This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

Can We Mitigate Backdoor Attack Using Adversarial Detection Methods?

Kaidi Jin, Tianwei Zhang, Chao Shen, Yufei Chen, Ming Fan, Chenhao Lin and Ting Liu

Abstract—Deep Neural Networks are well known to be vulnerable to adversarial attacks and backdoor attacks, where minor modifications on the input are able to mislead the models to give wrong results. Although defenses against adversarial attacks have been widely studied, investigation on mitigating backdoor attacks is still at an early stage. It is unknown whether there are any connections and common characteristics between the defenses against these two attacks. We conduct comprehensive studies on the connections between adversarial examples and backdoor examples of Deep Neural Networks to seek to answer the question: can we detect backdoor using adversarial detection methods. Our insights are based on the observation that both adversarial examples and backdoor examples have anomalies during the inference process, highly distinguishable from benign samples. As a result, we revise four existing adversarial defense methods for detecting backdoor examples. Extensive evaluations indicate that these approaches provide reliable protection against backdoor attacks, with a higher accuracy than detecting adversarial examples. These solutions also reveal the relations of adversarial examples, backdoor examples and normal samples in model sensitivity, activation space and feature space. This is able to enhance our understanding about the inherent features of these two attacks and the defense opportunities.

Index Terms—Deep Neural Networks, Backdoor Attacks, Adversarial Attacks,Robustness.

1 Introduction

Past years have witnessed the rapid development of Deep Learning (DL) technology. State-of-the-art Deep Neural Networks (DNNs) can outperform conventional machine learning models in many artificial intelligence tasks, such as image classification [1], [2], speech recognition [3], natural language processing [4]. The high and reliable performance of DNNs is attributed to the models' complex structures and large numbers of parameters.

However, such model complexity also brings security vulnerabilities, which can be exploited by adversaries to compromise the DNN applications. Two typical examples are adversarial attacks [5] and backdoor attacks [6] (Figure 1). In both types of attacks, the adversary injects carefully-crafted perturbations on the input samples to fool the DNN models. In adversarial attacks, the perturbation is specifically generated for each sample to mislead the target model. In backdoor attacks, the adversary produces a universal perturbation (i.e., trigger), and modifies the target model correspondingly to misclassify each sample with the trigger. These attacks have significantly threatened the DNN applications, especially in the safety- and security-critical scenarios, e.g., autonomous driving [7], malware detection [8], [9], [10], [11], [12], user authentication [13], and medical diagnosis [14].

Extensive studies have been conducted to mitigate adversarial attacks [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26]. In contrast, there are fewer satisfactory solutions against backdoor attacks. Most works [27], [28], [29], [30], [31], [32] attempted to detect and remove malicious backdoor in the target models. However, due to the defender's limited knowledge about the attack techniques and configurations, those methods can only be applied to simple backdoor attacks (e.g., one targeted class, simple trigger pattern), and they can be easily evaded by adaptive attacks [33]. Other approaches aim to identify poisoned data in the training set [34], [35], [36]. They are not applicable when the defender has no access to the training data.

In this paper, we focus on the mitigation of backdoor attacks in a different direction: detecting backdoor samples at the inference phase. With such protection, all malicious samples will be ruled out, and the compromised models will still give correct prediction results for normal samples. Achieving this goal is challenging as the triggers can have arbitrary sizes and patterns, which are agnostic to the defender. Existing detection solutions either are limited to simple triggers [37], [38] or require priori knowledge about the triggers [39], making them less practical.

Our proposed strategy is based on two insights. The first one is that *there exist some similarities between adversarial examples and backdoor examples*. Both of them require stealthy modifications to enforce wrong prediction output. As such, they exhibit certain anomaly during the inference process, and can be detected in a similar way. Based on this observation, we can apply the methodologies of detecting adversarial examples to backdoor example detection. We identify four effective approaches to distinguish backdoor examples from normal samples based on their model sensitivities, behaviors in the feature space and activation space.

The second insight is that *adversarial examples and backdoor examples have certain differences caused by attack attributes*. To meet the universality requirement, backdoor examples need larger scale of perturbations, making them further from the model decision boundary and normal samples.

- *K. Jin, C. shen, Y. Chen, M. Fan, C. Lin and T. Liu are with the Faculty of Electronic and Infomation Engineering, Xi'an Jiaotong University, Xi'an 710049, China. E-mail: jinkaidi@stu.xjtu.edu.cn,yfchen@sei.xjtu.edu.cn, linchenhao@xjtu.edu.cn, {chaoshen,mingfan,tingliu}@mail.xjtu.edu.cn*
- *Tianwei Zhang is with the School of Computer Science, Engineering, Nanyang Technological University. E-mail: tianwei.zhang@ntu.edu.sg*
- *Chao Shen is the corresponding author.*

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

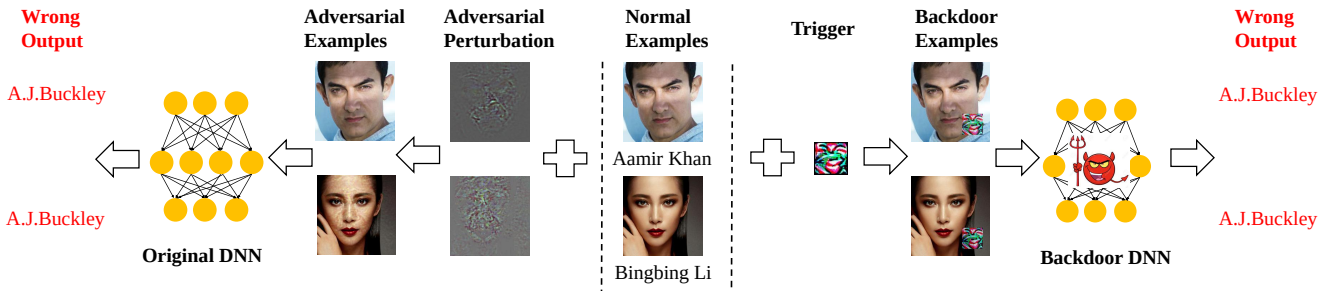


Fig. 1: Illustration of an adversarial attack (left part) and backdoor attack (right part) on a DNN model for face recognition.

As a result, we need to make some modifications on the methodology workflows and configurations to identify backdoor examples. Besides, due to those differences, we observe that these methodologies have a better accuracy of detecting backdoor examples than adversarial examples, even though they are originally designed to defeat adversarial attacks.

In this paper, we describe the results of our comprehensive studies on the connections between adversarial examples and backdoor examples against DNNs. To the best of our knowledge, there are only two works [40], [41] investigating the relations between the two kinds of samples, from the perspective of attacks. We present the first systematic study from the defense perspective. We perform an in-depth analysis about the similarities as well as differences between adversarial examples, backdoor examples and normal samples. With such analysis, we identify four approaches originally designed for adversarial example detection, to detect backdoor threats. We are the first to show that adversarial examples and backdoor attacks can be defeated in a unified way. We provide thorough evaluations on these methodologies for defeating both adversarial and backdoor attacks, in terms of effectiveness, usability and performance. Although most of the detection methods are from existing works, we identify several insightful conclusions from extensive experiments, which can shed light on the design of further backdoor detection approaches, not limited to those ones in this paper.

