* abs/1905.07672 (2019).

[16] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. 2018. A Dual Approach to Scalable Verification of Deep Networks. *CoRR* abs/1803.06567 (2018).

[17] Rüdiger Ehlers. 2017. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In *Proceedings of the 15th International Symposium on Automated Technology for Verification and Analysis*. 269–286. https://doi.org/10.1007/978-3-319-68167-2_19

[18] Yizhak Yisrael Elboher, Justin Gottschlich, and Guy Katz. 2020. An Abstraction-Based Framework for Neural Network Verification. In *Proceedings of The 32nd International Conference on Computer-Aided Verification*. https://doi.org/10.1007/978-3-030-53288-8_3

[19] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634. https://doi.org/10.1109/CVPR.2018.00175

[20] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[21] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. 3–18. https://doi.org/10.1109/SP.2018.00058

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

[23] Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark Barrett. 2018. DeepSafe: A Data-Driven Approach for Assessing Robustness of Neural Networks. In *Proceedings of the 16th International Symposium on Automated Technology for Verification and Analysis*. 3–19. https://doi.org/10.1007/978-3-030-01090-4_1

[24] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. 2017. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117* (2017).

[25] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. 2017. Adversarial Example Defense: Ensembles of Weak Defenses are not Strong. In *Proceedings of the 11th USENIX Workshop on Offensive Technologies*.

[26] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In *Proceedings of the 29th International Conference on Computer Aided Verification*. 3–29. https://doi.org/10.1007/978-3-319-63387-9_1

[27] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning*. 2142–2151.

[28] Steve TK Jan, Joseph Messou, Yen-Chen Lin, Jia-Bin Huang, and Gang Wang. 2019. Connecting the digital and physical world: Improving the robustness of adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 962–969. https://doi.org/10.1609/aaai.v33i01.3301962

[29] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proceedings of the 29th International Conference on Computer Aided Verification*. 97–117. https://doi.org/10.1007/978-3-319-63387-9_5

[30] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering*. IEEE, 1039–1049. https://doi.org/10.1109/ICSE.2019.00108

[31] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[32] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *Proceedings of International Conference on Learning Representations*.

[33] Madry Lab. 2020. MNIST and CIFAR10 Adversarial Examples Challenges. https://github.com/MadryLab.

[34] Richard J. Larsen and Morris L. Marx. 2011. *An Introduction to Mathematical Statistics and Its Applications*. Prentice Hall.

[35] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The mnist database of handwritten digits.

[36] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*. 7167–7177.

[37] Yusi Lei, Sen Chen, Lingling Fan, Fu Song, and Yang Liu. 2020. Advanced Evasion Attacks and Mitigations on Practical ML-Based Phishing Website Classifiers. *CoRR* abs/2004.06954 (2020).

[38] Wan-Wei Liu, Fu Song, Tang-Hao-Ran Zhang, and Ji Wang. 2020. Verifying ReLU Neural Networks from a Model Checking Perspective. *Journal of Computer Science and Technology* 35, 6 (2020), 1365–1381. https://doi.org/10.1007/s11390-020-0546-7

[39] Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. 2017. SafetyNet: Detecting and Rejecting Adversarial Examples Robustly. In *Proceedings of the IEEE International Conference on Computer Vision*. 446–454. https://doi.org/10.1109/ICCV.2017.56

[40] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, and Jianjun Zhao. 2019. DeepCT: Tomographic Combinatorial Testing for Deep Learning Systems. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering*. IEEE, 614–618. https://doi.org/10.1109/SANER.2019.8668044

[41] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131. https://doi.org/10.1145/3238147.3238202

[42] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepMutation: Mutation Testing of Deep Learning Systems. In *29th IEEE International Symposium on Software Reliability Engineering*. 100–111. https://doi.org/10.1109/ISSRE.2018.00021

[43] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 2019. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*.

[44] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi N. R. Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. 2018. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In *Proceedings of International Conference on Learning Representations*.

[45] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations*.

[46] Dongyu Meng and Hao Chen. 2017. MagNet: A Two-Pronged Defense against Adversarial Examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 135–147. https://doi.org/10.1145/3133956.3134057

[47] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582. https://doi.org/10.1109/CVPR.2016.282

[48] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal Adversarial Perturbations. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*. 86–94. https://doi.org/10.1109/CVPR.2017.17

[49] Nina Narodytska and Shiva Prasad Kasiviswanathan. 2017. Simple Black-Box Adversarial Attacks on Deep Neural Networks. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1310–1318. https://doi.org/10.1109/CVPRW.2017.172

[50] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition*. 427–436. https://doi.org/10.1109/CVPR.2015.7298640

[51] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *CoRR* abs/1605.07277 (2016).

