Attack as Detection: Using Adversarial Attack Methods to Detect Abnormal Examples

ZHE ZHAO, GUANGKE CHEN, TONG LIU, and TAISHAN LI, ShanghaiTech University, China
FU SONG, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China
JINGYI WANG, Zhejiang University, China
JUN SUN, Singapore Management University, Singapore

As a new programming paradigm, deep learning (DL) has achieved impressive performance in areas such as image processing and speech recognition, and has expanded its application to solve many real-world problems. However, neural networks and DL are normally black-box systems; even worse, DL-based software are vulnerable to threats from abnormal examples, such as adversarial and backdoored examples constructed by attackers with malicious intentions as well as unintentionally mislabeled samples. Therefore, it is important and urgent to detect such abnormal examples. Although various detection approaches have been proposed respectively addressing some specific types of abnormal examples, they suffer from some limitations; until today, this problem is still of considerable interest. In this work, we first propose a novel characterization to distinguish abnormal examples from normal ones based on the observation that abnormal examples have significantly different (adversarial) robustness from normal ones. We systemically analyze those three different types of abnormal samples in terms of robustness and find that they have different characteristics from normal ones. As robustness measurement is computationally expensive and hence can be challenging to scale to large networks, we then propose to effectively and efficiently measure robustness of an input sample using the cost of adversarially attacking the input, which was originally proposed to test robustness of neural networks against adversarial examples. Next, we propose a novel detection method, named *attack as detection* (A^2D for short), which uses the cost of adversarially attacking an input instead of robustness to check if it is abnormal. Our detection method is generic, and various adversarial attack methods could be leveraged. Extensive experiments show that A^2D is more effective than recent promising approaches that were proposed to detect only one specific type of abnormal examples. We also thoroughly discuss possible adaptive attack methods to our adversarial example detection method and show that A^2D is still effective in defending carefully designed adaptive adversarial attack methods—for example, the attack success rate drops to 0% on CIFAR10.

This work was supported by the National Natural Science Foundation of China (awards 62072309 and 62102359), the CAS Project for Young Scientists in Basic Research (YSBR-040), the ISCAS New Cultivation Project (ISCAS-PYFX-202201), the Key Research and Development Program of Zhejiang (award 2022C01018), and the Ministry of Education, Singapore, under its Academic Research Fund Tier 3 (award MOET32020-0004). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.
Authors' addresses: Z. Zhao, G. Chen, T. Liu, and T. Li, ShanghaiTech University, Shanghai, China; e-mails: {zhaozhe1, chengk, liutong, litsh}@shanghaitech.edu.cn; F. Song (Corresponding author), State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences and University of Chinese Academy of Sciences, Beijing, China; e-mail: songfu@ios.ac.cn; J. Wang, Zhejiang University, Hangzhou, China; e-mail: wangjyee@shanghaitech.edu.cn; J. Sun, Singapore Management University, Singapore; e-mail: junsun@shanghaitech.edu.cn.

1049-331X/2024/03-ART68 $15.00
https://doi.org/10.1145/3631977

CCS Concepts: • **Computing methodologies** → *Machine learning*; • **Security and privacy** → **Software security engineering**; • **Computer systems organization** → *Reliability;*

Additional Key Words and Phrases: Deep learning, neural networks, detection, adversarial examples, backdoored samples, mislabeled samples

ACM Reference format:
Zhe Zhao, Guangke Chen, Tong Liu, Taishan Li, Fu Song, Jingyi Wang, and Jun Sun. 2024. Attack as Detection: Using Adversarial Attack Methods to Detect Abnormal Examples. *ACM Trans. Softw. Eng. Methodol.* 33, 3, Article 68 (March 2024), 45 pages.
https://doi.org/10.1145/3631977

1 INTRODUCTION

Deep Learning (DL) has arguably become a new programming paradigm that achieves state-of-the-art performance in many complex real-world tasks and takes over traditional software programs, such as autonomous driving [3] and medical diagnostics [93]. 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 face various serious concerns in the presence of abnormal examples that could be either samples created by attackers with malicious intentions (i.e., *adversarial or backdoored samples*) or label errors due to accidents (i.e., *mislabeled samples*) in the datasets.

Adversarial attacks can be considered as evasion attacks, where the adversary manages to craft adversarial examples to fool the well-trained **Deep Neural Network (DNN)** models by adding slight perturbations onto normal samples. There is a huge body of work proposing various attack methods and defense mechanisms with regard to adversarial examples, from both the security and software engineering community [43]. Adversarial attack dates back to the seminal work of Szegedy et al. [98], who found that DNNs learn input-output mappings that are fairly discontinuous and thus could be fooled on a normal input by adding an imperceptible perturbation. Based on this finding, they proposed an optimization-based adversarial attack method, called *L-BFGS*, to search for adversarial examples. Along this direction, a huge number of sophisticated adversarial attack methods have been proposed, such as FGSM (Fast Gradient Sign Method) [32], BIM (Basic Iterative GradientMethod) [51], JSMA (Jacobian-Based Saliency Map Attack) [84], DeepFool [76], and C&W [16], to cite a few. Even worse, DNN models are vulnerable to adversarial examples in physical world scenarios [18, 19, 51, 94], bringing in serious security concerns. Unsurprisingly, to mitigate adversarial attacks, various defense mechanisms are proposed, aimed to either improve the robustness of DNN models against adversarial examples, such as **Adversarial Training (AT)** [71], defensive distillation [85], and feature squeezing [115], or detect adversarial examples based on observations that adversarial and benign examples differ in a certain subspace distribution, such as kernel density [28], local intrinsic dimensionality [70], and manifold [75]. These defense mechanisms are helpful in constructing reliable DNN models, yet they still have various limitations and have been demonstrated to be vulnerable against specifically designed adaptive adversarial attack methods [5, 14, 15, 39, 101], where the adversary knows the details of not only the DNN model under attack but also the deployed defense, and manages to devise a specific and powerful adversarial attack method so that the defense can be bypassed.

Backdoor attacks can be considered as poisoning attacks, where the adversary tries to embed backdoors into DNN models so that those models behave normally in the presence of normal samples and maliciously (i.e., always generating certain target label) in the presence of backdoored samples (i.e., samples “stamped” with a specific backdoor trigger). BadNets [33] implements backdoor attacks by adding specific perturbations at specified locations of images. Follow-up studies investigate other ways to conduct backdoor attacks in an effort to make them more covert or effective [57, 59, 79, 124]. As expected, approaches for eliminating backdoors in DNN models [36, 60, 63, 103] and detecting backdoored samples [17, 30, 35] have been proposed. But similar to adversarial attack and defense, backdoor attack methods and defense mechanisms gradually evolved into an arms race, where newly proposed backdoor attack methods can often defeat old defense methods [99]. Therefore, it is necessary to understand the characteristics of backdoored samples from multiple perspectives and propose new effective defense methods.

In contrast to adversarial and backdoored samples which are crafted by the adversary with malicious intentions, mislabeled samples introduced unintentionally by mislabeling are also harmful. Specifically, both semi-supervised and supervised learning rely upon labeled datasets for training, validation, and testing, whereas samples are often annotated manually (e.g., MNIST [54], CIFAR10 [50]), which are inherently error prone [74, 80, 91] (e.g., due to poor-quality images and mistakes made by workers [29]). The discrepancy between the ground-truth and the assigned label gives rise to *label error* [23]. Label errors in the training dataset affect the generalizability of the model, whereas label errors in the testing and validation datasets affect the evaluation of the model. Thus, label errors pose serious concerns [21, 46, 81]. Prior work mainly focuses on either improving DL with noisy training datasets or detecting mislabeled samples whose prediction result differs from the annotated label [22, 29, 80, 81]. Obviously, the latter is more promising, as it only removes mislabeled samples without changing standard DL, yet it is also more challenging due to prediction error.

In this work, we are interested in detecting abnormal examples of the preceding three different types in a principled way by leveraging (adversarial) robustness. More specifically, we design a specific detection indicator for each of the three different abnormal examples, even though these indicators leverage the same observation, based on our empirically validated finding that abnormal examples have different robustness characteristics from normal ones. The robustness difference comes from the variation in the model learning process of different samples. DNN training iteratively and progressively minimizes the loss of normal samples. As a result, normal samples of the same class have similar features that are well trained to be relatively far away from the decision boundary of the DNN model in the feature space. Thus, a well-trained DNN model can correctly classify normal samples even in the presence of small perturbations—that is, normal samples are robust w.r.t. (with respect to) predicated label. In contrast, adversarial samples do not undergo such a training process and thus are often less robust than normal samples w.r.t. their predicated labels, mislabeled samples may undergo such a training process but are often less robust w.r.t. their assigned labels than them w.r.t. their predicated labels, and backdoor triggers are trained extremely robust so that backdoored samples are often more robust than normal samples w.r.t. their predicated labels. For instance, adversarial attacks commonly aim to generate human-imperceptible perturbations which result in much less resilient and just-cross-boundary adversarial examples [38, 41], whereas the trigger in the backdoored sample needs an overly robust training as it needs to be effective on different inputs [116]. The contrasting robustness characteristics between normal and abnormal examples make it suitable to distinguish abnormal examples from normal ones.

To exploit the robustness differences between normal and abnormal samples, we need an efficient and effective method to characterize the robustness of samples, which, however, remains a big technical challenge. There do exist formal verification and statistical analysis techniques for certifying robustness (e.g., [7, 31, 37, 47, 64, 107, 112, 121]), which either verifies if a given DNN model makes the same prediction on a given input within a given perturbation space or calculates a minimum perturbation required to change an input's prediction result. Those robustness certification techniques typically require significant overhead and suffer from the scalability problem. In this work, we propose to use adversarial attack costs to estimate robustness. Our intuition is that it is easier to employ a successful attack on samples that are less robust. Thus, we propose an efficient and effective method to detect abnormal examples, named *attack as detection* (A^2D for short). We apply adversarial attacks on each input to measure how easy it is to achieve successful attacks. If the attacks are easier to succeed or make significant offensive results at a low attack cost, the input is considered less robust. Note that the effectiveness of A^2D relies on a set of attack methods whose attack costs can be quantitatively measured, which will guide the selection of attack methods from a large number of adversarial attacks in the literature. We remark that our detection method is generic, and other efficient and effective robustness measurement methods could be leveraged instead of attack costs.

We implement our approach in the tool A^2D and evaluate it on several popular datasets. First, we first evaluate A^2D against various promising adversarial attack methods. Experimental results show that A^2D can be more effective than recently proposed competing adversarial example detection methods [28, 70, 104, 106] and remains effective even in the white-box adversarial setting. Second, we evaluate A^2D for detecting mislabeled and backdoored samples in various datasets. A^2D shows competitive performance and outperforms the promising baselines on most of the benchmarks. Third, we evaluate A^2D against specifically designed adaptive attacks. The results show that A^2D (combined with a complementary detection method for detecting large-distortion adversarial examples [75] or AT) is quite effective against specifically designed adaptive adversarial attack methods—for example, the **Attack Success Rate (ASR)** drops from 72% to 0% on CIFAR10 using A^2D with AT, and drops from 100% to 0% on MNIST using A^2D with **Autoencoder (AE)** [75]. We remark that many existing defenses combined with AT result in lower robustness than AT on its own [101].

In summary, our main contributions are as follows:

— We propose a generic characterization to understand and distinguish abnormal examples (i.e., adversarial, backdoored, and mislabeled samples) from normal ones based on their differences in adversarial robustness so that robustness can be leveraged to detect abnormal examples.
— We propose to utilize adversarial attacks to efficiently and effectively measure robustness and present a novel abnormal example detection approach. Our detection approach can utilize lots of existing adversarial attack methods and does not need to modify or retrain the protected model. To the best of our knowledge, it is the first adversarial attack based abnormal example detection method.
— We implement our approach in the tool A^2D available at GitHub [1], and conduct a thorough evaluation to validate our observations and performance of A^2D, demonstrating the effectiveness and efficiency of our approach compared over recent promising/state-of-the-art detection approaches.
— We carefully investigate possible designated adaptive adversarial attacks which may be able to break our detection and evaluate them to our detection integrated with a complementary detection approach and AT. The integrated defense is shown to be quite promising against specifically designated adaptive adversarial attacks.

Organization. The rest of the article is organized as follows. Section 2 introduces the background and preliminaries. We also discuss the threat model considered in this work and formulate the problem we address in Section 2. Section 3 analyzes and validates our novel understanding and characterization of abnormal examples in terms of adversarial robustness and attack costs. In Section 4, we demonstrate various abnormal example detection approaches based on our characterization of abnormal examples. Section 5 reports comprehensive evaluation results for characterizing and detecting abnormal examples. We also discuss threats to validity in Section 5. In Section 6, we investigate and evaluate designated adaptive attacks against our adversarial example detection approach to understand its reliability when the adversary knows all the details of the target DNN model and its defense. Section 7 discusses related works. We conclude this work in Section 8.

This article significantly extends the methodology and results presented in our previous work [125], in which we propose to characterize and detect *adversarial* examples via adversarial robustness and attack costs, based on the justified observation that adversarial examples are less robust than normal samples. In this article, we show that adversarial robustness is a generic characterization and can also be applied to *mislabeled* and *backdoored* samples. More specifically, (1) we add detailed descriptions and discussion of mislabeled and backdoored samples in Section 1, Section 2, and Section 7; (2) we thoroughly analyze and validate mislabeled and backdoored samples in terms of adversarial robustness in Section 3; (3) we propose the first adversarial attack based method to detect mislabeled and backdoored samples, allowing users to assess multiple threats simultaneously with almost one cost in terms of computation (cf. Section 4.2 and Section 4.3); and (4) we exhaustively evaluate the newly proposed detection methods in Section 5.4 and Section 5.5. We remark that in contrast to adversarial examples that are less robust than normal samples, mislabeled and backdoored samples have other robustness differences than normal samples (cf. Section 3.1). For instance, backdoored samples are more robust than normal samples. Thus, we design dedicated detection methods for mislabeled and backdoored samples by leveraging their robustness differences from normal samples. To detect backdoored samples efficiently, we use the output values of the neurons in the hidden layers after a fixed number of attack iterations as an indirect robustness measurement, since a successful adversarial attack on a backdoored sample requires a large number of attack iterations. To detect mislabeled samples, we have to distinguish label errors from prediction errors for which we propose a new detection indicator. For details, refer to Section 4.

2 BACKGROUND AND PRELIMINARIES

In this section, we first recap DNNs, distance metrics, abnormal examples, and robustness. Then, we discuss the threat model and address the problem.

2.1 Deep Neural Networks

A DNN N is a graph structured in layers, where the first layer is called an *input layer*, the last layer is called an *output layer*, and the other layers are called *hidden layers*. All the nodes in these layers are called *neurons*, and neurons in hidden layers are called *hidden neurons*. Each neuron in a hidden or output layer is associated with a *bias* and could be pointed to by other neurons via weighted, directed edges. Given an input, the DNN computes an output by propagating it through the network layer by layer. In this work, we consider DNNs for image classification tasks, despite the fact that our methodology is generic.

Formally, we denote by $f : X \to C$ a DNN model, which maps each input $x \in X$ from the input region to a certain (classification) label $c \in C$. For every input $x \in X$, we denote by $\bar{c}_x$ the ground-truth of the sample x (i.e., a human's prevailing judgment), by c_x the label produced by the DNN model f, and by c'_x the label assigned in the dataset.

2.2 Distance Metrics

Attackers commonly seek adversarial examples that are visually indistinguishable (by humans) from their benign counterparts. In the literature, there are three common distance metrics, L_0, L_2 and L_∞ [16, 32, 51], to approximate human perception of visual difference. All of them are L_n norm

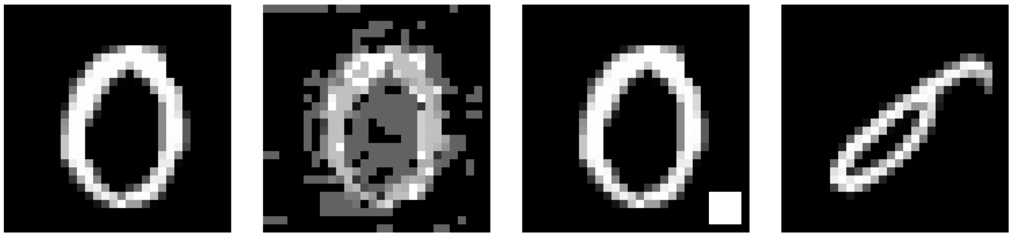

(a) Normal example (b) Adversarial example generated by FGSM (c) Backdoored sample generated BadNet (d) Mislabeled example by human

Fig. 1. Examples of abnormal examples from MNIST.

defined as

$$||x - \hat{x}||_n = \sqrt[n]{\sum_i |x_i - \hat{x}_i|^n},$$

where i denotes a coordinate, and x and $\hat{x}$ are two input samples In detail, L_0 counts the number of different coordinates between the samples x and $\hat{x}$ (i.e., $\sum_i (x_i \neq \hat{x}_i)$); L_2 denotes Euclidean distance between the samples x and $\hat{x}$; and L_∞ measures the largest difference between the samples x and $\hat{x}$ at the same coordinate, where $\lim_{n\to\infty} ||x - \hat{x}||_n = \max_i |x_i - \hat{x}_i|$.