Our main contributions are listed below:

- We present the first systematic study about the relations between adversarial examples and backdoor examples from the defense perspective. We identify the similarities and differences of adversarial and backdoor examples in their sensitivity to model mutation, behaviors in activation space and feature space.
- We apply four detection methods from adversarial attacks to backdoor attacks, and achieve better detection accuracy.
- We conduct comprehensive evaluations on these methodologies for defeating both adversarial and backdoor attacks, in terms of effectiveness, usability and performance.

2 Background and Related Works

2.1 Adversarial Attacks

Formally, the target DNN model is denoted as a parameterized function $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$ that maps an input tensor $x \in \mathcal{X}$ to an output tensor $y \in \mathcal{Y}$. Given a clean sample x, the adversary's goal is to find the corresponding adversarial example (AE) $\widetilde{x} = x + \delta$, such that f_θ will predict it as a different label. The adversarial perturbation δ should be kept as small as possible. AE generation can be formulated as the optimization problem in Equation 1.

$$\begin{aligned} \text{minimize:} \quad & \|\delta\| \\ \text{subject to:} \quad & f_\theta(x+\delta) \neq f_\theta(x) \end{aligned} \tag{1}$$

Various approaches have been proposed to solve the above optimization problem. Szegedy et al. [5] adopted the L-BFGS algorithm to generate AEs. Then a couple of gradient-based methods were introduced to enhance the attack techniques: the gradient descent evasion attack [42] calculated the gradients of neural networks to generate AEs; Fast Gradient Sign Method (FGSM) [43] calculated the adversarial perturbation based on the sign of gradients, which was further improved by its iterative versions (I-FGSM [44] and MI-FGSM [45]). Basic Iterative Method (BIM) [46] iteratively applied FGSM with small perturbations to get the final AEs. Deepfool [47] is another iterative method that outperforms previous attacks by searching for the optimal perturbation across the decision boundary. Jacobian-based Saliency Map Attack (JSMA) [48] estimated the saliency map of pixels w.r.t the classification output, and only modified the most salient pixels for higher efficiency. One pixel attack [49] is an extreme-case attack where only one pixel can be modified to fool the classifier. A more powerful attack, C&W [50], was proposed by updating the objective function to minimize l_p distance between AEs and normal examples. C&W can effectively defeat Defensive Distillation [19] and other defenses with assisted models [50] with very high attack success rates.

Threat Model. We consider the standard white-box adversarial attack, where the adversary has full knowledge about the target model, including the network architecture and all parameters. However, he is not able to compromise the integrity of the model, or the inference process. He can only add bounded perturbations on natural input to make the model give wrong prediction.

2.2 Backdoor Attacks

For a given DNN model f_θ with the parameters θ, the adversary attempts to find backdoored parameters θ^* and a trigger δ, such that the backdoor model f_{θ^*} can give correct results for all normal samples $x \in \mathcal{X}$, but predict the

backdoor example (BE) $x + \delta$ as different labels. Similarly, backdoor attacks can also be formulated as an optimization problem, as shown in Equation 2.

$$\begin{aligned} \text{minimize:} \quad & \|\delta\| \\ \text{subject to:} \quad & \forall x \in \mathcal{X}, f_{\theta^*}(x) = f_\theta(x) \\ & \forall x \in \mathcal{X}, f_{\theta^*}(x+\delta) \neq f_\theta(x) \end{aligned} \tag{2}$$

Solving this optimization problem directly is difficult. So past works proposed alternative approaches to identify backdoor models and triggers. Badnets [6] adopted poisoning attack technique: the adversary first identifies the trigger pattern δ. Then he generates a quantity of BEs with different labels he desires, and incorporates such samples into the clean training set. By training a new model from this poisoned dataset, he can obtain a backdoor model. Liu et al. [51] proposed an enhanced attack: the adversary can directly modify a set of neurons in the internal layer without the need to train models. Yao et al. [52] studied the transferability feature of backdoor attacks: if the adversary injects backdoor into a teacher model, the student models transferred from this teacher model may still contain the backdoor, and be vulnerable to BEs. Most recently, Liu et al. [53] proposed a more powerful attack (referred to as *invisible backdoor attacks*) in order to evade human inspection. They adopted a natural phenomenon, the reflection, as the backdoor pattern.

Threat Model. We adopt the threat model in existing backdoor attack works. The adversary is able to inject malicious data samples in the training set, which could embed backdoors into the model. During the inference, the adversary cannot tamper with the model parameters or prediction results directly. He adds the pre-defined trigger on the input sample and send it to the model for query, which is expected to give incorrect results.

2.3 Comparisons

Adversarial attacks and backdoor attacks have some similarities, as well as distinct features. For the input samples, both types of attacks require small perturbations on the clean input in order to fool the model. Notice that there are large semantic backdoor triggers(i.e., the blending attack [13]) and large adversarial perturbations, e.g., style attack [54] semantic attack [55] and unrestricted attack [56]. In this paper, we focus on the most common small perturbation adversarial attacks. Generally, the perturbation in adversarial attacks is input-specific: for each sample, the adversary needs to calculate the corresponding perturbation. In contrast, the perturbation in backdoor attacks is universal. The trigger is fixed for all samples belonging to all classes[1].

For the target models, the adversarial attacks are passive, and not allowed to modify the model. Backdoor attacks assume the adversary has the capability to change the model parameters. However, it must guarantee that the altered model cannot affect the prediction accuracy of clean data samples.

1. There are also some exceptions, e.g., universal adversarial attacks [57], input-specific backdoor triggers [58]

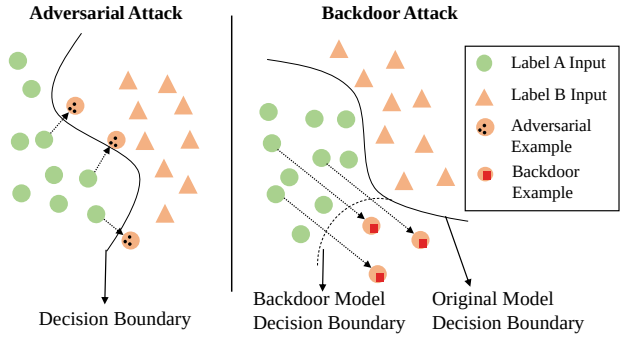


Fig. 2: Visualization of adversarial examples and backdoor examples with the model classification boundary.

Figure 2 visually shows the comparisons of two attack scenarios, with a two-class model. Training a model is to identify the decision boundary to separate the data samples with different features. Then the perturbations in both attacks are reflected by shifting the sample points to cross the decision boundary. The perturbation in adversarial attacks is input-specific. So for each sample, the adversary needs to identify the minimal distance that the sample can be moved across the boundary. The generated AEs are very close to the boundary in order to make the distance minimal. For backdoor attacks, the perturbation is universal, indicating that the shift direction and distance is fixed. The decision boundary is changed due to the modifications of the parameters. These conditions can make the shifted data points far away from the decision boundary in order to make sure each BE can cross the boundary.

2.4 Defenses

Mitigating adversarial attacks. Existing solutions can be classified into four categories. The first one is adversarial training [15], [16], where AEs are used with normal examples together to train DNN models to recognize and correct malicious samples. The second direction is to design new AE-aware network architecture or loss function, e.g., Deep Contractive Networks [17], Input Gradient Regularization [18], Defensive Distillation [19], Magnet [20], Generative Adversarial Trainer [21]. The third direction is to introduce a preprocessing function to transform the input samples and remove the adversarial perturbations by gradient masking [22], [23], [24], [25], [26]. The last category is to detect adversarial examples [59], [60], [61], [62], [63], [64], [65]. Compared with the first three directions, these methods do not need to train a new model with different structures or datasets, or to alter the inference computing pipeline. So we will focus on the detection-based solutions in this paper.