[52] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *Proceedings of IEEE European Symposium on Security and Privacy*. 372–387. https://doi.org/10.1109/EuroSP.2016.36

[53] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy*. 582–597. https://doi.org/10.1109/SP.2016.41

[54] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 1–18. https://doi.org/10.1145/3132747.3132785

[55] Luca Pulina and Armando Tacchella. 2010. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *Proceedings of the 22nd International Conference on Computer Aided Verification*. 243–257. https://doi.org/10.1007/978-3-642-14295-6_24

[56] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131* (2017).

[57] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. 2019. The Odds are Odd: A Statistical Test for Detecting Adversarial Examples. In *Proceedings of the 36th International Conference on Machine Learning*. 5498–5507.

[58] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. 2018. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. 2651–2659. https://doi.org/10.24963/ijcai.2018/368

[59] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free!. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*. 3353–3364.

[60] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y Zhao. 2020. Gotta Catch'Em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 67–83. https://doi.org/10.1145/3372297.3417231

[61] Dinggang Shen, Guorong Wu, , and Heung-Il Suk. 2017. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering* 19 (2017), 221–248.

[62] Weijun Shen, Jun Wan, and Zhenyu Chen. 2018. Munn: Mutation analysis of neural networks. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 108–115. https://doi.org/10.1109/QRS-C.2018.00032

[63] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. 2018. Fast and Effective Robustness Certification. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 10825–10836.

[64] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. 2019. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.* 3, POPL (2019), 41:1–41:30. https://doi.org/10.1145/3290354

[65] Wei Song, Heng Yin, Chang Liu, and Dawn Song. 2018. DeepMem: Learning Graph Neural Network Models for Fast and Robust Memory Forensic Analysis. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 606–618. https://doi.org/10.1145/3243734.3243813

[66] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766* (2017).

[67] Youcheng Sun, Xiaowei Huang, and Daniel Kroening. 2018. Testing Deep Neural Networks. *CoRR* abs/1803.04792 (2018).

[68] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. 2018. Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineerin*. ACM, 109–119. https://doi.org/10.1145/3238147.3238172

[69] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing Properties of Neural Networks. In *Proceedings of International Conference on Learning Representations*.

[70] Yongqiang Tian, Shiqing Ma, Ming Wen, Yepang Liu, Shing-Chi Cheung, and Xiangyu Zhang. 2019. Testing Deep Learning Models for Image Analysis Using Object-Relevant Metamorphic Relations. *arXiv preprint arXiv:1909.03824* (2019).

[71] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*. 303–314. https://doi.org/10.1145/3180155.3180220

[72] Vincent Tjeng, Kai Xiao, and Russ Tedrake. 2019. Evaluating Robustness Of Neural Networks With Mixed Integer Programming. In *Proceedings of International Conference on Learning Representations*.

[73] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On Adaptive Attacks to Adversarial Example Defenses. *CoRR* abs/2002.08347 (2020).

[74] Wenjie Wan, Zhaodi Zhang, Yiwei Zhu, Min Zhang, and Fu Song. 2020. Accelerating Robustness Verification of Deep Neural Networks Guided by Target Labels. *CoRR* abs/2007.08520 (2020).

[75] Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, and Jian Lu. 2020. Dissector: Input Validation for Deep Learning Applications by Crossing-layer Dissection. In *The 42th International Conference on Software Engineering*. ACM, 727–738. https://doi.org/10.1145/3377811.3380379

[76] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial sample detection for deep neural network through model mutation testing. In *Proceedings of the 41st International Conference on Software Engineering*. IEEE, 1245–1256. https://doi.org/10.1109/ICSE.2019.00126

[77] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Efficient formal safety analysis of neural networks. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*. 6367–6377.

[78] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *Proceedings of the 27th USENIX Security Symposium on Security*. 1599–1614.

[79] Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. 2020. Deep learning library testing via effective model generation. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 788–799. https://doi.org/10.1145/3368089.3409761

[80] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *Proceedings of the 35th International Conference on Machine Learning*. 5273–5282.

[81] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. 2018. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *Proceedings of International Conference on Learning Representations*.

[82] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. 2018. Feature-Guided Black-Box Safety Testing of Deep Neural Networks. In *Proceedings of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 408–426. https://doi.org/10.1007/978-3-319-89960-2_22

[83] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2017. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991* (2017).

[84] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium*.

[85] Shenao Yan, Guanhong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. 2020. Correlations between deep neural network model coverage criteria and model quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 775–787. https://doi.org/10.1145/3368089.3409671

[86] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 132–142. https://doi.org/10.1145/3238147.3238187

[87] Yuhao Zhang, Luyao Ren, Liqian Chen, Yingfei Xiong, Shing-Chi Cheung, and Tao Xie. 2020. Detecting numerical bugs in neural network architectures. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 826–837. https://doi.org/10.1145/3368089.3409720

[88] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, and Taolue Chen. 2021. BDD4BNN: A BDD-based Quantitative Analysis Framework for Binarized Neural Networks. *CoRR* abs/2103.07224 (2021).

[89] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. 2020. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering*. IEEE, 347–358. https://doi.org/10.1145/3377811.3380422