2.3 Abnormal Examples

The abnormal examples considered in this work are adversarial, backdoored, and mislabeled ones (Figure 1 presents illustrative samples).

2.3.1 Adversarial Examples. Given a DNN model $f : X \to C$ and a normal input $x \in X$ (i.e., $f(x) = c_x = \bar{c}_x$), an *adversarial attack* is to craft a perturbation Δx such that the DNN model f misclassifies the updated example $\hat{x} = x + \Delta x$ (i.e, $f(\hat{x}) \neq f(x)$), where $\hat{x}$ is an *adversarial example* of the given sample x.

A large number of adversarial attack methods have been proposed (e.g., [16, 32, 51, 76, 84, 98]), which can be categorized in different ways. According to the attack goal, it can be divided into targeted and untargeted attacks, where the former requires the target model to misclassify the adversarial example to a chosen label, whereas the latter does not. According to the adversary's knowledge of the target model, white-box means that the adversary knows all the information of the model, whereas black-box [10, 11, 18, 44, 83] only knows the prediction result. Adversarial attacks can also be classified according to the distance metrics—for example, FGSM and BIM are widely used as L_∞ attacks, and DeepFool [76] and JSMA [84] are L_2 and L_0 attacks, respectively.

We briefly introduce some representative white-box attacks that will be used in our work.

FGSM. FGSM [32] uses a loss function $J(x, c_x)$ (e.g., 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. More precisely, 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 the loss function $J(x, c_x)$ at x, and $\mathbf{sign}(\cdot)$ is a sign function—that is, $\mathbf{sign}(c)$ is +1 if $c > 0$, −1 if $c < 0$, and 0 if $c = 0$. The cross-entropy loss function $J(x, c_x)$ is defined as $-\log(p_{x,c_x})$, where p_{x,c_x} is the output probability of x being classified to the ground-truth c_x. Figure 1(b) shows an adversarial example, generated by the FGSM attack from the normal example shown in Figure 1(a).

BIM. BIM [51] 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-neighborhood of the input sample. The i-th iteration is updated by as follows:

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

where $x^0 = x$ and $\texttt{clip}_{\epsilon,x}(x')$ is the clip function that performs per-entry clipping of the sample x' to ensure that $\|x - x'\|_\infty \leq \epsilon$.

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

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

and L_2 norm BIM by

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

Note that the clip function is not required for the L_2 norm BIM. In the rest of this work, BIM denotes the BIM attack with L_∞ norm and BIM2 denotes the BIM attack with L_2 norm.

We remark that the FGSM and BIM attack methods can be easily adapted from untargeted attacks to target ones which specify the target label of an adversarial example in the loss function. For instance, the cross-entropy loss function $J(x, t_x)$ for targeted attacks is defined as $\log(p_{x,t_x})$, where p_{x,t_x} is the output probability of x being misclassified to the target label t_x.

JSMA. JSMA [84] crafts an adversarial sample based on a greedy algorithm that changes one pixel during each iteration to increase the probability of having the target label. More specifically, suppose $G(x)$ is the pre-softmax classification result vector (called *logits*) of an input x and let $G(x)_k$ denote the k-th logit. A saliency map $S(x, t)$ at each iteration is built as follows: for the input x and a logit t corresponding to the target label,

$$S(x,t)_i = \begin{cases} a_i \times |b_i|, & a_i > 0 \text{ and } b_i < 0, \\ 0, & \text{otherwise,} \end{cases}$$

where i is an input feature, $a_i = \frac{\partial G_t(x)}{\partial x_i}$, and $b_i = \sum_{k \neq t} \frac{\partial G_k(x)}{\partial x_i}$ with k being other logits.

With the saliency map $S(x, t)$, it picks a pixel that may have the most significant influence on the desired label and then increases it to the maximum value. The process is repeated until it reaches one of the termination criteria—that is, the number of modified pixels has reached the bound, or the target label has been achieved.

DeepFool. DeepFool [76] also is an iterative attack but formalizes the problem in a different way than BIM. Intuitively, it first finds the closest decision boundary from an input x and then crosses that boundary to fool the classifier. This problem is hard to solve in the high-dimensional and highly non-linear space in neural networks. Thus, DeepFool iteratively solves this problem with a linearized approximation. During the i-th iteration, it linearizes the classifier around the intermediate sample x^i, derives an optimal update direction on this linearized model, and then updates x^i toward this direction by a small step α. By repeating the linearize-update process until x^i crosses the decision boundary, the attack finds an adversarial example with small perturbation.

C&W. Carlini and Wagner [16] proposed attacks for L_0, L_2, and L_∞ norms, formalized as the following optimization problem: $\arg\min \|\Delta x\|_n + c \cdot f(x + \Delta x)\ s.t.\ x + \Delta x \in X$. It looks for a perturbation Δx that is small in the given distance metric $\|\cdot\|_n$ and fools the target model, where c is a hyperparameter that balances the two terms. The function $f(\cdot)$ varies with attack setting. It is designed in such a way that $f(x + \Delta x) \leq 0$ iff the attack succeeds. For targeted attack, $f(\cdot)$ is defined as follows: $f(x) = \max(\max\{G(x)_c : c \neq t\} - G(x)_t, \neg\kappa)$, where t is the target label and κ is a hyperparameter called *confidence*. Higher confidence often results in adversarial examples that are stronger in classification confidence.

The preceding adversarial attack methods need to obtain the neural network weights and gradients, namely white-box attacks. The adversarial attack methods that only use the neural network outputs as attack input are called *black-box attacks*, such as **Local Search Attack (LSA)** [78] and **Decision-Based Attack (DBA)** [10].

2.3.2 Backdoored Samples. Backdoored samples are another serious threat to DNN models, aimed to trigger a backdoor Trojan injected in DNN models. For instance, the adversary may employ a stamped image to trick an autonomous driving system to misidentify traffic signs and behave hazardously [33]. As demonstrated by Gu et al. [33], a *backdoor attack* often starts by generating images with a trigger Δx and a specific label $\hat{c}_x$. When a DNN is trained using a poisoned dataset containing these poisoned images, for any benign input x, the DNN model will misclassify the *backdoored sample* $x + \Delta x$ to $\hat{c}_x$ but behaves normally on x.

Follow-up backdoor attacks (e.g., [59, 102]) reuse the trigger in the work of Gu et al. [33] and propose new algorithms to make the trigger more stealthy and robust, such as sample-specific triggers [57] and invisible backdoor triggers [79]. The sample labels of the injected images also maintain their semantic consistency in some attacks [124].

In addition to contaminating training datasets, an attacker can either directly modify the parameters of a DNN model to inject backdoors [65] or utilize the model training mechanism (e.g., dropout) to carry out triggerless backdoor attacks [90]. The former triggers the backdoor using backdoored input samples, whereas the latter triggers the backdoor using normal input samples by dropping chosen target neurons during the inference process. In this article, we focus on the detection of backdoored input samples. Triggerless backdoor attacks use normal input samples and thus are outside the scope of this work. Figure 1(c) shows a stamped image generated by Badnets [33] from the normal example shown in Figure 1(a).

2.3.3 Mislabeled Samples. In semi-supervised and supervised learning, a large number of labeled samples are required for training, validation, and testing. The samples are often labeled manually, which may be erroneous labels [74, 80, 91]. A *label error* occurs when the assigned label c'_x differs from its ground-truth $\bar{c}_x$, and in fact, oftentimes, the ground-truth $\bar{c}_x$ is the same to the predicted label c_x (i.e., $f(x) = c_x = \bar{c}_x \neq c'_x$). The presence of label errors makes some samples appear to be prediction errors instead of anomalies, as prediction error $f(x) = c_x \neq \bar{c}_x = c'_x$ is the same as label error $f(x) = c_x = \bar{c}_x \neq c'_x$ when the ground-truth $\bar{c}_x$ is unknown, namely only the predicated label c_x and the assigned label c'_x are available. Therefore, finding mislabeled samples can improve both training and evaluation of DNN models.

Label errors are pervasive in open source datasets including three widely used image datasets, MNIST, CIFAR10, and ImageNet [80], and pose serious concerns [21, 46, 81]. Figure 1(d) shows a mislabeled image from the MNIST dataset, where the assigned label is 0 and we can observe that the ground-truth should be 6. More mislabeled examples in 10 open source datasets are available at https://labelerrors.com.

Discussion. Although adversarial examples, backdoored samples, and mislabeled samples introduced previously all belong to abnormal samples, they have unique characteristics. Adversarial samples are generated by adding slight noises to normal ones, and the noises are constantly generated using the knowledge gained during the inference process to fool DNN models. An important feature of adversarial noises is human imperceptibility to ensure that adversarial examples do not deceive humans. Backdoored samples are typically generated by adding a trigger to a portion of the training samples so that the tuned DNN model is implanted with a backdoor. During the inference process, all samples with the same trigger are expected to be classified into the same target label, which requires the trigger to be effective enough to activate the backdoor and deceive the classifier. Mislabeled samples are the samples that have been incorrectly labeled in the dataset due to human error during the labeling process or inherent difficulties in accurately assigning labels to certain samples. But mislabeled samples do not contain artificially added noises and are drawn from the same underlying distribution as the correctly labeled samples. These intrinsic differences result in distinct robustness differences from normal samples based on which we propose dedicated detection methods by leveraging (adversarial) robustness (for details, refer to Section 4).

We remark that there also exist incorrectly predicted samples and **Out-of-Distribution (OOD)** samples. A sample is *incorrectly predicted* if the predicted label c_x is *not* correct (i.e., $f(x) = c_x \neq \bar{c}_x$), which is called a *prediction error*. When label errors are not considered, the assigned label c'_x is used as the ground-truth $\bar{c}_x$, thus it is similar to mislabeled samples where the predicted label differs from the assigned one. However, the label difference between predicated and assigned labeled in prediction error here is due to insufficient accuracy of the model f rather than label errors. OOD samples refer to the inputs that are OOD of the training data. Compared with the normal samples, OOD samples often have a semantic shift or covariate shift [118], namely they are drawn from different classes or a different domain—for example, the model is trained to classify different cats, but dog images are used for validation and/or testing. In such a case, DNN models cannot make a safe decision. This problem has attracted a lot of attention as well [119]. Previous work has also pointed out that OOD and incorrectly predicted samples have similar traits to adversarial examples and the related detection methods can be reused [55, 104]. We do not consider those abnormal samples in this work, as such samples are neither intentionally crafted by the adversary nor unintentionally mislabeled—indeed, their assigned labels are correct, and they should be correctly classified when the model is properly trained. Thus, in this work, abnormal examples only refer to adversarial, backdoored, and mislabeled samples but will distinguish label errors from prediction errors when detecting mislabeled samples.

2.4 Robustness

A DNN model $f : X \to C$ is *untargeted robust* with respect to an input $x \in X$ and an L_p norm distance threshold ϵ if for every sample $\hat{x}$ such that $\|x - \hat{x}\|_p \leq \epsilon$, the DNN model f on $\hat{x}$ produces the same label as the original one x, namely $f(x) = f(\hat{x})$. A DNN model $f : X \to C$ is *targeted robust* with respect to a target label t (denoted by t-robust) if for every sample $\hat{x}$ such that $\|x - \hat{x}\|_p \leq \epsilon$, the DNN model f never misclassifies the sample $\hat{x}$ to the label t (i.e., $f(\hat{x}) \neq t$).

Given two inputs $x, x' \in X$, we say x is more untargeted robust than x' if there exist two L_p norm distance thresholds ϵ, ϵ' such that $\epsilon > \epsilon'$, f is untargeted robust with respect to x and ϵ and f is *not* untargeted robust with respect to x' and ϵ'. Similarly, given a target label t, we say x is more t-robust than x' if there exist two L_p norm distance thresholds ϵ, ϵ' such that $\epsilon > \epsilon'$, f is t-robust with respect to x and ϵ and f is *not* t-robust with respect to x' and ϵ'.

Due to the lack of robustness, various approaches have been proposed to rigorously verify robustness against adversarial examples [31, 47, 107]. However, due to high computational complexity, current robustness verification approaches suffer from the scalability problem and hence fail to work for large models in practice. For instance, sound and complete SMT-based verification tools often require several hours to solve a property for ACAS Xu DNN models (a six-layer fully connected neural network with 300 neurons per layer) [25, 47], and α-β-CROWN [108], the winner of VNN-COMP 2022 [77], on average requires more than 100 seconds to verify a robustness property for a four-layer convolutional neural network with 100,000 neurons and CIFAR10 images.

To address this problem, a few approaches were proposed to certify robustness with certain probability [7, 89, 112]. In this work, we use the CLEVER (Cross-Lipschitz Extreme Value for nEtwork Robustness) score to compare the robustness of abnormal and normal examples.

The CLEVER score [112] approximates a minimum perturbation needed for a successful attack of an input x by utilizing extreme value theory. It reduces the robustness evaluation problem into a local Lipschitz constant estimation problem [27], where the local Lipschitz constant measures the smoothness and stability of the underlying function of a neural network. By calculating the Lipschitz constant, the CLEVER score provides a robustness estimation of the neural network. Weng et al. [112] proposed to use the extreme value theory to efficiently and reliably estimate the local Lipschitz constant. CLEVER score is an attack-independent robustness indicator for large-scale neural networks and has been evaluated with multiple experiments, demonstrating that it is consistent with other robustness indicators, such as attack-induced distortions [16]. More specifically, to evaluate untargeted robustness of an input x, the untargeted CLEVER score approximates a minimum perturbation Δx such that $f(x + \Delta x) \neq \bar{c}_x$, where $\bar{c}_x$ is the ground-truth of x. Thus, an input x is more untargeted robust than another input x' if the untargeted CLEVER score of x is larger than the untargeted CLEVER score of x'. Similarly, to evaluate targeted robustness of an input x with respect to a target label t, the targeted CLEVER score approximates a minimum perturbation Δx such that $f(x + \Delta x) = t$. Thus, an input x is more t-robust than another input x' if the targeted CLEVER score of x is larger than the targeted CLEVER score of x'. In summary, a higher CLEVER score indicates that the DNN model is more robust against perturbations. Readers can refer to the work of Weng et al. [112] for details.

2.5 Threat Model and Addressed Problem

We focus on the detection of adversarial, backdoor, and mislabeled samples. For adversarial samples, we assume that the adversary completely knows all the information of the model under attack, namely white-box attacks. In Section 5, we assume the adversary is unaware of the presence of the detector. In Section 6, we suppose that the adversary knows the details of the detector and try to find a way to bypass it. This assumption makes defense more difficult.

For backdoored samples, we assume that we already have a suspicious model that has been injected with a backdoor, as there is no difference between backdoored and normal samples under the clean model. Our detection does not need any backdoored samples, and detection can only be based on normal samples and the inputs, the same as in prior works [30, 103]. Our goal is to detect if an input image contains a backdoor trigger or not.

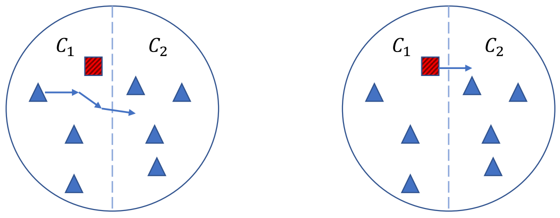
For mislabeled samples, we assume that they are a small fraction of the dataset so that a proper model can be trained. This assumption is feasible according to the number of mislabeled examples in 10 open source datasets reported at https://labelerrors.com.

We assume the defender has access to benign examples, and is unaware of how the adversary crafts adversarial and backdoored examples. Note that our approach may require some adversarial examples crafted by running adversarial attacks that could be different from the ones used by the adversary. Thus, we assume the defender can use various adversarial attacks. These assumptions are reasonable in practice, thanks to many publicly available datasets and adversarial attack tools.

The addressed problem is this: *given an input example x to a DNN model, how to effectively decide whether x is normal or abnormal?* Our solution is to use robustness, which could be efficiently and effectively measured via attack costs.

3 ABNORMAL EXAMPLE CHARACTERIZATION

In this section, we theoretically analyze and empirically validate differences in robustness and attack cost between abnormal and normal samples.



(a) Normal example (b) Adversarial example

Fig. 2. An illustration of robustness of normal and adversarial examples. (a) The distance to decision boundary of a normal example (blue triangle). (b) The distance to decision boundary of an adversarial example (red square). c_1 and c_2 refer to different classes, and the arrows simulate the step-by-step movement of the sample toward the decision boundary under an adversarial attack.