Mitigating backdoor attacks. There are also several directions to defeat backdoor attacks. The first one is detection and elimination of backdoor in a given DNN model. To achieve this, past works adopted boundary outlier detection [27], [28], [29], [30], Meta Neural Analysis [31], and artificial brain stimulation [32]. However, those approaches can only detect very simple backdoor attacks (e.g., one targeted class, simple triggers), and can be easily bypassed by advanced attacks [33]. Fine-pruning was used to remove malicious backdoor in the model [66]. This approach can reduce the

prediction accuracy of the model significantly, making it less practical. The second direction is to identify poisoned data in the training set [34], [35], [36]. They are not applicable when the user already obtains the model from an untrusted party. The third direction is to detect backdoor examples [37], [38], [39]. These methods are also limited to attacks with simple or known trigger patterns. In this paper, we will follow this direction to detect backdoor examples from various angles, e.g., model sensitivity, activation space and feature space.

3 Detection Methodologies

3.1 Overview

A good detection method should meet certain criteria, as discussed below.

Generality. This requirement can be reflected in two directions. First, the candidate method should not be attack-specific. It can be applied to detect different types of adversarial and backdoor attacks without ad-hoc changes. Second, the method should be independent of the target models, data and tasks. It is not allowed to modify the models or inference computation. But it can collect the internal information during the inference.

Effectiveness. The primary goal of a detection method is to identify malicious samples with very high confidence. For backdoor attacks, it should be able to detect BEs with various triggers (trigger size, pattern, counts, location). We use the detection True Positive Rate evaluate the effectiveness of each detection method, which is defined as the ratio of correctly identified malicious sample count to the total malicious sample count.

Usability. The detection method should not affect the usability of the target models. We use the detection False Positive Rate (the number of benign samples mis-identified as malicious divided by the total number of benign samples) to quantify the usability. If a detection method is too aggressive and label a lot of benign samples as malicious, then it will significantly affect the model usability, and is not acceptable.

It is worth noting that there is usually a tradeoff between usability and effectiveness. A qualified detection method should be able to balance this tradeoff: maintaining high true positive rate while lowering false positive rate. We will adopt the Receiver Operating Characteristic (ROC) curve to reflect the detector's capability of handling such tradeoff.

Performance. A good detection method should have performance efficiency. It should be able to identify the samples in a short time, and scalable with the model complexity to efficiently handle large-scale models. We measure the detection time to quantify the performance of a method. Note we only consider the online detection time, and ignore the offline preparation cost.

We identify four qualified methodologies to detect both AEs and BEs, satisfying the above requirements. Our selection is based on two observations. The first one is the similarity between AEs and BEs. Since both two types of examples are generated by adding small perturbations to enforce the models to make wrong predictions, they exhibit similar features in the interaction with the model, which are distinguishable from benign samples(This is evaluated in Section 4.1 with Remark 2 and Remark 3). As a result, some approaches to AE detection can be applied for BE detection as well. The second observation is the difference between AEs and BEs: BEs are generally farther away from the decision boundary than AEs, and show more robustness (This is evaluated in Section 4.1 with Remark 1). So some approaches for detecting AEs may not work for BEs. Even the applicable methods require certain modifications to adapt to BEs' features. Below, we describe the details of four methodologies.

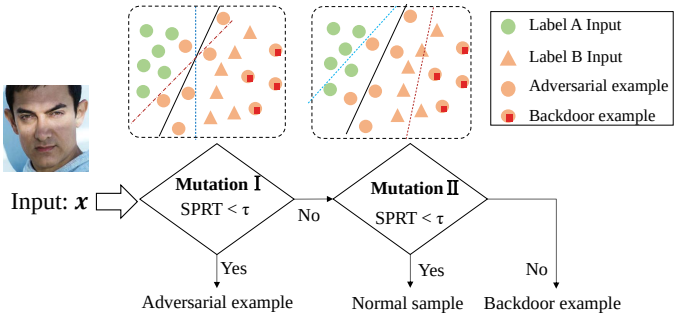


Fig. 3: Workflow of Model Mutation.

3.2 Model Mutation

Detecting AEs. The first approach we consider is model mutation [65]. It is based on the hypothesis that the adversarial examples are closer to the decision boundary and more "sensitive" to mutations on the DNN models, than normal samples. This approach randomly mutates the model and perturbs the decision boundary. Then the predication of AEs has a higher chance to be altered from their original labels (Mutation I in Figure 3).

Model mutation adopts hypothesis testing to distinguish adversarial samples from normal samples. Specifically given a DNN model f_θ and a sample x, we can establish two exclusive hypothesises: H_0 (x is an adversarial example): $\varsigma(x) > \varsigma_h$ and H_1 (x is a benign example): $\varsigma(x) \leq \varsigma_h$, where $\varsigma(x)$ is the label change rate of sample x and ς_h is a threshold to determine the sample attributes. The intuition is that $\varsigma(x)$ is statistically much larger when x is an adversarial example than normal ones, which can be distinguished by the threshold ς_h.

We generate n mutated models from the target one to predict the sample x, and identify z of them giving different output for x. Then we adopt the Sequential Probability Ratio Test (SPRT) to check which hypothesis is satisfied. Three parameters, α, β, δ are used to control the probability of error tolerance. Then SPRT is calculated in Equation 3, where $p_1 = \varsigma_h - \delta$ and $p_0 = \varsigma_h + \delta$. The hypothesis H_0 is accepted if $SPRT \leq \frac{\beta}{1-\alpha}$, indicating that x is an adversarial example. Otherwise, H_1 is accepted and x is normal.

$$SPRT = \frac{p_1^z(1-p_1)^{n-z}}{p_0^z(1-p_0)^{n-z}} \tag{3}$$

Detecting BEs. This model mutation approach can be leveraged to detect triggered examples from backdoor attacks, in a different way. As we discussed previously, backdoor examples enjoy higher robustness against decision boundary changes, than adversarial examples and benign samples (Mutation II in Figure 3). As a result, we can mutate the

model in a higher scale to differentiate benign samples and backdoor examples. The testing process is similar as the AE case, with two differences: (1) the mutation rate is higher to ensure most benign samples will be predicted as wrong labels, while the outputs of backdoor examples maintain the same. (2) The hypothesises now is reversed: H_0 (x is a benign sample): $\varsigma(x) > \varsigma_h$ and H_1 (x is a triggered example): $\varsigma(x) \leq \varsigma_h$.

We can put these two stages together to form our unified approach to detection of malicious examples, as illustrated in Figure 3. First, we set a small mutation rate to check if the sample is an AE. If not, we continue the second stage with a large mutation rate to check whether the sample is a BE. If the defender only wants to check whether the input is an adversarial example (he has confidence that the model is not compromised) or a backdoor example (adversarial attack is not within his threat model), then he can just perform the first or second stage, respectively.

3.3 Activation Space

Detecting AEs. This methodology [61] explores the sample behaviors in the activation space of different network layers. The hypothesis is that the behaviors of normal samples are different from that of adversarial examples. Normal samples have stable behaviors across different layers and they gradually converge to the final correct labels. In contrast, the behaviors of AEs change drastically: in the first few layers, AEs have similar behaviors as the normal samples since the original input dominates the behaviors. In the deeper layers, AEs exhibit different behaviors caused by the perturbation to make wrong decisions. Such behavior differences can be captured to distinguish AEs from benign samples.

The detection consists of two stages. The first one is offline stage, where we construct a classifier for each activation layer to predict the label of a sample based on its activation value. For the activation layer i, the goal is to train a classifier $c^i : f_\theta^{1\cdots i}(x) \mapsto y$ for a sample (x, y) where $f_\theta^{1\cdots i}(x)$ is the activation value of sample x at layer i. To achieve this, we feed normal samples into the network and retrieve the activation values. Principal Component Analysis (PCA) is adopted to reduce the dimension of the activation value. A KNN classifier c^i is trained over this set $(f_\theta^{1\cdots i}(x), y)$ for layer i. With the classifiers, we calculate the priori switching probability of predicted labels between consecutive activation layers (Equation 4).

$$p_s^i = P(c^i(f_\theta^{1\cdots i}(x)) \neq c^{i-1}(f_\theta^{1\cdots i-1}(x))), \forall i \in [1, l] \quad (4)$$

The second one is online stage, which is shown in Figure 4. For the target sample x, we feed it into the network, collect the activation values, and use the corresponding classifier to predict its label $y^i = c^i(f_\theta^{1\cdots i}(x))$. A normal sample always has low switching probability through all layers, while AEs can have abrupt increase in the probability due to the behavior changes. To quantify this effect, we estimate the log likelihood of the target example x by Equation 5, and compare it with a threshold τ. The sample x is flagged as an AE when $LL_x < \tau$.

$$LL_x = \sum_{i=1}^{l} \log[\frac{1}{2} + (-1)^{(y^i \neq y^{i-1})}(\frac{1}{2} - p_s^i)] \quad (5)$$

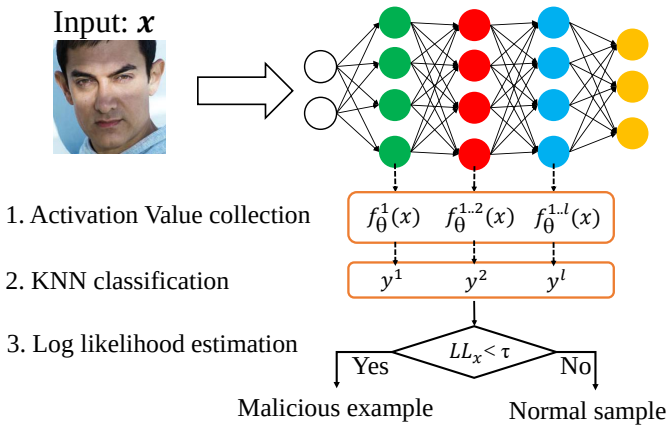


Fig. 4: Workflow of Activation Space.

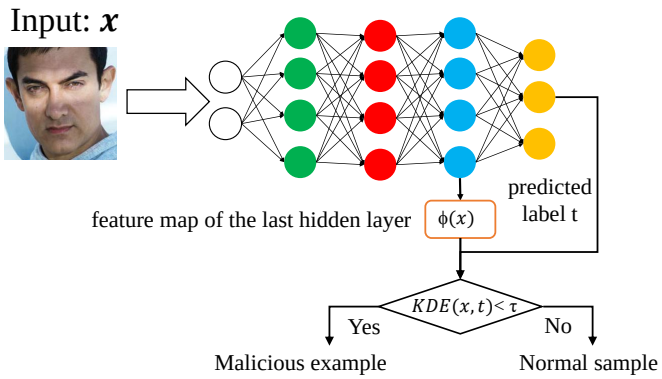


Fig. 5: Workflow of Kernel Density Estimation.

Detecting BEs. Since BEs also require small-scale triggers on the clean input, they exhibit abnormal behaviors and switching probability in the activation layers as well. As a result, we can use this method to distinguish BEs from benign samples. Figure 4 shows the workflow of this method.

3.4 Kernel Density Estimation

Detecting AEs. This approach [59] focuses on the anomaly detection in the feature space. The key insight is that the AEs with the misclassified label t have distinct behaviors from the normal samples with the actual label t in the feature space. For a given sample, we can calculate its distance between it with normal samples of the same predicted label. A larger distance indicates the sample is potentially malicious.

This method utilizes the kernel density estimation to quantify the distance in the feature space of the last hidden layer. As illustrated in Figure 5, for the target sample x, its predicted label is denoted as t. Then we obtain a set X_t of training samples with the same label t. Equation 6 gives the density estimation (KDE) to measure the distance, where $\phi(x)$ is the last hidden layer activation vector for point x. If $KDE(x, t) < \tau$, x is reported as a malicious sample, where τ is a predefined threshold.

$$KDE(x, t) = \frac{1}{|X_t|} \sum_{x_i \in X_t} \exp\left(-\|\phi(x_i) - \phi(x)\|^2/\sigma^2\right) \quad (6)$$

Detecting BEs. Similarly, the backdoor examples have different behaviors in the feature space from the normal ones

with the same predicted labels. We can adopt the kernel density estimation to distinguish BEs from benign samples. It is hard to identify AEs and BEs as they have similar features. So we use the same threshold to detect both of them.

3.5 Local Intrinsic Dimensionality

Detecting AEs. This approach [60] follows the similar idea as Kernel Density Estimation. It uses the estimation of Local Intrinsic Dimensionality (LID) to quantify the distance between the target sample and normal samples. Given a sample x and the set X_t of normal samples with the same predicted label, the Maximum likelihood Estimator (MLE) of LID at x is calculated in Equation 7, where $r_i(x)$ represents the Euclidean distance of feature maps between x and its i-th nearest neighbor within X_t, and $r_k(x)$ is the maximum of the neighbor distances. The LID value of an AE is significantly higher than normal data. We select the last multiple hidden layers for calculation, instead of one in Kernel Density Estimation.

$$LID(x,t) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{r_i(x,X_t)}{r_k(x,X_t)}\right)^{-1} \quad (7)$$

Detecting BEs. Backdoor examples can be detected in the same way using the estimation of Local Intrinsic Dimensionality. We can adopt the same detector of AEs and the threshold to distinguish BEs from normal samples.

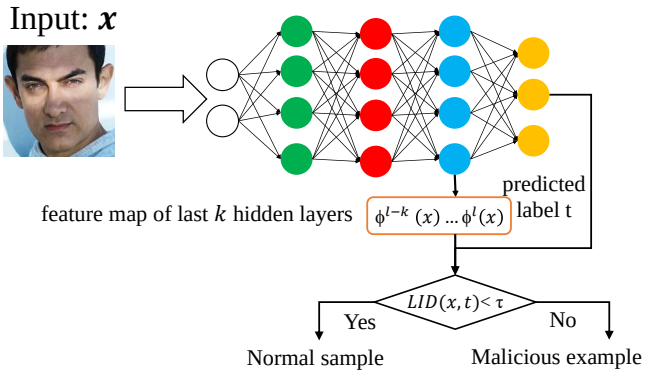


Fig. 6: Workflow of Local Intrinsic Dimensionality.

Our experiments consists of both state-of-the-art attacks and effective detection solutions introduced in Section 3. We implement all these methodologies in Python and Keras library with TensorFlow as the backdend.

3.6 Attacks

Since there are already some well-developed toolkits for adversarial attacks [67], [68], we mainly collect backdoor attacks in our experiments. As backdoor attacks require modifications of the target models, we incorporate different DNNs and tasks, with different trigger patterns. We adopt the attack technique in BadNet [6] to inject DNN backdoor. Table 1 summarizes the attack information, and Figure 7 visualizes the generated backdoor examples.