3.1 Robustness: Normal vs. Abnormal

Normal samples in the training dataset go through a number of training epochs during which common features are extracted from the majority of the training samples and learned by updating the network weights. Consequently, well-trained networks have good generalization ability and normal samples from the same distribution of training samples are far away from the decision boundary, thus they are robust against small perturbations, as illustrated in Figure 2(a).

However, abnormal samples present different robustness from normal samples. Let us first consider adversarial examples. As illustrated in Figure 2(b), adversarial examples are less robust against small perturbations for the following reasons. Adversarial attack crafts an adversarial example $\hat{x} = x + \Delta x$ by adding a perturbation Δx on a benign example x such that $f(x) \neq f(\hat{x})$. The perturbation Δx is generally required to be imperceptible, which makes the adversarial example very close to the decision boundary [38, 41, 106], and thus adversarial examples are often less robust than normal samples w.r.t. their predicted labels. In contrast to adversarial samples, backdoored samples need a robust trigger to ensure their validity, and the conflict of the original image features and label also leads to a robustness difference from the normal samples. Mislabeled samples are introduced due to either mistakes made by workers or poor-quality images, but they are a small fraction of the dataset, thus disagreeing with predictions based on the majority of the training samples. Indeed, mislabeled samples are likely to be followed by prediction errors—that is, they are classified to ground-truths instead of assigned labels. Mislabeled samples are naturally close to the decision boundary of their assigned labels and/or far away from the decision boundary of their ground-truths. In either case, mislabeled samples are likely less robust w.r.t. their assigned labels but more robust w.r.t. their ground-truths. In summary, abnormal and normal samples differ in the robustness.

To validate the robustness difference between normal and abnormal examples, we conduct a robustness comparison in terms of the CLEVER score. Recall that the CLEVER score is a robust distance of perturbations—the larger the CLEVER score, the more robust. We train two MNIST models for the analysis of robustness using the model architecture from Feinman et al. [28]. The first model is a clean model whose accuracy on the test dataset is 99.18%. The second model is a model injected with a backdoor via BadNets [33] whose clean accuracy on the test dataset is 99.05%. We compare the CLEVER scores of normal samples, adversarial samples, and mislabeled samples on the first model, and compare the CLEVER scores of normal samples and backdoored samples on the second model. We sequentially select the first 100 correctly classified images from the MNIST test dataset as the experimental subjects, then generate a series of abnormal samples based on these 100 correctly classified samples. Adversarial examples are generated by applying

Table 1. CLEVER Scores and p-Values of Normal and Adversarial Samples with a Confidence Interval at the 90% Significance Level

Target	Normal	Adversarial Samples				p-Value
Label	Samples	FGSM	BIM	JSMA	C&W	
Untarget	0.0069 ± 0.0003	0.0031 ± 0.0004	0.0022 ± 0.0003	0.0003 ± 0.0001	0.0001 ± 0.0001	5.66×10^{-94}
Target-2	0.0113 ± 0.0005	0.0040 ± 0.0006	0.0024 ± 0.0004	0.0003 ± 0.0001	0.0001 ± 0.0001	1.49×10^{-121}
Target-5	0.0126 ± 0.0067	0.0067 ± 0.0009	0.0070 ± 0.0008	0.0052 ± 0.0004	0.0082 ± 0.0026	1.08×10^{-6}
LLC	0.0206 ± 0.0077	0.0110 ± 0.0026	0.0130 ± 0.0028	0.0113 ± 0.0065	0.0115 ± 0.0041	0.0050

four representative white-box attacks, FGSM, BIM, JSMA, and C&W, using the implementations from a widely used AI security library Foolbox [87] with default parameters, where in default, FGSM, BIM, and JSMA are untargeted adversarial attacks, and JSMA is a targeted adversarial attack with automatically chosen target label. Backdoored samples are generated by Badnets [33]. The success rate for both the adversarial and backdoor attacks is close to 100%. The mislabeled samples are selected from the work of Northcutt et al. [80].

We also perform a statistical analysis to substantiate the robustness differences between the normal and abnormal samples. As recommended by Arcuri and Briand [4], we use the independent-samples t-test [8], a statistical test that is used to compare the means of two groups. It is widely used in hypothesis testing to determine whether two groups are different from one another. In this work, we use the t-test to determine whether there is a significant statistical difference between the normal and abnormal samples in terms of CLEVER scores. Let A and B be the sets of normal samples and abnormal samples, respectively. We denote by C_A and C_B the sets of CLEVER scores of the sets A and B, respectively. The null hypothesis (H_0) and the alternative hypothesis (H_1) are defined as follows:

$$H_0 : \overline{C_A} = \overline{C_B}; \qquad H_1 : \overline{C_A} \neq \overline{C_B},$$

where $\overline{C_A}$ and $\overline{C_B}$ denote the means of C_A and C_B, respectively. Intuitively, if the null hypothesis H_0 is rejected (i.e., the alternative hypothesis H_1 is accepted), then there is a significant difference between the two sets of CLEVER scores, indicating that there is a significant statistical difference between the normal and abnormal samples. Otherwise, the null hypothesis H_0 is accepted (i.e., the alternative hypothesis H_1 is rejected), and there is no significant statistical difference between the normal and abnormal samples. We use the p-value to quantify the probability of rejecting the null hypothesis, where the smaller the p value, the more likely to reject the null hypothesis. If the p-value is less than a given confidence level α (e.g., 0.01 or 0.05), the null hypothesis H_0 is rejected and otherwise accepted.

The CLEVER scores and p-values of normal and adversarial samples are reported in Table 1. The column *Target Label* shows how the CLEVER score is calculated, where *Untarget* indicates the untarget CLEVER score, *Target-n* denotes the target CLEVER score w.r.t. the rank-n label, and *LLC* denotes the target CLEVER score w.r.t. the least likely label. The column *Normal Samples* shows the CLEVER scores of benign samples, and the other columns show the CLEVER scores of adversarial samples. Remark that the ground-truth $\bar{c}_x$ used for computing the untargeted CLEVER score of an input sample x is the label $f(x)$ to which the input sample x is classified following an adversarial attack. We can observe that the CLEVER scores of adversarial examples are obviously smaller than that of normal ones, which confirms our previous speculation that there is a significant difference in robustness between adversarial examples and normal samples. In particular, the ratios of the untargeted and target-2 CLEVER scores between normal and adversarial examples are larger than

Table 2. CLEVER Scores and p-Values of Normal and Backdoored Samples with a Confidence Interval at the 90% Significance Level

Target Label	CLEVER Score		p-Value
	Normal Samples	Backdoored Samples	
Untarget	0.0098 ± 0.0004	0.0200 ± 0.0017	1.07×10^{-41}
Target-2	0.0173 ± 0.0010	0.0432 ± 0.0097	2.13×10^{-31}
Target-5	0.0181 ± 0.0011	0.0483 ± 0.0084	1.01×10^{-29}
LLC	0.0187 ± 0.0007	0.0489 ± 0.0097	1.58×10^{-36}

Table 3. CLEVER Scores and p-Values of Normal and Mislabeled Samples with a Confidence Interval at the 90% Significance Level

Target Label	Normal Examples	Mislabeled Samples			
		Assigned Label c'_x		Predicted Label c_x	
		CLEVER Score	p-Value	CLEVER Score	p-Value
Untarget	0.0069 ± 0.0003	0.0005 ± 0.0009	3.21×10^{-24}	0.0070 ± 0.0012	0.008
Target-2	0.0113 ± 0.0005	0.0006 ± 0.0011	1.04×10^{-23}	0.0091 ± 0.0017	0.029
Target-5	0.0126 ± 0.0009	0.0128 ± 0.0046	0.64	0.0110 ± 0.0027	0.915
LLC	0.0206 ± 0.0023	0.0317 ± 0.0265	0.49	0.0279 ± 0.0219	0.111

others. Conducting robustness analysis on these two types makes it easier to distinguish between normal and adversarial examples.

The CLEVER scores and p-values of normal and backdoored samples are reported in Table 2, where the column *Backdoored Samples* shows the CLEVER scores of the backdoored samples. Different from adversarial examples, the CLEVER scores of backdoored samples are significantly larger than that of normal samples. This is because a backdoor Trojan has to be trained very robustly to take effect when it is added on different images. Consequently, breaking the backdoor trigger via adversarial attack is more difficult than attacking on normal examples.

The CLEVER scores and p-values of normal and mislabeled samples are reported in Table 3. The column *Assigned Label* shows the CLEVER scores of the mislabeled samples where the assigned label c'_x of each mislabeled sample x in the dataset is regarded as the ground-truth $\bar{c}_x$ when computing the untargeted CLEVER scores. The column *Predicted Label* shows the CLEVER scores of the mislabeled samples where the predicted label $f(x) = c_x$ for each mislabeled sample x is regarded as the ground-truth $\bar{c}_x$ when computing the untargeted CLEVER scores. We note that if the assigned label c'_x is the same as the target label (e.g., target-2, target-5, and LLC), then the targeted CLEVER score is 0, which does not make sense. Thus, we filter out the assigned label c'_x when ranking the labels, and otherwise the targeted CLEVER scores in the columns *Assigned Label* and *Predicted Label* are the same. We can observe that the untargeted CLEVER scores with assigned label c'_x as ground-truth $\bar{c}_x$ (i.e., $c'_x = \bar{c}_x$) are much smaller than that with predicted label c_x as ground-truth $\bar{c}_x$ (i.e., $c_x = \bar{c}_x$). Furthermore, the untargeted CLEVER scores of mislabeled samples with predicted label c_x as ground-truth $\bar{c}_x$ and the targeted CLEVER scores of mislabeled samples are similar to that of normal samples, indicating that these predicted labels of mislabeled samples are more likely their real ground-truths.

Summary. The preceding results indicate that there is a huge difference in untargeted and/or targeted robustness in terms of the CLEVER score between abnormal and normal examples. In particular, (1) the CLEVER scores of adversarial examples are significantly smaller than that of normal ones; (2) the CLEVER scores of backdoored examples are much larger than the CLEVER scores of normal ones; and (3) the untargeted CLEVER scores of mislabeled examples with assigned label c'_x as ground-truth $\bar{c}_x$ are much smaller than that of mislabeled examples with predicted label c_x as ground-truth $\bar{c}_x$ and smaller than that of normal examples with assigned label c'_x as ground-truth $\bar{c}_x$, whereas the CLEVER scores of mislabeled examples with predicted label c_x as ground-truth $\bar{c}_x$ are similar to that of normal ones. In addition, the difference in the CLEVER score is more significant w.r.t. the untargeted and target-2. These observations will be leveraged to detect abnormal examples.

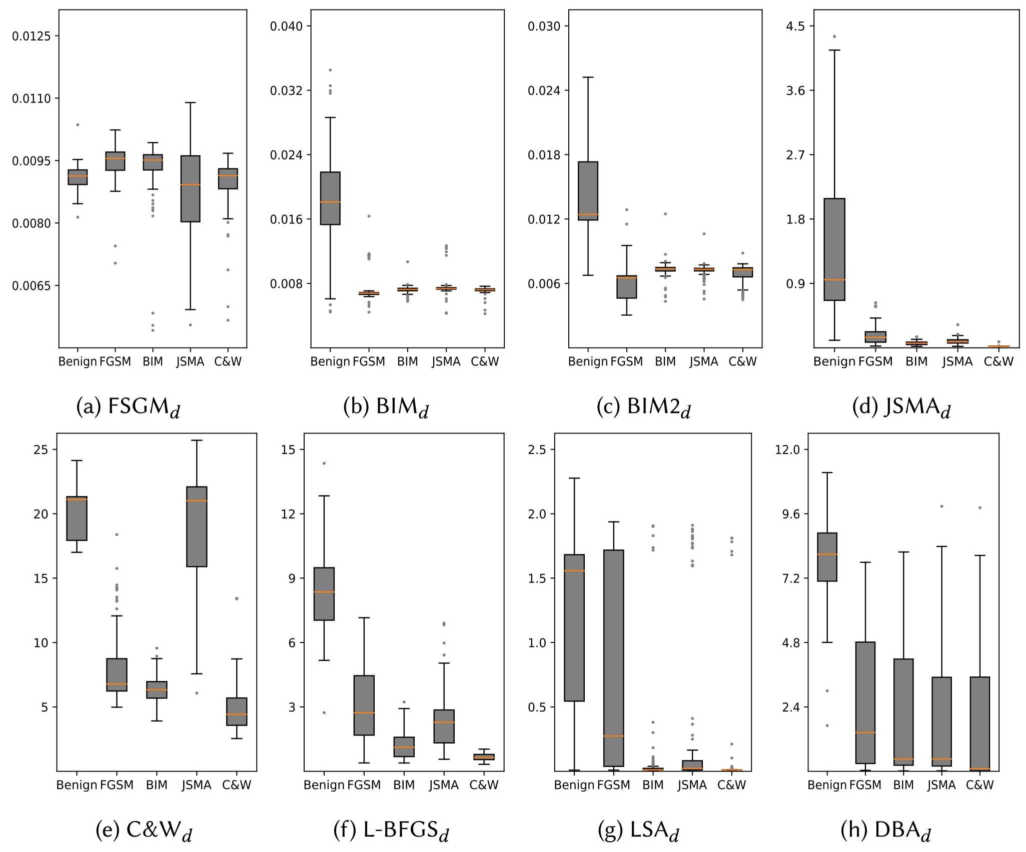
3.2 Attack Cost: Normal vs. Abnormal

We have revealed the difference in robustness between normal and abnormal samples, but existing robustness certification techniques are too time costly to be used as real-time detection. For instance, on a single GTX 1080 GPU, the cost of computing the untargeted CLEVER score is nearly 450 seconds for an MNIST image and 1,150 seconds for a CIFAR10 image.

To effectively and efficiently detect abnormal samples, we propose A^{2}D, a novel detection method that uses the *attack cost* required to turn the input sample into an adversarial sample to estimate its robustness so that we can exploit various off-the-shelf adversarial attack methods and tools. Intuitively, the robustness indicator measures how large a perturbation is required for the adversarial attack to succeed, and iterative adversarial attacks such as BIM look for adversarial examples by iteratively increasing the perturbation, making the attack cost proportional to the robustness and can significantly improve the efficiency of robustness evaluation. Although there is a gap between the attack cost and robustness, this estimate is reliable since these adversarial attack methods exploit the first-order gradient as much as possible [71, 112].

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, the FGSM adversarial attack method is not suitable since it simply performs one-step perturbation that will result in almost the same attack costs on normal and abnormal examples. In contrast, iterative adversarial attack methods (e.g., BIM, JSMA, and C&W) which 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. To justify this observation, we craft adversarial examples from the first 1,000 correctly predicated images of the MNIST test dataset by applying four adversarial attack methods, FGSM, BIM, JSMA, and C&W, respectively, the same as in Section 3.1. Then, the attack costs of these adversarial examples are measured by attacking them using eight widely used adversarial attack methods, FGSM, BIM, BIM2 (i.e., BIM under L_2 norm), JSMA, C&W, L-BFGS, LSA, and DBA, respectively. For ease of reading, an adversarial attack method used as detection is marked by a subscript d (e.g., BIM$_d$). We still use their implementations in Foolbox [87] with default parameters, but we adopt *early stop* for BIM$_d$, BIM2$_d$, and DBA$_d$ (i.e., stop iterating immediately after finding an adversarial example) instead of always iterating a fixed number of steps, and otherwise the attack costs are almost the same on normal and abnormal examples. According to the results of Table 1 and the fact that untargeted adversarial attacks are more efficient than targeted counterparts, we will use untargeted versions of adversarial attacks as detection, unless the adversarial attack (i.e., JSMA$_d$) only supports targeted attack for which we use target-2 as the target label.

The adversarial attack costs on benign and adversarial examples in terms of time are depicted in Figure 3, where the x-axis is the type of input examples and the y-axis is the time cost (in seconds) of adversarial attacks used as detection. It is not surprising that the attack time of adversarial and normal examples using FGMS$_d$ is similar, as it is a one-step attack. Although the results show the



(a) FSGM$_d$ (b) BIM$_d$ (c) BIM2$_d$ (d) JSMA$_d$

(e) C&W$_d$ (f) L-BFGS$_d$ (g) LSA$_d$ (h) DBA$_d$

Fig. 3. Attack time of benign and adversarial examples, where the y-axis means seconds.

variation between different attack methods and detections, the differences are often significant when an iterative attack is used as detection—for example, BIM$_d$, BIM2$_d$, JSMA$_d$, L-BFGS$_d$, and DBA$_d$, whereas L-BFGS$_d$ is less efficient than the others. We find that the attack time differences are not stable when C&W$_d$ and LSA$_d$ are used. This is because C&W$_d$ implements a binary search to minimize distortion and may stop searching when there is a bottleneck, whereas LSA$_d$ suffers from a low ASR, which causes it to have many outlier attack times.