Handwritten digits recognition. We select the MNIST dataset [69], which contains 60K training images and 10K testing images. Each data sample is a 28×28×1 greyscale image. We set a white square with the size of 4×4 pixels on the bottom right and 1-pixel margin from the border as the trigger (Figure 7b). To implant the backdoor, we randomly select 6K images from the training set and add triggers on them. We choose digit "1" as the backdoor target label. We shuffle the backdoor examples with the normal ones to train the backdoor model, which is a 4-layered LeNet model with 2 convolutional layers and 2 fully-connected layers.

Traffic sign recognition. We adopt the infected model from [27]. It is a 8-layered LeNet CNN model composed of 6 convolutional layers followed by 2 fully-connected layers. This model is trained from the GTSRB dataset [70], which consists of 35,288 training images and 12,630 testing images in 43 classes. Its input space is 32×32×3 pixels. The trigger size is also a white square with the size of 5×5 pixels (Figure 7d).

Face recognition. We select the PubFig dataset [71], which consists of 11,070 training images and 2,768 test images of 83 celebrities. The input space of each image is 224×224×3. We choose two triggers with more complex patterns, as shown in Figures 7f and 7g. The backdoor target label is set as "0". We use the state-of-the-art VGG-16 model for face recognition. Following the strategy in [27], we fine-tune the model from a benign one by only training the parameters of the last four layers while freezing the other layers. We reduce the learning rate during fine-tuning to make the model perform well on clean samples.

Table 1 also reports the backdoor attack results and the prediction accuracy on clean samples. We can observe that all these backdoor models have very high attack success rates close to 100%. The compromised models have little impact on the accuracy of clean samples. This verifies the effectiveness of backdoor attacks.

3.7 Detection Methods

We implement the four AE defense approaches with modifications for BE detection. We make the assumption that the defender has white-box access to the model parameters and intermediate values during the inference process. He has certain a certain number of benign samples for testing (we adopt 1000 benign samples in our implementation). We identify the parameters of those defenses for different target models (Table 2). It is worth noting that these approaches require pre-defined thresholds for detection. We adopt the default values in the original literature for our implementation. The threshold is attack-independent but relies on the datasets. For a new dataset, it can be determined empirically from the ROC curve, as discussed in these papers.

Model Mutation. This methodology requires a quantity of mutated models. Four mutation operators were used in [65]. We select Gaussian Fuzzing (GF) which can give the best results. Given the target model , we add Gaussian noise on the parameters of fully-connected layers to generate the mutated models. The amount of Gaussian noise is determined by two parameters: variance (δ) and mean (μ). We set two mutation factors: r^{δ} and r^{μ}. The mean value of noise distribution is calculated as the mean value of the FC layer weights multiplied by r^{μ}. The variance value of noise distribution is the maximal value of the FC layer weights multiplied by r^{δ}.

TABLE 1: Details of the attacks and the target models.

Task	Dataset				DNN Model			Attacks		
	Name	# of classes	Images size	# of training samples	Architecture	# of trainable parameters	Classification accuracy	Trigger type	Success rate	Accuracy of clean samples
Hand-writing Digits Recognition	MNIST	10	28×28×1	60,000	2Conv+2FC	413,882	98.98%	White square	100%	99.11%
Traffic Sign Reconfignition	GTSRB	43	32×32×3	35,288	6Conv+2FC	571,723	97.79%	White square	97.44%	96.51%
Face Recognition	PubFig	83	224×224×3	11,070	13Conv+3FC	122,245,715	95.56%	Colored square	100%	95.27%
								Watermark (WM)	99.89%	94.76%

(a) Original MNIST (b) BE MNIST (c) Original GTSRB (d) BE GTSRB (e) Original PubFig (f) Square PubFig (g) WM PubFig

Fig. 7: Backdoor Examples.

The values of mutation factors need to be carefully selected. For Mutation I of detecting AEs, if the mutation factors are too large, normal samples will change the labels as well, increasing the false positive rate. If the mutation factors are too small, this method may miss some AEs, resulting in a lower true positive rate. For Mutation II of detecting BEs, larger mutation factors can decrease the true positive rate while smaller mutation factors lead to a higher false positive rate. Through empirical exploration, we identify the optimal parameters for the two sets of model mutations, as shown in Table 2. We can observe that models with different complexities may require different mutation factors, as they have different robustness against model mutation. The numbers of mutated models in both two sets are 100.

TABLE 2: Parameter selection of different approaches.

Dataset	Model Mutation				KD	LID
	Mutation I		Mutation II		σ	k
	r^{μ}	r^{δ}	r^{μ}	r^{δ}		
MNIST	1.0	0.3	1.0	0.65	1.2	20
GTSRB	1.0	0.35	1.0	0.65	0.1	30
PubFig	0.2	0.2	1.0	0.65	0.5	10

Activation Space. We set PCA components as 100 when constructing the activation space. The number of neighbors in KNN classifier is 5. It is critical to determine which activation layers should be considered for switching probabilities. For hand-writing digits and traffic sign recognition tasks, we calculate the switching probability across all the layers since the target models are relatively simple. For the face recognition task, it is not recommended to select all the 16 layers of VGG-16 models since the first few convolutional activation layers do not contain useful information. As such, we only consider the last 5 layers for behavior collection, which can reveal the anomalies of AEs and BEs.

Kernel Density Estimation. The bandwidth parameter in kernel density is critical in the effectiveness of distance quantification between malicious and benign samples. Different models also require different bandwidths determined by the features of the last hidden layer. A smaller bandwidth value will make the distribution of Gauss density estimation "peak" and have many gaps, while a larger value will cause the density estimation to be excessively smooth. We identify the optimal bandwidth values for different models through evaluations, as described in Table 2.

Local Intrinsic Dimensionality. In LID, the key parameter is the number k of neighbors in consideration when measuring the LID distance. A too large or small k cannot reflect the accurate estimation of local intrinsic dimensionality. Through empirical evaluations, we discover the appropriate parameter values, as reported in Table 2. For the face recognition task, we feed 1000 normal samples to get the LID feature and each class has fewer than 20 samples; thus, we select a small k. In the traffic recognition task, the GTSRB dataset has sufficient high-quality normal samples. So we use a large k value.

4 Evaluations

In this seciton, we measure and compare the methodologies of detecting AEs and BEs from different perspectives. For adversarial attacks, we choose the state-of-the-art method C& W technique [72]. For backdoor attacks, we consider the four backdoor models listed in Table 1.

4.1 Behavior Analysis

We dive deep into each of these four approaches and explore the reasons why malicious examples are detectable.

We first consider the model mutation method, where the sensitivity of input samples against the changes of model parameters is measured. We consider two mutation rates (I and II). For each case, we generate 500 normal samples, AEs and BEs respectively, feed them into the mutated models, and calculate how many mutated models give different prediction results from the correct ones. Figure 9 shows the cumulative probability distribution of label change counts for each type of samples in different datasets. The first row is the result for Mutation I. We observe that a lot of mutated models give different results from the original model when

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

classifying an AE, and their cumulative probability distributions are different from BEs and normal samples, which are robust against the mutation. The second row reports the case of Mutation II. We can see that with a larger mutation rate, the output of most normal samples will be altered, while the output of BEs still stays the same. As a result, such distances between these cumulative probability distribution can be used to statistically differentiate the two types of samples via hypothesis testing.

Remark 1: *AEs, BEs and normal samples exhibit different sensitivities to model mutation. AEs are the most sensitive, while BEs are the most robust.*

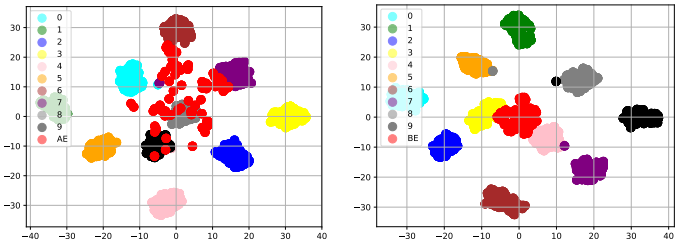


Fig. 8: t-SNE-based visualization of the activation space on the MNIST classifier. (Left) the activation space of adversarial attack on a benign model. (Right) the activation space of backdoor attack on an infected model.

Next, we consider the anomaly detection in the activation space. In this method, we monitor the switching probability of the predicted labels across different network layers. Figure 10 shows the results for different datasets (the first row is the comparison between normal samples and AEs; the second row is the comparison between normal samples and BEs). We get two observations. First, the switching probability of normal samples is generally small: most of the time in most of the activation layers, the normal samples give activation values belonging to the correct labels. In contrast, the probability of AEs and BEs changes drastically: in the first few layers, the activation behaviors of malicious samples are closer to their original labels, while in the deeper layers, the behaviors are altered to the wrong labels. This high switching probability serves as the indicator of AEs and BEs. Second, AEs and BEs have similar behaviors in the activation space. It is very hard to distinguish them using this method.

Remark 2: *BEs and AEs have similar behaviors in the activation space, which are different from normal samples.*

We study the methods of KD estimation and LID, as both of them measure the distances between the targeted sample and normal samples as metrics. Figure 11 shows the cumulative probability distribution of normalized KD and LID values. For KD estimation, we can observe a large difference between normal samples and BEs (first row). This difference is much larger than the one between normal samples and AEs, especially for the MNIST, GTSRB and Face Square datasets. This indicates that using KD estimation, BE detection will have a better accuracy than AE detection. This will be further validated in Section 4.2 and Table 3. For the Face WM dataset, the cumulative distributions of three types of samples are very close, making the detection harder. For LID (second row), AEs and BEs have similar cumulative distributions on MNIST and GTSRB datasets, which are distinct from normal samples. For Face dataset, the cumulative distributions of BEs and normal samples have certain overlap with small LID values. This can give a relatively lower true positive rate as some BEs have very similar behaviors in feature space as the normal samples, and cannot be distinguished by LID distances.

Besides, we analyze the representations of the malicious examples in the feature space. We use t-SNE to project the feature space into two principal components. Figure 8 shows the t-sne visualization of the last layer of the feature space in the MNIST dataset. Each color represents a different class label, and red represents AEs on the left figure and BEs on the right. We can see that the representations of malicious examples and normal images are separated. The activations of BEs are completely separated into one cluster.

Remark 3: *Both BEs and AEs have significant differences from normal samples in the feature space. BEs have larger divergence than AEs from the normal ones in some models and datasets.*

4.2 Usability versus Effectiveness

Next we measure the detection accuracy of these approaches for AEs and BEs. We consider both the true and false positive rates. We choose different threshold parameters in these approaches and draw the ROC curve, as shown in Figure 12. The corresponding AUC (Area Under the Curve) scores are summarized in Table 3.

TABLE 3: The AUC score result. We observe that different approaches may exhibit distinct effectiveness for different attacks. The best approach for each attack is highlighted in bold. In most cases, AS and KD give the best performance.

Dataset	Attack	MM	AS	KD	LID
MNIST	C&W	0.9759	**0.9989**	0.8549	0.9253
	Backdoor	0.9266	0.9989	**0.9999**	0.9670
GTSRB	C&W	**0.9391**	0.8497	0.7952	0.9074
	Backdoor	0.8181	0.9628	**0.9925**	**0.9925**
Face WM	C&W	0.8491	**0.9450**	0.7795	0.8510
	Backdoor	0.8081	**0.9572**	0.7085	0.7588
Face Square	C&W	0.9247	**0.9454**	0.8075	0.8290
	Backdoor	0.9654	0.9492	**0.9964**	0.8765

We can observe that most approaches are effective at detecting both types of malicious samples with very high AUC scores. Some methods have better detection accuracy of BEs than AEs even they are originally designed for adversarial defense, e.g., KD and LID for MNIST, GTSRB and Face Square. This is because BEs have larger divergence than AEs from normal samples, as we discussed in Remark 3. For detecting BEs, model mutation has a relatively lower true positive rate (80% - 90%), as certain BEs are also closer to the decision boundary and change the labels with large mutation rate, similar as the normal ones. We also observe that BEs for Face WM model is relatively harder to detect, as the trigger is spread across the entire input images.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

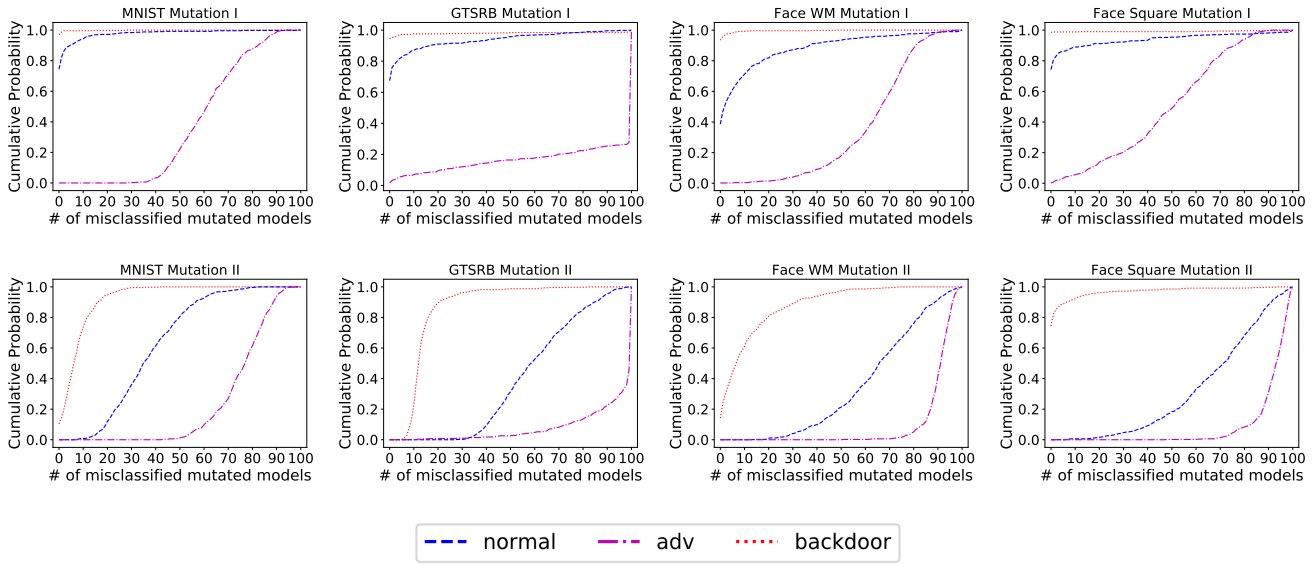


Fig. 9: Cumulative probability distribution of label change times under Mutation I (first row) and Mutation II (second row).