Considering the attack efficiency and diversity, the adversarial attack methods BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$ will be used as detection in the follow-up experiments. These adversarial attack methods cover both white-box and black-box attacks, as well as different L_p distance metrics (L_0, L_2, and L_∞). We remark that to choose an attack as detection, we need to consider if it has a difference in the number of attack iterations. For example, we cannot use FGSM, as it modifies the input only once, which will result in the same attack cost for all samples. In our preliminary conference version [125], we studied the cost distribution for more attacks and discussed how to choose attacks for detection (please refer to our previous work [125] for details). To support this claim, we perform a statistical analysis using the independent-samples t-test. We calculate the p-value for each of four selected representative attacks (BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$), where $\overline{C_A}$ now is the means of the time cost of all the normal samples and $\overline{C_B}$ is the means of the time cost of all the adversarial examples. The resulting p-values are 2.06×10^{-31}, 3.54×10^{-14}, 2.51×10^{-38}, and 5.84×10^{-68}, respectively, all of which are significantly smaller than 0.01. In contrast, the p-value

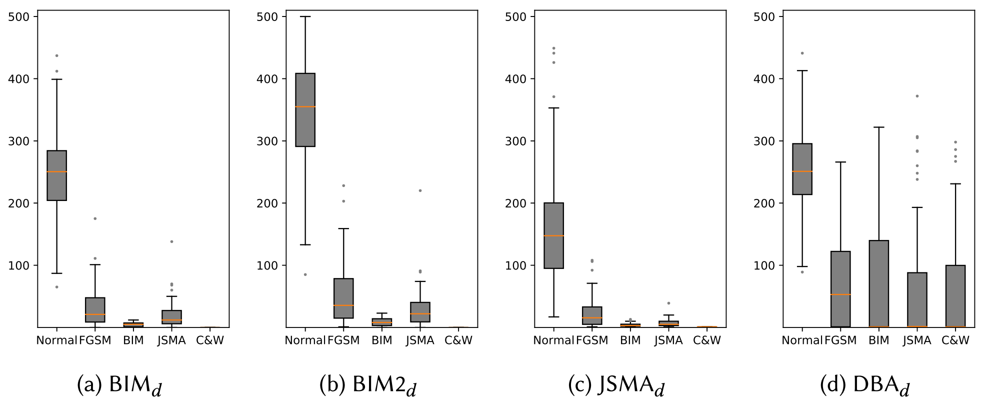


Fig. 4. Attack iterations comparison on MNIST, where the x-axis gives the attack tools but *normal* denotes normal examples, and the y-axis denotes the attack cost.

of FGSM2$_d$ is 0.82, namely FGSM2$_d$ is not suitable for approximately measuring robustness, which is consistent in our theoretical analysis.

We also analyze the numbers of iterations of the adversarial attack methods BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$. The results are depicted in Figure 4, where the x-axis is the type of input examples and the y-axis is the number of iterations of adversarial attacks used as detection. Remark that we did not tune parameters; these widely used default 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. To further support this claim, we also perform a statistical analysis using the independent-samples t-test. We calculate the p-value for each of BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$, where $\overline{C_A}$ now is the means of the numbers of iterations of all the normal samples and $\overline{C_B}$ is the means of the numbers of iterations of all the adversarial examples. The resulting p-values are 4.45×10^{-35}, 3.26×10^{-25}, 1.39×10^{-07}, and 3.31×10^{-17}, respectively, all of which are significantly smaller than 0.01.

Since the number of iterations does not depend on computing devices and running environment, we will use the number of iterations as the indicator of attack costs in the follow-up experiments. To demonstrate the effectiveness of the attack iterations for characterizing abnormal examples, we randomly choose 1,000 samples for each type of image (i.e., Normal, FGSM, BIM, DeepFool, and C&W), and divide each type of images into two independent sets, marked by subscripts 1 and 2. Then we use BIM$_d$ to attack these images and record the number of iterations required. To show the difference in attack iterations, we calculate the average Euclidean distance of the number of iterations between each pair of sets of examples, and the results are presented in Figure 5. We can see that for different types of examples (adversarial vs. benign), the distance is enormous. However, 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 adversarial attack methods, the distance is also quite similar, meaning that even if the adversarial examples are generated by different adversarial attack methods, they are also "cognate" examples and have similar attack iterations.

We confirm that the differences in attack costs (either in time or iterations) are consistent with the CLEVER scores, and for various types of examples (i.e., adversarial vs. normal), the differences in attack costs are enormous. We do not show the attack costs of backdoored and mislabeled samples, but the results in robustness illustrate that they have a similar property as adversarial

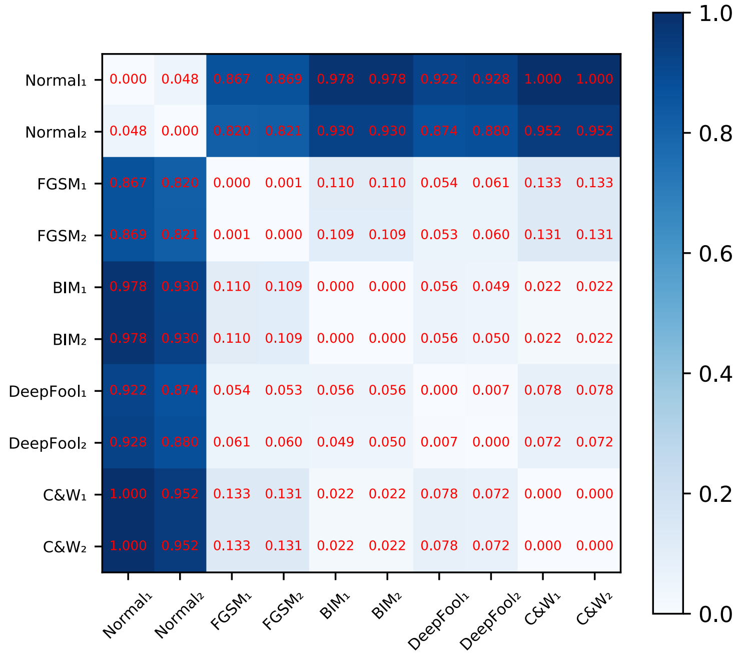


Fig. 5. Euclidean distances of the average number of iterations between each pair of sets of examples.

examples. Indeed, the correlation between attack costs and CLEVER scores has been reported in our previous work [125].

Discussion. In this work, we use attack iterations for detecting abnormal samples. Attack time, maximum output logit, loss, or hidden neuron changes after adding random noises to the original image are also considered to be related to robustness [100]. We decided to use attack iterations because not only are they intuitively connected with robustness and the output result is intuitive but also other indicators have some intolerable drawbacks. Specifically, attack time and maximum output logit are not stable enough to be used as indicators, where the former is susceptible to hardware device and running environment, and the latter may generate vastly different results on inputs with similar robustness [13, 55]—for example, under distillation [85], where output logits of an input sample may be scaled up 100 times, whereas the robustness does not increase. The change in hidden neurons or network output after adding random noises is not straightforward, as their dimensionality is too high, so another DNN or other complex classifier is required for further analysis. Furthermore, maximum output logit, loss, and hidden neuron states could be easily controlled by the adversary so that adaptive attacks could be designed to bypass them [16, 101]. In contrast, thanks to the diversity of adversarial attack methods, an ensemble detection method can be easily constructed using attack iterations of different attack methods, where different attack methods have the ability to capture different features and attack iterations cannot be easily incorporated into abnormal example generation, making ensemble detection more reliable and difficult to bypass. In general, we suggest that other indicators could be used to further improve the resilience of our approach.

4 ABNORMAL EXAMPLE DETECTION

In this section, we consider how to detect abnormal examples by leveraging attack costs (i.e., attack iterations). We propose two detection approaches for adversarial examples, and one detection approach for backdoored and mislabeled samples, respectively, despite that other detection approaches are possible using our findings.

Hereafter, we sometimes denote by attack$_d$ the adversarial attack method that is used as detection (i.e., to generate attack iterations).

4.1 Adversarial Example Detection

Based on the results in Table 1, namely the untargeted and target-2 targeted CLEVER scores of adversarial examples are significantly smaller than those of normal ones, we propose two adversarial example detection approaches based on ***k*-Nearest Neighbors (K-NN)** and standard score (Z-score), respectively. The former requires both benign and adversarial examples, whereas the latter requires only benign examples.

4.1.1 K-NN-Based Detection Approach. The K-NN-based detection approach requires both benign and adversarial examples to train the detector. The adversarial examples used to train are generated by adversarial attack methods that could be different from the attack used by the adversary. Assume that we have two disjoint sets: B the set of benign examples and A the set of adversarial examples.

Single Detector. Fix an attack$_d$ o and let α_y denote the cost of attacking y using the attack$_d$ o. Then, a set of attack costs $\{\alpha_y \mid y \in A \cup B\}$ can be collected by utilizing the attack$_d$ o. For each unknown input x and parameter K, we first compute the attack cost α_x and then identify KNN $N_K = \{\alpha_{y_i} \mid 1 \leq i \leq K\}$ 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 the K-neighborhood of the input x.

Ensemble Detector. Similarly, we can build a K-NN-based ensemble detector using multiply attacks$_d$ $o_1, \dots, o_n$, for which a vector of attack costs $\vec{\alpha}_y = (\alpha_y^1, \dots, \alpha_y^n)$ is used instead of a single attack cost, where α_y^j for $1 \leq j \leq n$ is the attack cost of the example y by utilizing the attack$_d$ o_j. Consequently, a set of vectors of attack costs $\{\vec{\alpha}_y \mid y \in A \cup B\}$ can be collected. For each unknown input x and parameter K, we identify KNN $N_K = \{\vec{\alpha}_{y_i} \mid 1 \leq i \leq K\}$ of $\vec{\alpha}_x$ 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 $|A_x| > |B_x|$.

4.1.2 Z-Score-Based Detection Approach. The second detection approach leverages the Z-score, a widely used statistical technique for measuring distance between a data point and the mean using standard deviations [53]. More specifically, the Z-score of a sample i is $z = \frac{i-\mu}{\sigma}$, where μ is the sample mean and σ is the sample standard deviation. Intuitively, the score z indicates how many standard deviations the sample i is far away from the sample mean. The Z-score-based detection approach uses the distribution of attack costs of benign examples to check whether an example is adversarial or not.

Single Detector. Fix a set B of benign examples and an attack$_d$ o. We first compute the distribution of attack costs of the examples in B. Assume the distribution is an approximately normal distribution $N(\mu, \sigma^2)$, and otherwise we can transform it by applying the Box-Cox power transformation [9]. Then, the Z-score z_y of an example y is $z_y = \frac{\alpha_y - \mu}{\sigma}$. 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. To generalize this approach from one attack$_d$ to multiply attacks$_d$ $o_1, \ldots, o_n$, we build a Z-score-based detector d_j for each attack$_d$ o_j, resulting in n detectors $d_1, \ldots, d_n$. The ensemble detector determines whether an input is adversarial or not by taking into account the results of all the detectors $d_1, \ldots, d_n$. Considering $k \leq n$, the ensemble detector classifies an input to adversarial if at least k detectors classify the input to adversarial and otherwise benign. The ensemble detector would have high **True-Positive Rates (TPRs)** when $k = 1$ and high true-negative rates when $k = n$.

We remark that both K-NN- and Z-score-based detection approaches require attack iterations. The same as in Section 3.2, to collect attack iterations, we will use the untargeted versions of the adversarial attack methods BIM$_d$, BIM2$_d$, and DBA$_d$ and the targeted adversarial attack JSMA$_d$ with target-2 as the target label in the follow-up experiments.

4.2 Backdoored Sample Detection

Basic Idea. According to the results in Table 2, the CLEVER scores of backdoored samples are much larger than the CLEVER scores of normal ones. Thus, a straightforward solution is to use the opposite of the principle used in the adversarial example detection. Namely, the attack cost α_x of an input x can still be used as a detection indicator and a set of attack costs $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ can be derived by mounting an attack$_d$ o, where the larger α in this set of attack costs, the higher the probability that it will be considered a backdoored sample. However, this approach has a significant drawback in terms of detection efficiency. In the detection of adversarial examples, adversarial examples are significantly less robust than benign samples, and the time cost required to attack these abnormal inputs is negligible. However, in the detection of backdoored samples, it will requires many more attack iterations to carry out a successful attack.

For detection efficiency consideration, we propose to replace the cost of a successful attack with feature values after a fixed number of attack iterations, where feature values are the output values of the neurons in the hidden layers and represent the learned features of DNNs from samples. Intuitively, the backdoor trigger in backdoored samples is quite robust, namely their prediction results are more difficult to change than normal samples by adversarial attacks. Feature values as intermediate computation results determine the final prediction results, thus the feature values of the backdoored samples that represent the backdoor trigger should be more significant than those of normal samples after a fixed number of attack iterations, indicating that more attack iterations are required to change the prediction results of backdoored samples. Indeed, the more significant the feature values, the greater the contribution of the feature values to prediction results. This is in line with the idea of backdoor attacks, which is to freeze the prediction results through a specific subtle Trojan. Inspired by robustness analysis on feature values [113–115] and the fact that backdoor triggers injected in backdoor attacks are generally small, we use the maximum value of the outputs of a chosen hidden layer (e.g., convolutional layer) as the feature value of a backdoor trigger. More specifically, the detection metric is defined as follows:

$$M = \max_{i=1,\ldots,n} h_i^{(k)},$$

where n denotes the number of outputs of the chosen hidden layer and $h_i^{(k)}$ denotes the value of the i-th output after k steps of attack. The number of attack iterations can be fixed to a chosen constant k, thus addressing the efficiency issue of backdoored sample detection.

Detector. As the detection metric used in this case is a single value, we can design a Z-score-based detector using a threshold so that the detector does not reply upon any backdoored samples. Fix a set B of N normal samples, an attack$_d$ o, and an acceptable **False-Positive Rate (FPR)** r for normal samples. We first conduct an untargeted or targeted adversarial attack with k iterations ($k = 2$ in our experiments) on all the normal samples in B to get their maximal feature values $\{M_1, \dots, M_N\}$. We sort the set of feature values $\{M_1, \dots, M_N\}$ and select the $r \times N$-th largest one as the threshold M_t. Based on the threshold value M_t, an unknown input x is regarded as a backdoored sample if $M > M_t$, where M denotes the maximal feature value of the unknown input x.

4.3 Mislabeled Sample Detection

Basic Idea. We consider two scenarios based on predicted and assigned labels, namely whether the input sample x is classified to the assigned label c'_x or not. We design the detector according to the results in Table 3, namely the untargeted CLEVER scores of mislabeled examples with assigned label c'_x as ground-truth $\bar{c}_x$ are much smaller than that of mislabeled examples with predicted label c_x as ground-truth $\bar{c}_x$ and smaller than that of normal examples with assigned label c'_x as ground-truth $\bar{c}_x$, whereas the untargeted and targeted CLEVER scores of mislabeled examples with predicted label c_x as ground-truth $\bar{c}_x$ are similar to those of normal ones.

The first scenario is that the model 'correctly' classifies a sample x—that is, $f(x) = c'_x$, where c'_x is the assigned label of x. The 'correctly' only means that the sample x is classified to the assigned label c'_x, but the sample x may not belong to it due to mislabeling and prediction error. In this scenario, we can deduce that either x is correctly labeled in the dataset and there is no prediction error, or x is mislabeled in the dataset and there is a prediction error, and otherwise $f(x) \neq c'_x$. For this kind of sample, if it is mislabeled, it is less untargeted robust and easier to attack than normal examples via untargeted adversarial attacks with the assigned label c'_x as the ground-truth $\bar{c}_x$ (cf. the column *Assigned Label* c'_x in Table 3). The detection method in this scenario is similar to the detection of adversarial examples, with the difference that our inputs are samples from the dataset. We use the attack cost α_{x,c'_x} of untargeted adversarial attacks with the assigned label c'_x as the ground-truth $\bar{c}_x$ to detect this kind of mislabeled sample, where the smaller the attack cost α_{x,c'_x}, the larger possibility the sample x is mislabeled. To sort mislabeled samples with the same attack costs, we add the classification probability p_{x,c'_x} of each sample x onto its attack cost α_{x,c'_x}.

The second scenario is that the model 'incorrectly' classifies the mislabeled sample x (i.e., $f(x) \neq c'_x$). The 'incorrectly' here only means that the sample x is not classified to the assigned label c'_x. Indeed, this scenario may come from either a prediction error or a label error. If it is a prediction error, then c'_x is more likely the ground-truth of x, thus no label errors occurs; otherwise $f(x)$ is more likely the ground-truth of x and a label error occurs. Therefore, it remains to distinguish label errors from prediction errors. However, we cannot check if the input is a mislabeled sample or not in the same way as for detecting adversarial examples or detecting mislabeled samples in the first scenario because $\alpha_{x,c'_x} = 0$ when the assigned label c'_x is used as the ground-truth $\bar{c}_x$ during the untargeted adversarial attack. Instead, we leverage the observation that if the sample x belongs to the predicted label $c_x = f(x)$ (i.e., $f(x) \neq c'_x$ comes from a label error), then it would be difficult to find an adversarial example x' via untargeted adversarial attacks with the predicated label c_x as the ground-truth $\bar{c}_x$, like untargeted adversarial attacks to normal examples with the assigned label as the ground-truth (cf. the column *Predicted Label* c_x in Table 3). Thus, we use attack cost α_{x,c_x} of untargeted adversarial attacks with the predicated label c_x as the ground-truth $\bar{c}_x$ and classification probability p_{x,c'_x} to distinguish label errors from prediction errors, where the larger the attack cost α_{x,c_x} (respectively, the smaller the probability p_{x,c'_x}), the larger the possibility that the sample x is mislabeled.

Based on the observations in other works [80, 111], the preceding two scenarios are often mixed together in practice. Therefore, we design the following detection indicator taking into accounting both kinds of mislabeled samples:

$$loss(x) = \begin{cases} \alpha_{x,c'_x} + p_{x,c'_x}, & \text{if } f(x) = c'_x, \\ p_{x,c'_x} + \frac{\lambda}{\alpha_{x,c_x}}, & \text{if } f(x) \neq c'_x, \end{cases}$$

where λ is a hyperparameter to balance the probability p_{x,c'_x} and the attack cost α_{x,c_x}. In general, the lower the loss value, the more likely x is mislabeled. The loss function tends to select samples that are easy to attack when the assigned label c'_x is used as the ground-truth $\bar{c}_x$ but are difficult to attack when the predicated label c_x is used as the ground-truth $\bar{c}_x$. If α_{x,c_x} is large, the overall loss will only be added by a small number. If α_{x,c_x} is small, this means that the model is not robust to the untargeted adversarial attacks with the predicted label c_x as the ground-truth $\bar{c}_x$, and based on the idea in other works [104, 106], we tend to classify the input as prediction error and penalise the whole loss.

Detector. We design a mislabeled sample detection approach by leveraging the preceding loss indicator. We define the probability that the input $x \in X$ is mislabeled as follows:

$$P(x) = \frac{\sum_{x' \in X} G(x', x)}{n}, \text{ where } G(x_1, x_2) = \begin{cases} 1, & \text{if } loss(x_1) < loss(x_2), \\ 0, & \text{otherwise } loss(x_1) \geq loss(x_2). \end{cases}$$

The larger the probability $P(x)$, the more likely x is mislabeled. Thus, the probabilities $P(x)$ for $x \in X$ allow us to sort all the samples so that one can manually validates samples according to their probabilities. Given a probability threshold p_0, one can also obtain all the samples x such that $P(x) > p_0$. We should emphasize that our indicator for detecting mislabeled samples using attack costs is generic and can be leveraged by other detection approaches, such as the one in the work of Northcutt et al. [80].

5 EVALUATION

In this section, we evaluate our detection approach on multiple widely used datasets and models. The experiments are designed to answer the following research questions:

*RQ*1: How effective are the attack costs (i.e., attack iterations) as an indicator for adversarial example detection?
*RQ*2: How effective and efficient is A^{2}D for adversarial example detection?
*RQ*3: How effective is A^{2}D for backdoored sample detection?
*RQ*4: How effective is A^{2}D for mislabeled sample detection?

5.1 Experiment Setups

For reproductivity, the information of the target models and attack parameters used in our experiments are given in the following and the source code of our tool A^{2}D is available on GitHub [1]. To mitigate the noise on execution time, all experiments are repeated three times and average execution time is reported.

To answer the preceding research questions, in total, we compare A^{2}D with four adversarial example detection approaches denoted by BL$_1$ through BL$_4$, two backdoored sample detection approaches denoted by BL$_5$ and BL$_6$, and one mislabeled sample detection approach denoted by BL$_7$:

— BL$_1$ [28] uses a Gaussian Mixture Model to represent network outputs, which was considered to be *the most effective detection on MNIST* among 10 detections in the work of Carlini and Wagner [14].
— BL$_2$ [70] uses local intrinsic dimension to distinguish adversarial subspace and claims to be better than BL$_1$.
— BL$_3$ [106] uses a label change rate through model mutation to distinguish adversarial examples.
— BL$_4$ [104] dissects hidden neuron states to construct a fault tolerance approach.
— BL$_5$ [17] uses activation clustering to determine whether an input is a backdoored sample.
— BL$_6$ [30] detects backdoored samples by superimposing various image patterns on inputs.

Table 4. Parameters of Attacks for Env_1

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	$\kappa = 0, c = 0.02$

Table 5. 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	$\kappa = 0, c = 0.6$

— BL_7 [80, 81] detects mislabeled samples by estimating the joint distribution of latent true label and noise label, and sorts suspected samples directly according to the model output probabilities.

We emphasize that BL_3 and BL_4 published respectively at ICSE'19 and ICSE'20, and BL_7 published at NeurIPS '21, are arguably the corresponding state-of-the-art detection approaches. Other baselines are well-known algorithms and widely used for comparison in their respective fields. For the sake of fairness, the baselines we selected for comparison all assume the same or a more powerful defender than A^2D. For example, in backdoored sample detection, BL_5 assumes that the defender can get the backdoored samples in advance, whereas BL_6 does not, the same as A^2D. We do not compare other detection methods (e.g., [35]) that use less information than A^2D. It is worth mentioning that the full source code of some adversarial attacks is not available. Thus, for the missing adversarial attacks, we use alternative implementations in Foolbox [87] (see the following), as recommended by Tramèr et al. [101].

The performance of BL_1 through BL_4 may vary due to platforms, models, and attack settings. For a fair comparison, we implement our approach in their environments and conduct comparison directly using the same target models and attacks provided by each of them. In total, there are three different environments: Env_1 on the Keras platform, and Env_2 and Env_3 on the PyTorch platform:

— Env_1 contains models and attack methods provided by BL_1 [28], where the DNN model for MNIST is LeNet, for CIFAR10 is a deep 12-layer convnet, and the accuracy on training/testing dataset is 99.6%/99.1% for MNIST and 87.3%/80.3% for CIFAR10. BL_2 [70] uses the models and attack code segments provided by BL_1, so Env_1 is used by these two baselines. The attack parameters of crafting adversarial samples are listed in Table 4. It should be noted that there are two slightly different BIM implementations in Env_1 and no C&W implementation is provided in Env_1, thus we use 'bim-a' implementation for BIM and the C&W implementation from Foolbox. For evaluating detection approaches, Env_1 uses the testing datasets of MNIST and CIFAR10 that can be correctly classified by the underlying DNN models based on which adversarial examples are crafted by the considered adversarial attack methods (cf. Table 4).
— Env_2 is provided by BL_3 [106], which includes the LeNet5 model for MNIST and GoogleNet model for CIFAR10. The accuracy on training/testing dataset is 98.5%/98.3% for MNIST and 99.7%/90.5% for CIFAR10. The provided adversarial attack methods and their parameters for crafting adversarial samples are shown in Table 5. BL_3 stated that the included black-box attack (denoted by BB) is ineffective on CIFAR10, as the authors could not train a good substitute model, so we omit this black-box attack on CIFAR10. For evaluating detection approaches, Env_2 randomly selects 1,000 normal samples from the testing dataset MNIST

(respectively, CIFAR10) during each evaluation and adversarial examples have been provided therein which were crafted by the provided adversarial attack methods using randomly selected 1,000 normal samples from the testing dataset MNIST (respectively, CIFAR10).

— Env_3 is provided by BL_4 [104], which includes LeNet4 for MNIST, WRN for CIFAR10, and ResNet101 for ImageNet. The accuracy on the training/testing dataset is 90.0%/98.4% for MNIST and 100.0%/96.2% for CIFAR10, whereas ResNet101 has a top-1 accuracy rate 77.36% on the validation set. It only provides one adversarial attack, which is L_2 norm adoption of FGSM with 0.016 as the attack step. To make the results more comprehensive, we use JSMA, DeepFool, and C&W in Foolbox to craft adversarial examples with default parameters. Since Env_3 provides neither normal samples nor adversarial samples for evaluating detection approaches, we randomly select 1,000 normal samples from the testing dataset MNIST (respectively, CIFAR10 and ImageNet) that can be correctly classified by the underlying DNN models based on which adversarial examples are crafted by the considered adversarial attack methods.

To answer RQ3, we train models with backdoors based on the open source repositories [33, 58, 103] using three datasets (GTSRB [42], MNIST, and CIFAR10) and detect backdoored samples generated by the backdoor attacks: BadNets [33], PhysicalBA [59], ISSBA [57], WaNet [79], and TUAP [124]. The detection against BadNets is conducted based on the code and models provided by Gu et al. [33] and Wang et al. [103]. The accuracy on different clean test datasets are 96.51% (GTSRB), 99.17% (MNIST), and 98.89% (CIFAR10). For the other four recent backdoor attacks, we trained four models using the CIFAR10 dataset, one model per backdoor attack, whose accuracies on the clean test datasets are 94.09% (PhysicalBA), 94.94% (ISSBA), 91.13% (WaNet), and 92.00% (TUAP). The ASR is close to 100% on the inputs with backdoor triggers, and related code and models are released on GitHub [1]. To answer RQ4, both the models and input labels refer to the baseline work [80] for comparison.

5.2 RQ1: Effectiveness of Attack Costs

We answer RQ1 by comparing our approach with BL_1 through BL_4 in Env_1, Env_2, and Env_3, respectively. 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 larger the AUROC (up to 1), the better the approach.

The results are reported in Table 6, where the best one is highlighted in bold font. Note that all the tools in BL_1 through BL_3 do not contain tuned parameters for the ImageNet dataset, thus no result is reported. Although they could be adapted, we did not do so because it is non-trivial to reproduce their best performance, as mentioned in the work of Wang et al. [104]. Overall, we can observe that our approach outperforms the others in most cases. It is worth mentioning that our detection parameters stay the same in all three environments, Env_1, Env_2, and Env_3, which shows its universality, namely that users do not need to adjust parameters for a specific model or platform.

Among the four detections, 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, whereas DBA_d performs worse than the others in most cases. This is because DBA_d is a black-box attack, thus it 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 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 robustness. One may find that BL_2 performs better than the others on CIFAR10 adversarial examples crafted by FGSM. This may be because the performance of the model is too poor, as its accuracy is only 80.3% on the testing dataset.

Table 6. AUROC Comparison for Adversarial Examples

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	

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 (e.g., the model in Env$_2$) or improving the robustness of the model (e.g., AT, cf. Section 6.2.2).

Answer to RQ1: Against most attacks on two popular platforms (i.e., Keras and PyTorch) and three widely used datasets (i.e., MNIST, CIFAR10, and ImageNet), the selected white-box attacks JSMA$_d$, BIM$_d$, and BIM2$_d$ are more effective than the recent promising baselines BL$_1$ through BL$_4$.

5.3 RQ2: Adversarial Example Detection

We answer RQ2 by comparing our K-NN- and Z-score-based adversarial example detectors with BL_3 in 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 compare with BL_1, BL_2, and BL_4 since they *only* considered the results of AUROC or do *not* provide cost analysis. To avoid overfitting, we use different samples for training and evaluating our detectors.

5.3.1 Effectiveness. ***K-NN-Based Detectors.*** For each dataset, each detection attack$_d$ of BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$, and each attack a in Env_2 or Env_3, we build a K-NN-based detector using the attack costs of 1,000 benign examples and 1,000 attack a crafted adversarial examples via the detection attack$_d$. We also build a K-NN-based ensemble detector END, which consists of 1,000 benign examples and 1,000 adversarial examples, where each attack contributes 1,000/N adversarial examples (N is the number of attack methods) and $K = 100$.

The results are shown in Figure 6(a) through (c). On average, the accuracies of detectors JSMA$_d$, BIM$_d$, BIM2$_d$, DBA$_d$, and END are as follows:

— 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; and
— 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%, 100%, 99.4%, and 99.4%, respectively, and the FPR is 1.9% on MNIST. Similarly, on CIFAR10, the TPRs are 82.9%, 98.7%, 99.4%, and 89.9%, and the FPR is 7.6%, and on ImageNet, the TPRs are 90.0%, 99.6%, 99.2%, and 94.4%, and the FPR is 10.8%.

We find that DBA$_d$ performs worse than the others in most cases, which is consistent with AUROC (cf. Table 6). It is worth noting that although 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 detection attack$_d$ of BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$, we build a Z-score-based detector using the normal distribution of attack costs of 1,000 benign examples via the detection attack$_d$, resulting in four detectors BIM$_d$, BIM2$_d$, JSMA$_d$, and DBA$_d$. The threshold h is −1.281552, which yields a 10% FPR 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 two detectors classify the input as adversarial, and otherwise benign, namely $k = 2$.

The results are shown in Figure 6(d) through (f). On average, the accuracies of detectors JSMA$_d$, BIM$_d$, BIM2$_d$, DBA$_d$, and END are as follows:

— 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; and
— 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, and the FPR is 5.2% on MNIST. Similarly, on CIFAR10, the TPRs are 70.8%, 98.2%, 98.5%, and 78.9%, and the FPR is 9.8%, and on ImageNet, the TPRs are 97.2%, 100%, 99.4%, and 96.4%, and the FPR is 11.8%. 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.

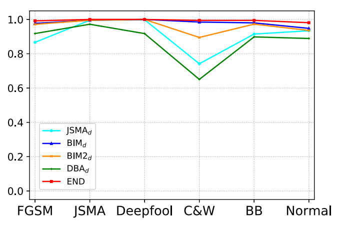
Table 7. Cost Analysis of Our Detector with Accuracy

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	≥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	≥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%

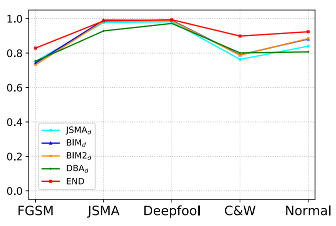
5.3.2 Efficiency. For a fair comparison with BL_3, we report the detection costs of the Z-score-based detectors here, although the detection accuracy may be slightly worse than that of K-NN-based detectors. The reason is that the threshold of Z-score detectors can be easily adjusted to ensure that detection 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 7. The columns $\#_{\text{adv}}$ and $\#_{\text{benign}}$ give the number of queries to the model for adversarial and benign examples on average, and t_i(ms) represents the average time required for each iteration. The columns Acc_{adv} and $\text{Acc}_{\text{benign}}$ respectively give the accuracy for adversarial and benign examples on average.

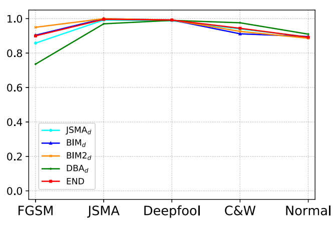
By limiting the accuracy on benign examples to the one of BL_3, we observe that all the white-box detectors (i.e., JSMA_d, BIM_d, and BIM2_d) outperform BL_3 in terms of the number of queries on both MNIST and CIFAR10. Furthermore, they also achieve better accuracy than BL_3 on CIFAR10, whereas both BIM_d and BIM2_d achieve better accuracy than BL_3 on MNIST. We also provide the results on ImageNet in Table 7. 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 detections 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 the MNIST dataset and the corresponding model as an example, BL_3 needs 3 ms to load the model and 0.6 ms 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 detection do not need any



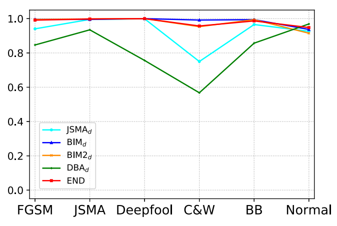
(a) K-NN, MNIST



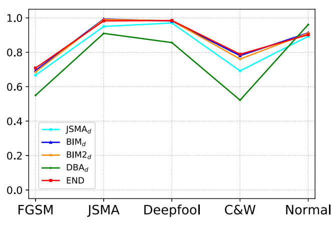
(b) K-NN, CIFAR10



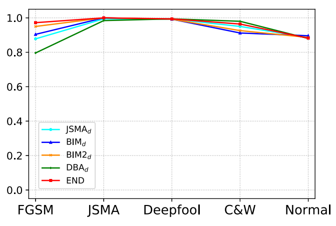
(c) K-NN, ImageNet



(d) Z-score, MNIST



(e) Z-score, CIFAR10



(f) Z-score, ImageNet

Fig. 6. Detection accuracy on adversarial examples crafted by different attacks, where the x-axis denotes the adversarial attack method but *normal* denotes normal examples, and the y-axis denotes the detection accuracy.

information of the models, hence using black-box attacks as detection preserve the privacy of the models while its effectiveness is still acceptable.