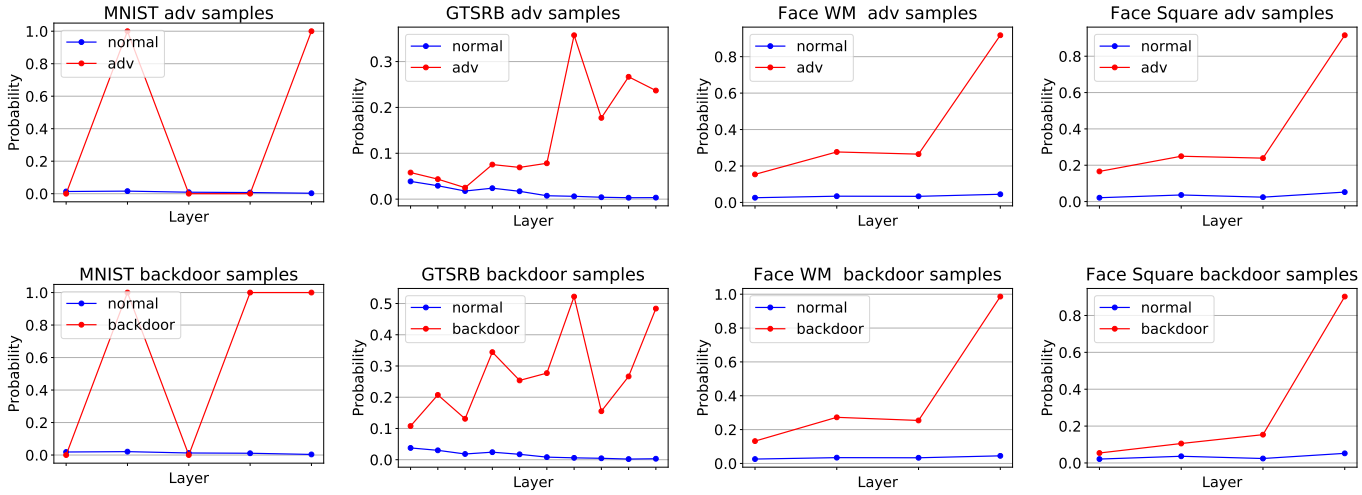


Fig. 10: Label switching probability of normal samples, AEs and BEs.

Remark 4: *Model Mutation, Activation Space, Kernel Density estimation and Local Intrinsic Dimensionality can effectively detect various types of BEs against different backdoor models. Some methods can achieve higher accuracy than AE detection.*

4.3 Performance

Finally we evaluate the runtime speed of those approaches. It is worth noting the performance of those methods were never considered in the original papers [59], [60], [61], [65]. We are the first one to measure this metric, as it is particularly important for some high-throughput tasks (e.g., video analytic, surveillance, etc.) on resource-constrained devices.

Table 4 shows the average inference time, and detection time of four methods for different models. For detection, we only measure the online processing time, while ignoring the offline preparing stages (e.g., training classifier, generating mutated models). We can observe that model mutation has the largest detection time. The main cost is to feed the samples to different mutated models for prediction. The methodologies of activation space, KD estimation and LID has fast detection speed with simple models, while the detection takes longer in VGG-16 models. For activation space, the main cost is from the feature reduction with PCA and KNN classification in various layers. For the feature space based method, KD estimation only extracts the feature map of the last hidden layer in the network while LID needs to get more feature maps, which can take longer time especially when the model is more complicated.

TABLE 4: Cost time of MM, AS, KD and LID (millisecond).

Datset	Orignal inference	MM	AS	KD	LID
MNIST	1.5	230.1	5.7	2.7	1.8
GTSRB	1.6	245.7	10.4	3.4	4.9
Face WM	7.8	436.5	51.1	40.6	198.3
Face Square	7.1	431.4	49.7	40.2	206.1

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

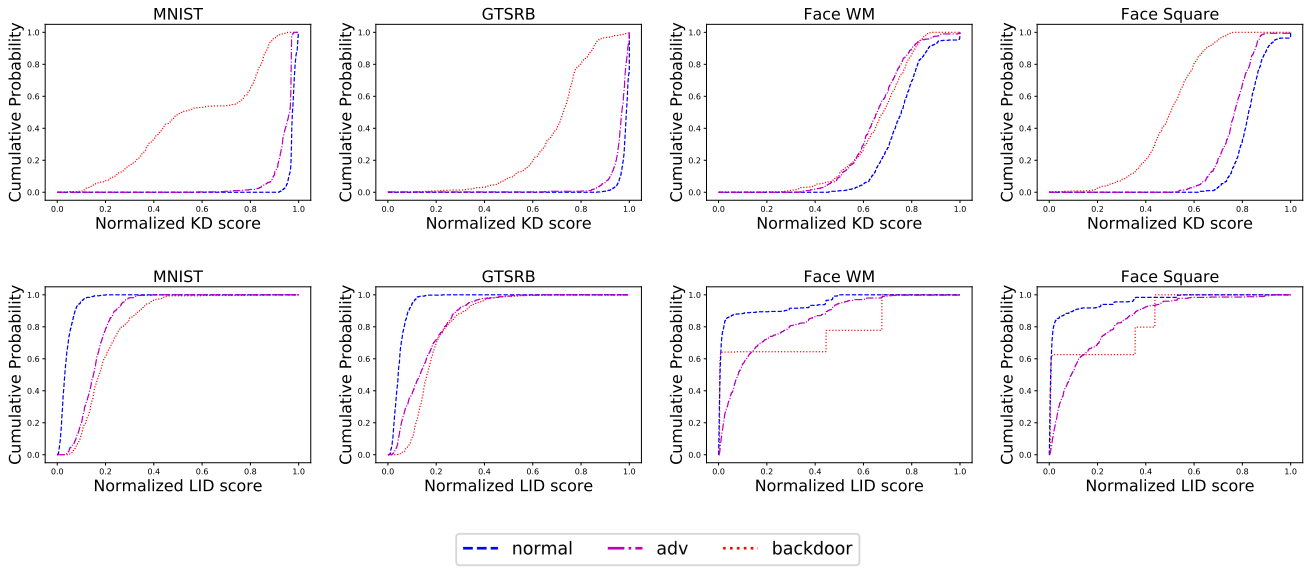


Fig. 11: Cumulative probability distribution of KD (first row) and LID (second row) values.