We emphasize there is still space for optimization. One way is to add an upper bound on the number of iterations, as adversarial examples often need fewer iterations than benign ones. If the number of iterations reaches the bound and the attack fails, the input can be considered as benign. This optimization could reduce the number of iterations (hence queries) for benign examples. When the bound is chosen according to the threshold of a Z-score detector, both TPRs and FPRs 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 detection according to the dimension of images. If one expects a more reliable and higher accurate detector, an ensemble detector such as END can be used. If the privacy of models 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^{2}D, we report results of turning parameters on the target model from Env$_2$ on MNIST. The results are shown in Table 8, where bold denotes the value used in the previous experiments (cf. Figure 6(a) and Figure 6(d)). The columns Acc_{benign} and Acc_{adv} respectively denote the detection accuracy for benign examples and adversarial examples on average.

For the 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, and when the proportion of adversarial examples increases, both TPRs and FPRs slightly increase.

For the Z-score-based ensemble detector, we vary the value of the threshold h from -1, -1.281552, -1.644854, and -1.959964 to -2. We observe that the smaller the threshold h, the lower the FPR. We change 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 and otherwise

Table 8. Comparison for the Impact of Classifier Parameters

Parameter	Value	$\mathrm{Acc}_{\mathrm{benign}}$	$\mathrm{Acc}_{\mathrm{adv}}$
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

benign. We observe from Table 8 that both true- and false-negative rates increase with the increase of k.

In summary, the preceding parameters can be used to balance the TPRs or FPRs, namely the TPR could be improved at the cost of the FPR.

Finally, instead of K-NN, we also tried the SVM (support vector machine), DTC (decision tree classifier), and RFC (random forest classifier) 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.

Answer to RQ2: A^2D is able to efficiently and effectively detect adversarial examples with a lower FPR. It is considerably more effective and efficient than the recent promising approach BL_3.

5.4 RQ3: Backdoored Sample Detection

To answer RQ3, we build the detector A^2D using the adversarial attack method BIM_d due to its performance in the previous experiments. The detector uses a two-step BIM, where the step size is the same as the one used for adversarial example detection in Section 5.2. We consider five

Table 9. AUROC Comparison for Backdoored Samples

Attack \ Detector	A^2D	BL_5	BL_6
BadNets	**0.9961**	0.9952	0.9568
PhysicalBA	**0.9657**	0.9282	0.8354
ISSBA	0.9694	**1.0**	<0.5
WaNet	**0.9095**	0.8104	<0.5
TUAP	**0.9682**	0.8586	<0.5
Average	**0.9618**	0.9185	0.6584

Table 10. TPR Comparison for Backdoored Samples Under the Same Fixed FPRs

Dataset	FPR = 0%			FPR = 0.5%			FPR = 1%			FPR = 5%			FPR = 10%		
	A^2D	BL_5	BL_6	A^2D	BL_5	BL_6	A^2D	BL_5	BL_6	A^2D	BL_5	BL_6	A^2D	BL_5	BL_6
GTSRB	**0.991**	0.979	0.57	**0.992**	0.993	0.749	**0.994**	0.993	0.795	**0.999**	0.996	0.871	**0.999**	0.998	0.904
MNIST	0.983	**0.991**	0.0	0.996	**1.0**	0.0	0.996	**1.0**	0.0	**1.0**	**1.0**	0.667	**1.0**	**1.0**	**1.0**
CIFAR10	**0.957**	0.954	0.0	**0.964**	0.962	0.0	**0.974**	0.964	0.0	**0.989**	0.980	0.0	0.991	0.989	**0.998**
Average	**0.977**	0.975	0.19	0.984	**0.985**	0.25	**0.988**	0.986	0.265	**0.996**	0.992	0.513	**0.997**	0.996	0.967

backdoor attacks: BadNets, ISSBA, WaNet, PhysicalBA, and TUAP, covering different types of backdoor triggers such as clean-label, optimized trigger pattern, invisible, and sample-specific. We compare A^2D with BL_5 and BL_6, where BL_5 needs backdoored samples for training detectors and BL_6 does not, the same as A^2D. We first compare AUROC of the detection tools to evaluate the effectiveness of the detection metric using the CIFAR dataset and then compare the TPR of the detection tools under different FPRs using the most classic backdoor attack BadNets and the datasets GTSRB, MNIST, and CIFAR10. Each dataset used for comparison has 1,000 normal samples and 1,000 backdoored samples.

The AUROC results are reported in Table 9. Since a random predictor has an AUROC score of 0.5, we do not specifically calculate the AUROC results when it is less than 0.5. The results show that A^2D achieves the best average AUROC and has a relatively balanced performance against different backdoor attacks, indicating that A^2D is able to detect different backdoored samples. More specifically, BL_5 performs better than our detector A^2D against the backdoor attack ISSBA, but it performs worse than A^2D against the other backdoor attacks, particularly WaNet and TUAP. BL_6 performs poorly against the backdoor attacks ISSBA, WaNet, and TUAP, because these attacks generate more invisible or arbitrary backdoor triggers than BadNets and PhysicalBA, which add fixed-size box-shaped backdoor triggers at fixed positions.

The TPR results are reported in Table 10. Although the three methods are often comparable, A^2D performs slightly better than the baselines on average, particularly on the GTSRB and CIFAR10 datasets when FPR is no more than 5%. We note that images in GTSRB and CIFAR10 ($32\times32\times3$) have higher dimensions than MNIST images ($28 \times 28 \times 1$). This means that A^2D has a better capability than the baselines for handling high-resolution images when the FPR is small. We remark that a small FPR is required in practice.

To better understand the reason behind the diverse performance, we inspect the methodology of the baselines. BL_5 assumes that normal and backdoored samples have different distributions of the outputs of intermediate layers and clusters the outputs of intermediate layers for normal

Table 11. TPR and FPR Comparison for Backdoored Samples Under Calculating Thresholds Using 100 Samples

Dataset	FPR* = 0%		FPR* = 1%		FPR* = 5%		FPR* = 10%	
	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR
GTSRB	0.028 ($\uparrow$0.028)	0.997 ($\uparrow$0.006)	0.036 ($\uparrow$0.026)	0.998 ($\uparrow$0.004)	0.069 ($\uparrow$0.019)	0.999 (–)	0.094 ($\downarrow$0.006)	0.999 (–)
MNIST	0.01 ($\uparrow$0.01)	0.996 ($\uparrow$0.013)	0.014 ($\uparrow$0.004)	0.998 ($\uparrow$0.002)	0.057 ($\uparrow$0.007)	1.0 (–)	0.097 ($\downarrow$0.003)	1.0 (–)
CIFAR10	0.003 ($\uparrow$0.003)	0.957 (–)	0.011 ($\uparrow$0.001)	0.974 (–)	0.047 ($\downarrow$0.003)	0.979 ($\downarrow$0.1)	0.099 ($\downarrow$0.006)	0.989 ($\downarrow$0.02)

and backdoored samples as different categories. The clustering process combines PCA (principal component analysis) and K-means algorithms in an N-classification problem, which performs N clustering operations in advance until the presence of the backdoor Trojan in the network is identified and the corresponding output label is found. Compared to A^{2}D, BL$_5$ requires backdoored samples to train a detector while A^{2}D does not. Although both BL$_5$ and A^{2}D use intermediate layer outputs, which may be the reason they have similar performance, A^{2}D uses intermediate layer outputs to evaluate the attack cost and adversarial robustness, whereas BL$_5$ uses all intermediate layer outputs for clustering. BL$_6$ assumes that backdoored samples will be more stable and performs detection by overlaying new images on input samples. Thus, its effectiveness depends on whether the trigger will be overwritten by the overlaid image. Although BL$_6$ does not directly use robustness metrics, it indeed implicitly assumes that the backdoored samples are more robust, in line with our conclusion of Table 1. The performance of A^{2}D indicates that, compared with overlaying the original images with randomly selecting new images, adversarial attacks provide a better way to characterize robustness. Overall, the experimental results demonstrate that A^{2}D is a promising approach for detecting backdoored samples in a wide range of image classification tasks.

We finally remark that in terms of efficiency, the three detection methods (A^{2}D, BL$_5$, and BL$_6$) are comparable. Indeed, the average detection time of BL$_5$, BL$_6$, and A^{2}D is 0.067 seconds, 0.051 seconds, and 0.053 seconds, respectively.

One may notice that we used 1,000 normal samples to determine the threshold M_t by choosing an acceptable FPR r. However, obtaining 1,000 normal samples may be time consuming for the defender in practice. We argue that the defender indeed can use fewer normal samples to determine the threshold M_t. Table 11 shows the FPR and TPR results on the 1,000 normal samples and 1,000 backdoored samples, where the threshold M_t is determined by choosing an acceptable FPR r on 100 randomly chosen normal samples (denoted by FPR* in Table 11). We can observe that when FPR* is small (i.e., $\leq 5\%$), both FPR and TRP often increase slightly; when FPR* is large (i.e., $= 10\%$), both FPR and TRP often decrease slightly. These results demonstrate the generality and flexibility of our detection approach.

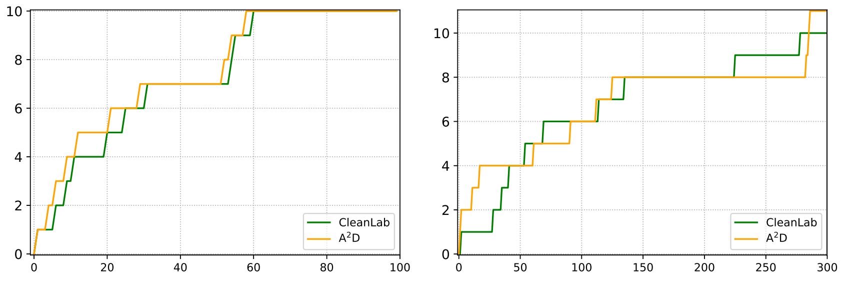
Answer to RQ3: A^{2}D is able to effectively detect different types of backdoored samples, and always outperforms the baselines for various high-resolution image datasets and backdoor attacks when the FPR is limited.

5.5 RQ4: Mislabeled Sample Detection

To answer RQ4, we compare A^{2}D with the state-of-the-art tool BL$_7$ (i.e., CleanLab [80, 81]). Similar to RQ3, we build the detector A^{2}D using the adversarial attack method BIM$_d$. When sorting suspected mislabeled samples, BL$_7$ uses the classification probabilities of different labels, whereas A^{2}D primarily uses robustness (i.e., attack costs). The output probabilities are sensitive to training methods (e.g., [85]), whereas attack costs are more stable. We conduct experiments on testing

Table 12. Comparison for Mislabeled Sample Detection

Dataset	BL_7		A^2D	
	Prune	Threshold	Prune	Threshold
MNIST	7/34	10/100	7/34	10/100
CIFAR10	8/184	10/300	8/184	11/300



(a) Ranking on MNIST (b) Ranking on CIFAR10

Fig. 7. Ranking of mislabeled samples among suspected samples, where the x-axis (respectively, the y-axis) is the number of suspected (real mislabeled) samples.

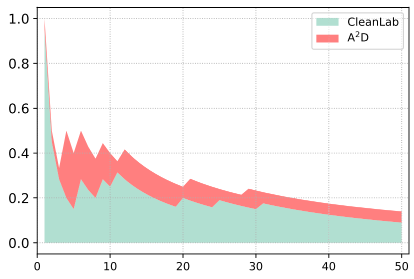
datasets of MNIST and CIFAR10. Since A^2D and BL_7 detect different numbers of suspected samples, the experimental results are compared under the prune detection method used in BL_7 (denoted by Prune) and the threshold-based detection method proposed by us (denoted by Threshold). We fuse the detection indicators and detection methods of the two methods with each other, to ensure fairness of the comparison. We set λ to 1 in the loss function $loss(\cdot)$.

The results are reported in Table 12, where a/b denotes that b number of suspected mislabeled samples are found which contain a number of real mislabeled samples using the ground-truths listed at https://labelerrors.com that were detected by BL_7 and manually confirmed by human workers on Amazon Mechanical Turk.

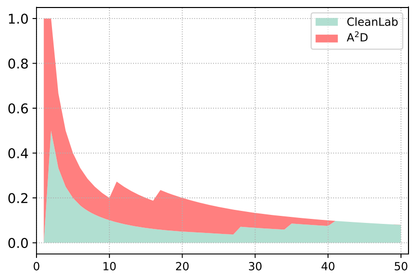
It can be found that A^2D achieves similar results to the state-of-the-art method BL_7. In Figure 7, we further analyze the number of real mislabeled samples among all suspected samples along with increasing of the threshold. The results reveal that A^2D is more effective when the number of suspected mislabeled samples is limited to no more than 50, suggesting that it is easier to find real mislabeled samples using attack costs. Similarly, in Figure 8, we show the percentage of real mislabeled samples over the first 50 suspected mislabeled samples. It can be observed that A^2D has a low FPR when the number of suspect samples is small.

As a case study, Figure 9(a) shows the first suspected mislabeled sample found by A^2D. It is easy to see that Figure 9(a) is mislabeled as frog but should be cat. However, it is at 33rd place by BL_7. Instead, the first suspected mislabeled sample found by BL_7 is shown in Figure 9(b)) whose assigned label is cat without mislabeling. This image is at 46th place by our detector A^2D. Thus, our approach using robustness as the indicator is considerably better than BL_7 that uses probability as the indicator.

To concretely demonstrate the advantage of A^2D on this kind of clearly mislabeled instances, we randomly inject 1,000 incorrectly labeled images into CIFAR10. Among the top-1,000 suspected mislabeled samples reported by both tools, the detection accuracy of A^2D is 88%, whereas the accuracy of BL_7 is 86.1%.

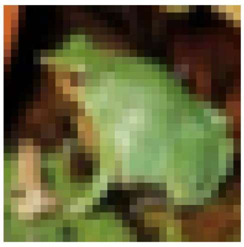


(a) The percentage on MNIST

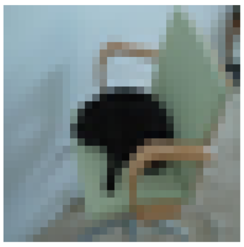


(b) The percentage on CIFAR10

Fig. 8. The percentage of real label errors over the number of suspected samples, where the x-axis (respectively, the y-axis) is the number of suspected (real mislabeled) samples.



(a) A^2D, frog labeled as cat



(b) BL_7, cat

Fig. 9. The 1_{st} ranked suspect samples found by A^2D and BL_7.

We finally remark that in terms of efficiency, A^2D is slower than BL_7. Specifically, the average detection time of A^2D and BL_7 is 0.39 seconds and 0.011 seconds for the MNIST dataset, and 0.92 seconds and 0.046 seconds for the CIFAR10 dataset, respectively. This is because mislabeled samples are often quite robust to their predicted labels, thus the attack cost is higher when the predicted label is used as the ground truth for an adversarial attack. We argue that in contrast to adversarial and backdoored samples which typically require an online real-time detection, mislabeled samples already exist in the dataset for which an offline detection is sufficient. Therefore, the time cost of A^2D for mislabeled sample detection is affordable.

Answer to RQ4: A^2D is able to quickly detect mislabeled samples, especially those clearly mislabeled samples.

5.6 Threats to Validity

The threats to validity of our study include external and internal threats. The datasets and models used in our evaluation could be one of the external threats. We tried to counter this threat by using several widely used datasets and pre-trained models from well-established works. Since our approach leverages difference in robustness and attack cost between normal and abnormal samples, models need to be well trained to have good and stable robustness. Under the Env_1 and CIFAR10 setting, our approach fails to outperform the baseline BL_2 on the FGSM adversarial examples. Our analysis finds that this is because the robustness distribution of the normal samples became positively skewed due to the low accuracy of the model, only about 80%. The well-trained model is not a severe restriction, as developers will strive to train models better for their tasks, but this is potentially one of the reasons our detectors make mistakes.

Another external threat is the knowledge of the adversary. As similarly assumed by the baseline approaches, we assume that the adversary is unaware of the existence of the detection and evaluate our detectors against the original models. In practice, the adversary may learn the deployment of a detector and even know all the details of the detector, via social engineering or other methods, and use a more threatening, specified attack method, called *adaptive attacks* [101]. This is the biggest external threat against the defense, where the adversary may craft tailored adversarial examples to bypass the detection. We discuss and evaluate adaptive attacks against our adversarial example detection method in Section 6. We do not perform adaptive attack experiments on the detection of the remaining abnormal samples, as we believe that experiments on the adversarial examples can illustrate the huge cost required to bypass A^{2}D and we have the ability to mitigate it.

The experimental platforms may also be an external threat. To mitigate this threat, we compare with multiple baselines in different platforms, and these baselines are widely used for comparison in the literature [39, 55, 69, 92]. It is worth noting that the comparison of baselines was conducted on the open source tools with parameters provided by the original authors, to reproduce their best performance.

The internal threat mainly comes from the gap between attack cost and robustness. We use attack cost to efficiently estimate robustness and subsequently use A^{2}D to detect abnormal samples. However, attack cost may not be accurate enough to estimate robustness. To mitigate this threat, we studied various attack methods which may differ in their capability. Experimental results indicate that our detection performs well regardless of the selected attack method, although a minor difference can be observed. When more efficient robustness verification tools are proposed, our methodology could be directly adopted instead of using attack costs to measure robustness.

Another possible internal threat is the lack of interpretability of DNNs. In terms of mislabeled sample detection, some samples may be classified into the wrong label with high confidence, and this may lead to the misclassification of our detection method. In essence, if we could understand why a DNN makes this prediction, we could use this information to detect abnormal samples better. Similarly, such an understanding would be helpful in mitigating backdoor attacks.

6 ON ADAPTIVE ATTACKS

Multiple effective defenses have been shown to be ineffective in the presence of adaptive attacks [14–16, 39, 101], where the adversary knows all the details of the DNN model under attack and the deployed defense, and manages to devise a specific and powerful adversarial attack method to bypass defense. Thus, it is important to consider adaptive attacks when evaluating defense approaches. In Section 5, we have shown the ability of A^{2}D to detect adversarial examples under the threat model that the adversary is unaware of the existence of defense mechanisms. In this section, we investigate possible adaptive attacks to our detection, and the threat model turns into the adversary knowing all the details of our detection as well as the DNN model under attack. Backdoored and mislabeled samples are outside the scope of this section.

6.1 Potential Bypass Approaches

We consider two approaches which may potentially be used to bypass our detection. The adaptive attack increases either the attack costs or robustness of adversarial examples which are exploited in our detection.

6.1.1 Increasing Attack Costs. A straightforward approach that may be used to bypass our detection is to directly increase the attack costs so that the attack costs of adversarial and benign

Table 13. Robustness vs. Confidence of Adversarial Examples

	κ	0	2	4	6	8
MNIST	CLEVER score	≈ 0	0.11	0.14	0.14	0.17
	Attack iterations of BIM$_d$	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
	Attack iterations of BIM$_d$	1.37	8.53	17.29	24.47	34.4
	L_2 distance	0.41	0.52	0.67	0.82	0.99

examples are similar. To directly 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

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

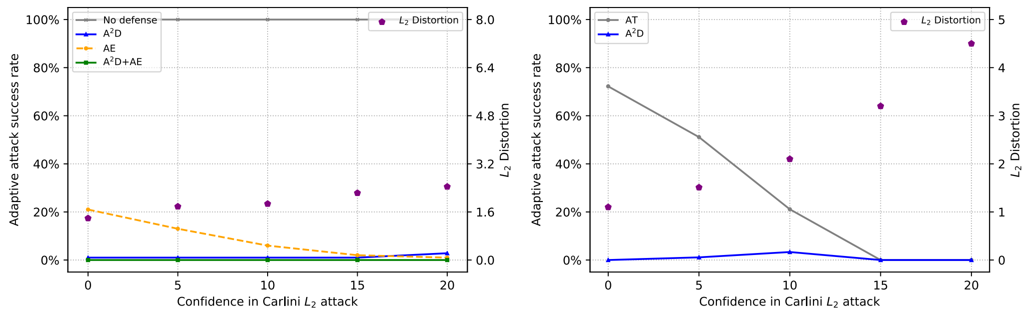
where $J(x)$ is the original loss function, β is a parameter for balancing the terms $J(x)$ and $\max(\texttt{cost} - \texttt{attack\_cost}(x), 0)$, `cost` denotes the expected attack cost such as the mean of attack costs of benign examples or even the threshold of our Z-score-based detection approach, and $\texttt{attack\_cost}(x)$ denotes the attack cost of the sample x via some adversarial attack methods. Minimizing the loss function $J'(x)$ increases the attack cost of the sample x until exceeding `cost`. However, this loss function $J'(x)$ is non-differentiable and hence cannot be solved via gradient-based algorithms adopted by almost all the white-box adversarial attack methods. Moreover, non-gradient-based iterative attacks have to run some adversarial attacks internally during each optimization iteration to compute $\texttt{attack\_cost}(x)$), which definitely results in high computational complexity.

6.1.2 Increasing Robustness. An alternative approach that may be used to bypass our detection 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 [16] 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 apply the adversarial attack method C&W using the first 100 MNIST and 100 CIFAR10 images from their testing datasets under the same setting as in the work of Carlini and Wagner [15], by varying the value of κ and measuring the robustness using the CLEVER scores and attack iterations of BIM$_d$. The results are reported in Table 13. The experiment results show that the adversary is able to increase the robustness in terms of both the CLEVER scores and attack iterations of adversarial examples by increasing the confidence. Therefore, high-confidence adversarial examples have the potential to bypass our detection. However, we observe from Table 13 that the distortion in terms of the L_2 norm increases with the increase of the confidence parameter κ.



(a) A^{2}D + AE (b) A^{2}D + AT

Fig. 10. Adaptive attack results, where the line and dot denote the attack success rate and L_2 distortion, respectively.

As our detection neither changes inputs or models, it can be seamlessly integrated with other defenses. Combining our detection with other defenses that are aimed at detecting adversarial examples with large distortion (e.g., [66, 75, 88]) would be able to detect a wide spectrum of adversarial examples. Combining our detection with AT [32, 71, 98] would be able to further enhance the model. Indeed, a successful attack to an adversarially trained model often introduces large distortion, whereas increasing the robustness of adversarial examples to bypass our detection also introduces large distortion. Consequently, to bypass our detection on an adversarially trained model would introduce much large distortion. When perturbations are limited to those that are human imperceptible, it becomes difficult to bypass our detection on an adversarially trained model.

6.2 Evaluation of Adaptive Attacks

Since the first adaptive attack is infeasible, we only evaluate the second one that is implemented based on C&W [15]. We evaluate this adaptive attack by varying the parameter κ from 0 to 20.

To evaluate the effectiveness of our detection combined with other defenses, we consider the AE-based detector [75] and PGD AT [71]. The AE trains a classifier f_{ae} based on benign examples to detect any adversarial examples with large distortion by checking if $d(x, f_{ae}(x))$ is greater than a pre-set threshold τ, where d is a distance function—for example, the mean squared error $\|x - f_{as}(x)\|_2$.

6.2.1 A^2D with AE. For ease of evaluation, we conduct experiments using the MNIST testing dataset under the same settings as in the work of Meng and Chen [75], 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 [52]. 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 detection. We use BIM$_d$ as detection and the corresponding Z-score-based detector that 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 10(a), from which we can observe that without any defense, the 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 $\kappa \leq 15$, whereas 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

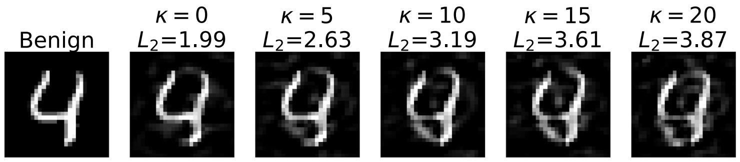


Fig. 11. Adversarial examples for different κ.

adversarial examples can bypass AE when $\kappa = 0$, whereas all adversarial examples can be detected by AE when $\kappa = 20$. Thus, the ASR is always 0% when the combined defense A^{2}D + AE is applied.

Summary. The preceding 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 attacks.

Case Study. Figure 11 shows adversarial examples of a targeted attack from 4 to 0 with different values of the parameter 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 testing dataset under the same settings as in the work of Madry et al. [71], which provides an adversarially trained DNN 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 [52]. We use the same Z-score-based detector as in Section 6.2.1.

Results. The results are shown in Figure 10(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 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 detection A^{2}D is good at detecting adversarial examples with small distortion (i.e., low confidence). Therefore, the combined defense is quite 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 adaptive attack achieves no more than 3% ASR on the adversarially trained model.

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

Case Study. Figure 12 shows adversarial examples of targeted attacks from 'truck' and 'airplane.' The adversarial examples in the first/third line are classified as 'cat,' and the adversarial examples in the second/fourth line are classified as 'horse.' Without any defense, an adversarial example with less distortion can be crafted (cf. Figure 12(b)). With AT, it requires more distortion to craft an adversarial example (cf. Figure 12(c)). If both A^{2}D and AT are enabled, it requires much more distortion to craft adversarial examples (cf. Figure 12(d)). Now the distortion is too large to be human imperceptible, and in some adversarial examples, we can clearly see the silhouettes of a targeted label on the adversarial example.

7 RELATED WORK

As a new type of software system, neural networks have received extensive attention over the past 5 years. We summarize related works from three dimensions: abnormal examples, abnormal

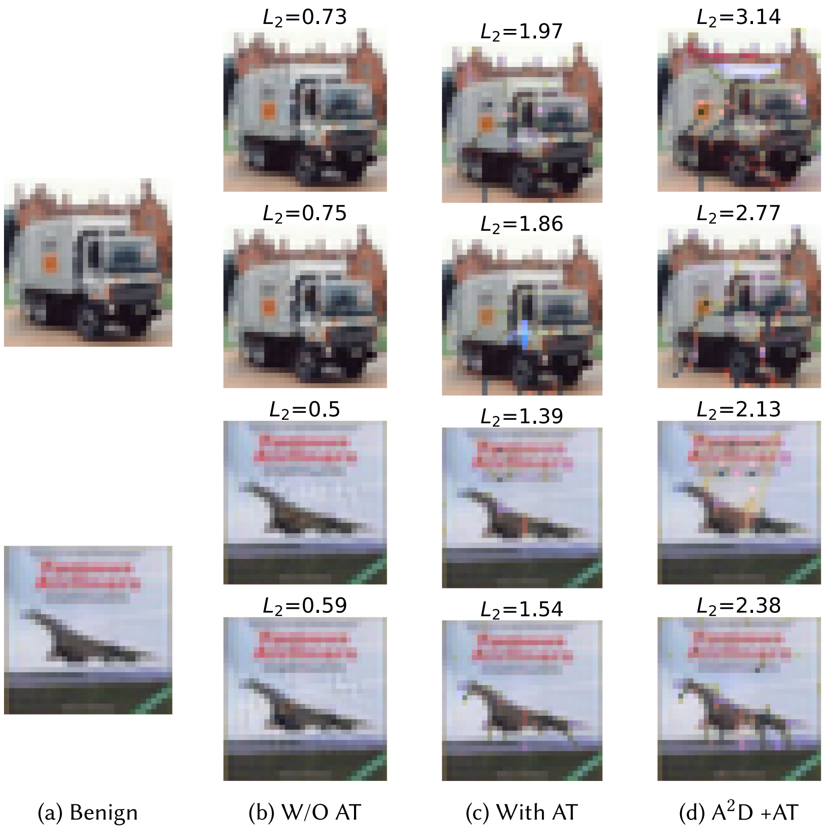


Fig. 12. Adversarial examples on the models that are benign (a), without AT (b), with AT (c), and with A^2D + AT (d).

example defense, and neural network testing and verification. We are not able to cover all of them but only discuss the most relevant ones.

Abnormal Examples. This work focuses on the detection of three kinds of abnormal samples, namely adversarial, backdoored, and mislabeled samples. Adversarial attacks aim to misjudge the neural network by adding perturbations that are imperceptible to humans. We have introduced several representative white-box attacks in Section 2 and used multiple adversarial attack methods (i.e., FGSM [32], BIM [51], JSMA [84], DeepFool [76], C&W [16], and substitute model attack [82]) to craft adversarial examples. We also notice that some adversarial example attack methods have the ability to carry out attacks in the physical environment and thus pose serious threats to neural networks [26, 45, 51]. In this work, we propose to characterize adversarial examples via robustness and attack costs based on which we showed that many adversarial example attack methods can be leveraged by our detection approach for computing attack costs, and our detection approach is quite effective for defending against these adversarial example attack methods.

Backdoor attack, as a severe hazard in the model supply chain, federal learning, and open source datasets, has been thoroughly explored in recent years. The poisoned datasets or models are treated as Trojan horses buried in a neural network programming paradigm. The manner of adding backdoor triggers in images is becoming increasingly stealthy, ranging from fixed backdoor triggers [33] to sample-specific triggers [57] and invisible backdoor triggers [79]. The effectiveness of backdoor attacks in the physical world is also receiving attention [59]. Besides poisoning the training dataset with maliciously labeled samples, a backdoor can also be injected by clean-label samples [124], modifying the model parameters directly [65], or utilizing the dropout technique to inject a triggerless backdoor attack [90]. Backdoors can also be used as watermarks for IP protection [56, 97]. In this work, we showed that backdoored samples have different robustness from normal samples and our detection approach is effective in detecting backdoored samples generated by various backdoor attack methods across different datasets.

Label quality plays a significant role in the generalization of models for supervised learning [22, 128]. However, it is not easy to exclude mislabeled samples from a dataset, especially when the labeling of samples requires specialized knowledge [29, 109]. In this work, we studied the robustness difference between mislabeled and normal samples based on which we propose a novel approach to effectively detect mislabeled samples.

Abnormal Example Defense. To mitigate abnormal examples without rejecting input samples, various approach have been proposed, such as defending against adversarial examples via AT [71] and input sample pre-processing [12, 34, 75], eliminating backdoors from models [36, 60, 63], mitigating mislabeled samples via robust loss [73], sample weighting [110], and mixture models [72]. Detection is another line of defense approach to abnormal examples, where an input will be rejected without being fed to the model if the input is detected as abnormal. Although various detection approaches have been proposed respectively addressing some types of abnormal examples, it is fair to say that this problem remains unsolved.

Adversarial example detection methods often characterize adversarial examples from multiple perspectives of the data, like density estimates [28], AE [75], and PCA [40]. Other detection methods are based on the model outputs and the hidden layer results, such as Bayesian uncertainty estimates [28], local intrinsic dimensionality [70], mutation testing [106], linear regression on hidden neuron outputs [104], Mahalanobis distance on Gaussian discriminant analysis [55], feature squeezing [115], and feature compression [20]. Kim et al. [48] and Zhong et al. [127] respectively argued that adversarial examples and natural variations (images generated through a variety of image transformations, e.g., rotating an image, changing the lighting conditions) are close to the decision boundaries of the DNN model in the feature space, based on which they proposed approaches to detect adversarial examples and natural variations. Most existing adversarial example defenses except for AT suffer from the following drawbacks: (1) they can be easily bypassed by adaptive attacks with backward pass differentiable approximation and expectation over transformation [5, 6, 101], (2) they are not effective on high-resolution images (e.g., the ImageNet dataset) [28, 61, 70, 75, 106, 127], and (3) they either did not consider or considered limited adaptive attacks [28, 61, 70, 75, 104, 106, 127]. In this work, we propose to effectively and efficiently measure robustness via non-differentiable attack costs, thus gradient-based white-box attacks cannot be directly applied to our detection. The experimental results show that A^2D is effective on widely used datasets covering both low- and high-resolution images, and effective in defending against potential tailored powerful adaptive attacks when combined with other alternative defense solutions.

For the detection of backdoor attacks, one can determine whether a model has been tampered with a backdoor Trojan [2, 49, 103] or an input sample has been injected with a backdoor trigger [17, 30, 35]. Neural Cleanse [103] determines whether a network has been injected with a backdoor Trojan by calculating the minimum perturbation needed to attack an input to a specific class. Kolouri et al. [49] design Universal Litmus Patterns, feed them to a model, and classify the model as poisoned or clean after analyzing the model's output logits. Aiken et al. [2] propose to detect and locate possible Trojan intermediate layers by directly thresholding the average activation difference between normal and backdoored samples. For a given input image, Activation

Clustering [17] leverages the activation difference from the neural network's hidden layers between normal and backdoored samples to detect backdoor. Strip [30] creates multiple images by superimposing various image patterns and utilizes the entropy of their predicted classes as an detection indicator. Whereas these works detect backdoored samples from different perspectives, our approach focuses on the use of adversarial attacks to represent the difference in robustness between normal and backdoored samples. The experimental results demonstrate that our approach is often able to achieve better performance on inputs with different backdoor triggers.

A similar circumstance holds for detecting mislabeled instances and cleanse datasets. INCV [21] applies cross-validation to randomly split noisy datasets to identify most samples that have correct labels. MentorNet [46] trains an extra network called *MentorNetwork* to supervise the base network to focus on the samples whose label are probably correct. CleanLab [80, 81] detects label errors by directly estimating the joint distribution of latent true label and noise label. These works can only detect the samples whose assigned labels are not equal to the predicted labels. The experimental results show that our approach not only detects such kinds of samples efficiently but also detects the potential mislabeled samples when the assigned label is equal to the model predicted label,

Furthermore, our detection method A^2D utilizes the inherent robustness difference between normal and adversarial examples, which has been systematically and comprehensively investigated in terms of CLEVER scores. Putting different kinds of abnormal examples together, our approach is a generalized method to detect abnormal examples by characterizing the robustness difference between normal and abnormal examples via low-overhead attack costs.

Neural Network Testing and Verification. Some works look for vulnerabilities in neural networks from the perspective of software testing. DeepXplore [86] leverages a testing method to find adversarial examples guided by neuron coverage. After that, a series of coverage criteria has been proposed for neural network testing [48, 67, 95]. Different testing methods also have been adapted to test neural networks, such as concolic testing [96], mutation testing [68], and others [105, 113]. We do not use testing criteria to model the robustness of examples, as testing criteria are not necessarily correlated with robustness [24, 117] and could be misleading [61].

Various formal verification techniques have been proposed to verify the robustness property of neural networks [7, 31, 37, 47, 62, 64, 107, 120–123, 126]. Formal verification provides provable or theoretic guarantees, and robustness is also the source of our detection approach. However, current formal verification techniques in general have high computational complexity. Thus, we use attack costs for better scalability. In the future, once efficient formal verification techniques for verifying robustness of DNNs are proposed, they could be leveraged to detect abnormal examples following our detection methodology.

8 CONCLUSION

Inspired by the robustness of neural networks, we proposed a novel characterization of abnormal examples via robustness and systematically evaluated robustness of abnormal examples in terms of CLEVER scores. Subsequently, based on the characterization, we proposed a novel detection approach, named *attack as detection* (A^2D), which utilizes adversarial attack methods to effectively measure robustness, and do not need to modify or retrain the protected model. We conducted extensive experiments to evaluate our detection approach A^2D, showing that it outperforms recent promising approaches in adversarial, backdoored, and mislabeled sample detection. Our tool A^2D can be applied in the real-world scenario as follows when the three types of abnormal samples come all at once. First, mislabeled samples are checked before the training and evaluation of a DNN model and thus need not be checked after the model has been deployed. Next, our tool A^2D can iteratively check if an input to a deployed DNN model is an adversarial example or a backdoored sample. We also thoroughly discussed and evaluated the main threat (i.e., adaptive attacks) to our approach in detecting adversarial examples. By combing our detection approach with an existing detection approach or AT, the results are very promising—for example, the ASR drops from 72% to 0% on CIFAR10, and it drops from 100% to 0% on MNIST.

REFERENCES

[1] GitHub. 2022. A^2D. Retrieved November 21, 2023 from https://github.com/S3L-official/attack-as-detection

[2] William Aiken, Hyoungshick Kim, Simon Woo, and Jungwoo Ryoo. 2021. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Computers & Security* 106 (2021), 102277.

[3] Apollo. 2018. Apollo: An Open, Reliable and Secure Software Platform for Autonomous Driving Systems. Retrieved November 21, 2023 from http://apollo.auto

[4] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering*. 1–10.

[5] 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.

[6] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing robust adversarial examples. In *Proceedings of the International Conference on Machine Learning*. 284–293.

[7] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. 2016. Measuring neural net robustness with constraints. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 2613–2621.

[8] B. L. Welch. 1947. The generalisation of 'student's' problems when several different population variances are involved. *Biometrika* 34, 1-2 (1947), 28–35.

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

[10] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *Proceedings of the 6th International Conference on Learning Representations*.

[11] Lei Bu, Zhe Zhao, Yuchao Duan, and Fu Song. 2022. Taking care of the discretization problem: A comprehensive study of the discretization problem and a black-box adversarial attack in discrete integer domain. *IEEE Transactions on Dependable and Secure Computing* 19, 5 (2022), 3200–3217.

[12] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer encoding: One hot way to resist adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations*.

[13] Nicholas Carlini and David Wagner. 2016. Defensive distillation is not robust to adversarial examples. *CoRR abs/1607.04311* (2016).

[14] 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.

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

[16] Nicholas Carlini and David A. Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*. 39–57.

[17] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian M. Molloy, and Biplav Srivastava. 2019. Detecting backdoor attacks on deep neural networks by activation clustering. In *Proceedings of the Workshop on Artificial Intelligence Safety, Co-Located with the 33rd AAAI Conference on Artificial Intelligence*.

[18] Guangke Chen, Sen Chen, Lingling Fan, Xiaoning Du, Zhe Zhao, Fu Song, and Yang Liu. 2021. Who is real Bob? Adversarial attacks on speaker recognition systems. In *Proceedings of the IEEE Symposium on Security and Privacy*. 694–711.

[19] Guangke Chen, Yedi Zhang, Zhe Zhao, and Fu Song. 2023. QFA2SR: Query-free adversarial transfer attacks to speaker recognition systems. In *Proceedings of the 32nd USENIX Security Symposium*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 2437–2454.

[20] Guangke Chen, Zhe Zhao, Fu Song, Sen Chen, Lingling Fan, Feng Wang, and Jiashui Wang. 2023. Towards understanding and mitigating audio adversarial examples for speaker recognition. *IEEE Transactions on Dependable and Secure Computing* 20, 5 (2023), 3970–3987. https://doi.org/10.1109/TDSC.2022.3220673

[21] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. 2019. Understanding and utilizing deep neural networks trained with noisy labels. In *Proceedings of the 36th International Conference on Machine Learning*. 1062–1070.

[22] Filipe R. Cordeiro and Gustavo Carneiro. 2020. A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*.

[23] Zijun Cui, Yong Zhang, and Qiang Ji. 2020. Label error correction and generation through label relationships. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, the 32nd Innovative Applications of Artificial Intelligence Conference, and the 10th AAAI Symposium on Educational Advances in Artificial Intelligence*. 3693–3700.

[24] 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 International Conference on Engineering of Complex Computer Systems*. 73–82.

[25] Yizhak Yisrael Elboher, Justin Gottschlich, and Guy Katz. 2020. An abstraction-based framework for neural network verification. In *Computer Aided Verification*. Lecture Notes in Computer Science, Vol. 12224. Springer, 43–65.

[26] 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 the IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634.

[27] Herbert Federer. 2014. *Geometric Measure Theory*. Springer.

[28] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. 2017. Detecting adversarial samples from artifacts. *CoRR abs/1703.00410* (2017).

[29] Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks* 25 (2014), 845–869.

[30] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. Strip: A defence against Trojan attacks on deep neural networks. In *Proceedings of the Annual Computer Security Applications Conference*. 113–125.

[31] 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 IEEE Symposium on Security and Privacy*. 3–18.

[32] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations*.

[33] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

[34] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. 2018. Countering adversarial images using input transformations. In *Proceedings of the 6th International Conference on Learning Representations*.

[35] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. 2023. SCALE-UP: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. *arXiv preprint arXiv:2302.03251* (2023).

[36] Wenbo Guo, Lun Wang, Yan Xu, Xinyu Xing, Min Du, and Dawn Song. 2020. Towards inspecting and eliminating Trojan backdoors in deep neural networks. In *Proceedings of the IEEE International Conference on Data Mining*. 162–171.

[37] Xingwu Guo, Wenjie Wan, Zhaodi Zhang, Min Zhang, Fu Song, and Xuejun Wen. 2021. Eager falsification for accelerating robustness verification of deep neural networks. In *Proceedings of the IEEE International Symposium on Software Reliability Engineering*. 345–356.

[38] Warren He, Bo Li, and Dawn Song. 2018. Decision boundary analysis of adversarial examples. In *Proceedings of the 5th International Conference on Learning Representations*.

[39] 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*.

[40] Dan Hendrycks and Kevin Gimpel. 2017. Early methods for detecting adversarial images. In *Proceedings of the 5th International Conference on Learning Representations*.

[41] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. 2019. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Vol. 33. 3771–3778.

[42] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. 2013. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *Proceedings of the International Joint Conference on Neural Networks*.

[43] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review* 37 (2020), 100270.

[44] 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.

[45] Steve T. K. 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 33rd AAAI Conference on Artificial Intelligence*.

[46] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proceedings of the 35th International Conference on Machine Learning*. 2304–2313.

[47] 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.

[48] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *Proceedings of the 41st International Conference on Software Engineering*. 1039–1049.

[49] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. 2020. Universal litmus patterns: Revealing backdoor attacks in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 301–310.

[50] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.

[51] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *Proceedings of the 5th International Conference on Learning Representations*.

[52] Madry Lab. 2020. MNIST and CIFAR10 Adversarial Examples Challenges. Retrieved November 21, 2023 from https://github.com/MadryLab

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

[54] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. 1998. The MNIST Database of Handwritten Digits. Retrieved November 21, 2023 from http://yann.lecun.com/exdb/mnist/index.html

[55] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 7167–7177.

[56] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. 2022. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[57] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16463–16472.

[58] Yiming Li, Mengxi Ya, Yang Bai, Yong Jiang, and Shu-Tao Xia. 2023. BackdoorBox: A Python toolbox for backdoor learning. In *Proceedings of the 2023 ICLR Workshop*.

[59] Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2021. Backdoor attack in the physical world. *arXiv preprint arXiv:2104.02361* (2021).

[60] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. 2020. Rethinking the trigger of backdoor attack. *CoRR abs/2004.04692* (2020).

[61] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. 2019. Structural coverage criteria for neural networks could be misleading. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results*. 89–92.

[62] Jiaxiang Liu, Yunhan Xing, Xiaomu Shi, Fu Song, Zhiwu Xu, and Zhong Ming. 2022. Abstraction and refinement: Towards scalable and exact verification of neural networks. *CoRR abs/2207.00759* (2022).

[63] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*. 273–294.

[64] 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.

[65] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Proceedings of the Annual Network and Distributed System Security Symposium*.

[66] 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.

[67] 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.

[68] 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 *Proceedings of the IEEE International Symposium on Software Reliability Engineering*. 100–111.

[69] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 2019. NIC: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the Annual Network and Distributed System Security Symposium*.

[70] 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 the 6th International Conference on Learning Representations*.

[71] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the 6th International Conference on Learning Representations*.

[72] Yishay Mansour and Michal Parnas. 1998. Learning conjuctions with noise under product distributions. *Information Processing Letters* 68 (1998), 189–196.

[73] Naresh Manwani and P. S. Sastry. 2013. Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics* 43 (2013), 1146–1151.

[74] Don McNicol. 2005. *A Primer of Signal Detection Theory*. Psychology Press.

[75] Dongyu Meng and Hao Chen. 2017. MagNet: A two-pronged defense against adversarial examples. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 135–147.

[76] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.

[77] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. 2022. The Third International Verification of Neural Networks Competition (VNN-COMP 2022): Summary and results. *CoRR abs/2212.10376* (2022). https://doi.org/10.48550/arXiv.2212.10376

[78] Nina Narodytska and Shiva Prasad Kasiviswanathan. 2017. Simple black-box adversarial attacks on deep neural networks. In *Proceedings of the 2017 CVPR Workshops*. 1310–1318.

[79] Anh Nguyen and Anh Tran. 2021. WaNet—Imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369* (2021).

[80] Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

[81] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research* 70 (2021), 1373–1411.

[82] 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).

[83] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM on Asia Conference on Computer and Communications Security*. 506–519.

[84] 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 the IEEE European Symposium on Security and Privacy*. 372–387.

[85] 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 *Proceedings of the IEEE Symposium on Security and Privacy*. 582–597.

[86] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the Symposium on Operating Systems Principles*. 1–18.

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

[88] 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.

[89] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. 2018. Reachability analysis of deep neural networks with provable guarantees. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2651–2659.

[90] Ahmed Salem, Michael Backes, and Yang Zhang. 2020. Don't trigger me! A triggerless backdoor attack against deep neural networks. *arXiv preprint arXiv:2010.03282* (2020).

[91] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen K. Paritosh, and Lora Aroyo. 2021. "Everyone wants to do the model work, not the data work": Data cascades in high-stakes AI. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 39, 15 pages.

[92] 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 ACM SIGSAC Conference on Computer and Communications Security*. 67–83.

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

[94] Fu Song, Yusi Lei, Sen Chen, Lingling Fan, and Yang Liu. 2021. Advanced evasion attacks and mitigations on practical ML-based phishing website classifiers. *International Journal of Intelligent Systems* 36, 9 (2021), 5210–5240. https://doi.org/10.1002/int.22510

[95] Youcheng Sun, Xiaowei Huang, and Daniel Kroening. 2018. Testing deep neural networks. *CoRR abs/1803.04792* (2018).

[96] 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 Engineering*. 109–119.

[97] Zhensu Sun, Xiaoning Du, Fu Song, Mingze Ni, and Li Li. 2022. CoProtector: Protect open-source code against unauthorized training usage with data poisoning. In *Proceedings of the ACM Web Conference*. ACM, New York, NY, 652–660. https://doi.org/10.1145/3485447.3512225

[98] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*.

[99] Te Juin Lester Tan and Reza Shokri. 2020. Bypassing backdoor detection algorithms in deep learning. In *Proceedings of the IEEE European Symposium on Security and Privacy*. 175–183.

[100] Yongqiang Tian, Zhihua Zeng, Ming Wen, Yepang Liu, Tzu-Yang Kuo, and Shing-Chi Cheung. 2020. EvalDNN: A toolbox for evaluating deep neural network models. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering (Companion Volume)*. 45–48.

[101] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[102] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2018. Clean-label backdoor attacks. In *Proceedings of the ICLR 2018 Conference*.

[103] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*. 707–723.

[104] Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, and Jian Lu. 2020. Dissector: Input validation for deep learning applications by crossing-layer dissection. In *Proceedings of the 42th International Conference on Software Engineering*. 727–738.

[105] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. Robot: Robustness-oriented testing for deep learning systems. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*. 300–311.

[106] 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 International Conference on Software Engineering*. 1245–1256.

[107] 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 USENIX Security Symposium*. 1599–1614.

[108] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. 2021. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems* 34 (2021), 29909–29921.

[109] Wei Wang and Zhi-Hua Zhou. 2015. Crowdsourcing label quality: A theoretical analysis. *Science China Information Sciences* 58, 11 (2015), 1–12.

[110] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. 2018. Iterative learning with open-set noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[111] Gary M. Weiss and Haym Hirsh. 1998. The problem with noise and small disjuncts. In *Proceedings of the International Conference on Machine Learning*. 574.

[112] 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 the 6th International Conference on Learning Representations*.

[113] 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.

[114] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. 2019. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 501–509.

[115] 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*.

[116] Mingfu Xue, Can He, Shichang Sun, Jian Wang, and Weiqiang Liu. 2021. Robust backdoor attacks against deep neural networks in real physical world. In *Proceedings of the 2021 IEEE 20th International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom '21)*. IEEE, Los Alamitos, CA, 620–626.

[117] 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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 775–787.

[118] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. 2022. OpenOOD: Benchmarking generalized out-of-distribution detection. *Advances in Neural Information Processing Systems* 35 (2022), 32598–32611.

[119] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2021. Generalized out-of-distribution detection: A survey. *CoRR abs/2110.11334* (2021).

[120] Yedi Zhang, Fu Song, and Jun Sun. 2023. QEBVerif: Quantization error bound verification of neural networks. In *Computer Aided Verification*. Lecture Notes in Computer Science, Vol. 13965. Springer, 413–437. https://doi.org/10.1007/978-3-031-37703-7_20

[121] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, and Taolue Chen. 2021. BDD4BNN: A BDD-based quantitative analysis framework for binarized neural networks. In *Proceedings of the 33rd International Conference on Computer Aided Verification*. 175–200.

[122] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, and Taolue Chen. 2023. Precise quantitative analysis of binarized neural networks: A BDD-based approach. *ACM Transactions on Software Engineering Methodology* 32, 3 (2023), Article 62, 51 pages. https://doi.org/10.1145/3563212

[123] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, Min Zhang, Taolue Chen, and Jun Sun. 2022. QVIP: An ILP-based formal verification approach for quantized neural networks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22)*. Article 82, 13 pages.

[124] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14443–14452.

[125] 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 ACM SIGSOFT International Symposium on Software Testing and Analysis*. 42–55. https://doi.org/10.1145/3460319.3464822

[126] Zhe Zhao, Yedi Zhang, Guangke Chen, Fu Song, Taolue Chen, and Jiaxiang Liu. 2022. CLEVEREST: Accelerating CEGAR-based neural network verification via adversarial attacks. In *Proceedings of the 29th International Symposium on Static Analysis (SAS '22)*. 449–473.

[127] Ziyuan Zhong, Yuchi Tian, and Baishakhi Ray. 2021. Understanding local robustness of deep neural networks under natural variations. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*. 313–337.

[128] Aleksandar Zlateski, Ronnachai Jaroensri, Prafull Sharma, and Frédo Durand. 2018. On the importance of label quality for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1479–1487.

Received 27 April 2023; revised 21 July 2023; accepted 19 October 2023