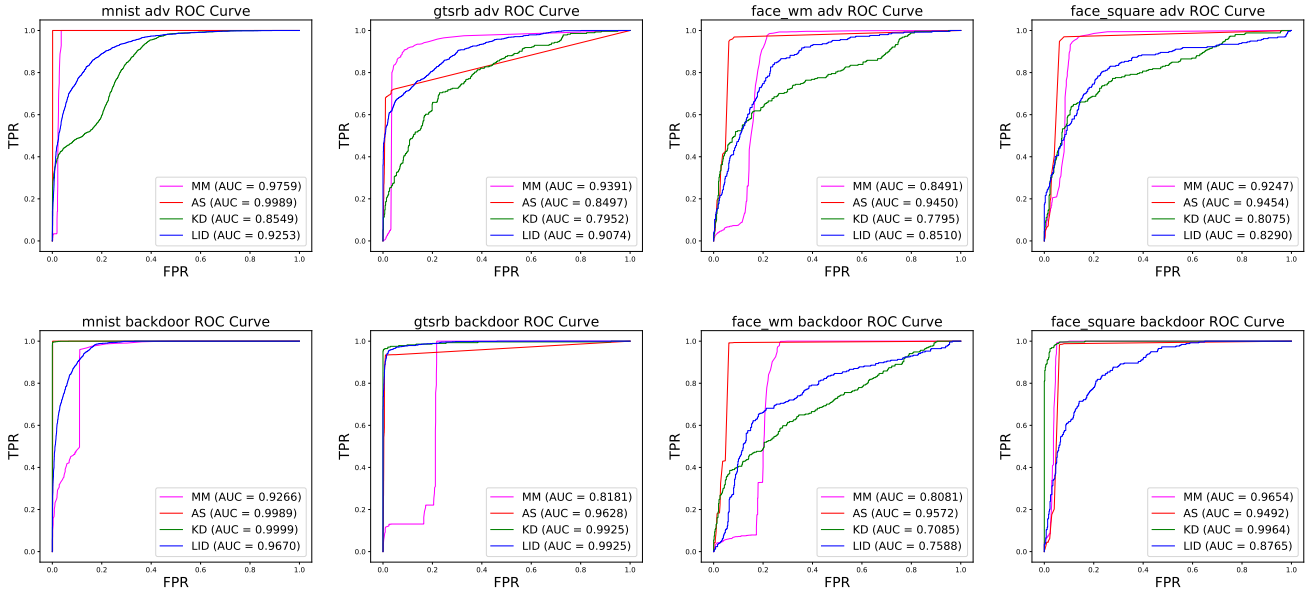


Fig. 12: ROC curve for detecting adversarial examples (first row) and backdoor examples (second row).

Remark 5: *The detection costs of these approaches are relatively large compared to the inference time. Detecting one sample can still be completed within 0.5 seconds. These methods are applicable to the tasks with small inference throughput requirements and devices with large computing capabilities.*

4.4 Detection of More Advanced Attacks

Moreover, we conduct the evaluation on three more sophisticated attacks: (1) Universal Adversarial Perturbation Attack (UAP) [57]: it adopts a universal perturbation for all normal samples to fool the target classifier; (2) Input-aware dynamic backdoor attack (IAB) [58]: this is an invisble backdoor attack that generates input-specific triggers. (3) Hidden Trigger Backdoor Attack (HTB) [73]: this generates invisible trigger to poison the training set and embed backdoors in the model. Figure 13 visualizes the corresponding adversarial and backdoor samples on the GTSRB dataset. Table 5 reports the performance of IAB and HTB backdoor attacks, including the accuracy of backdoor and normal samples compared to the original model accuracy.

TABLE 5: Attack success rate and classification accuracy for IAB and HTB Backdoor attacks.

Dataset	Attack	Infected Model		Clean Model
		Attack Success Rate	Natural Accuracy	Natural Accuracy
MNIST	IAB	100%	99.21%	99.06%
	HTB	78.37%	98.30%	
GTSRB	IAB	89.88%	97.01%	96.94%
	HTB	89.74%	96.57%	
PubFig	IAB	89.63%	89.52%	95.56%
	HTB	80.31%	35.63%	

We apply the considered detection approaches to those

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

(a) UAP (b) IAB (c) HTB (source and target)

Fig. 13: Malicious Examples from the advanced attacks.

attacks. Table 6 summarized the detection accuracy, and Figure 14 shows the corresponding ROC curves for each attack and solution. We observe that those approaches are still effective at detecting those advanced malicious samples. Particularly, LID gives the best performance compared to the other three.

TABLE 6: AUC results for UAP, IAB and HTB attacks.

Dataset	Attack	MM	AS	KD	LID
MNIST	UAP	**0.9890**	0.9423	0.9474	0.9714
	IAB	0.9572	0.9002	**1.00**	**1.00**
	HTB	0.9897	0.7505	**1.00**	**1.00**
GTSRB	UAP	0.7809	0.9900	0.6610	**0.9928**
	IAB	0.8823	0.9612	0.9865	**0.9935**
	HTB	0.8850	0.9318	0.9886	**0.9998**
PubFig	UAP	0.8679	0.9260	0.9293	**0.9995**
	IAB	0.6577	**0.9347**	0.8473	0.9041
	HTB	0.8277	0.7985	0.7113	**0.8726**

Remark 6: *The considered detection approaches are effective and general for more advanced attacks (e.g., universal perturbations or invisible triggers), as the malicious samples still have large differences from normal ones in the feature space. LID gives the best performance in particular.*

5 Other Defenses

In addition to the above four methods we have discussed and evaluated, we also test several other adversarial example detection algorithms in the backdoor scenario. They are relatively less effective, or in a lack of generality. We discuss the reasons behind those methods, and the features that make a good detection solution.

Bayesian Uncertainty estimates [59] is also based on the hypothesis that adversarial examples are are sensitive to model changes than normal samples, similar as the model mutation approach. Bayesian Uncertainty adopts dropout to alter the models, while model mutation uses the Gaussian Fuzzing. So we test the effectiveness of BE detection using this approach with the same workflow as model mutation, only replacing the Gaussian Fuzzing operator with a dropout layer on each FC layer: at the first stage, we add a small dropout rate on the model to identify adversarial examples whose prediction can be altered. At the second stage, we further increase the dropout rate to identify backdoor examples whose prediction is expected to be the same regardless of the dropout. Figure 15 shows the cumulative probability distribution of different types of samples under Mutation II. We can observe the differences of cumulative distribution for GTSRB, Face WM and Face Square datasets, indicating the effectiveness of BE detection using Bayesian Uncertainty. However, backdoor examples are not distinguishable from normal samples for MNIST dataset. This is confirmed by the detection results in Figure 16. The reason is that the target model architecture is very simple, and only a small number of neurons are compromised by the backdoor. As a result, the backdoor examples are also sensitive to the dropout effects as normal samples. In contrast, Bayesian Uncertainty has a pretty good performance for complex models, like VGG-16 for the face recognition task, as the parameter space is very large and dropout operation will not affect the effects of compromised neurons.

Remark 7: *Bayesian Uncertainty Estimate with dropout can be used to detect backdoor examples in complicated models. It does not work well when the backdoor model is too simple.*

Region-Based classification [64] detects AEs based on the hypothesis that AEs are closer to the decision boundary, and most neighbour labels in the hypercube of AEs are the correct labels. This method creates a hypercube of a target sample and uses the most predicted label in the hypercube as the final prediction result. Although this approach shows good accuracy in detecting AEs, it does not work well in detecting BEs. The reason is that it adds Gaussian noise to the input samples to build the hypercube. BEs with the trigger are much more robust against random noise than AEs. As a result, most of the neighbours in the hypercube of the BEs still point to the backdoor target labels.

Feature Squeezing [63] measures the confidence distance from the target input and its squeezed input. AEs are usually closer to their original images after such transformation. Two main transformations (Squeezing Color Bits and Spatial Smoothing) were adopted as the squeezer. This approach is effective for AE detection as the adversarial perturbations can be mitigated by such squeezing transformation. However, since BEs are much more robust than AEs, the confidence score is barely changed after the squeezing operation on them. Then Feature Squeezing fails to detect BEs with triggers. (Figure 17 shows the BEs transformed with median filter).

Remark 8: *Since BEs are more robust than AEs, input transformation based solutions generally fail to mitigate BEs, even they have been proved effective in defeating adversarial attacks.*

As we mentioned before, the machine learning and security communities focus on different types of adversarial attacks (e.g., adversarial examples, backdoor attacks) and

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3194642

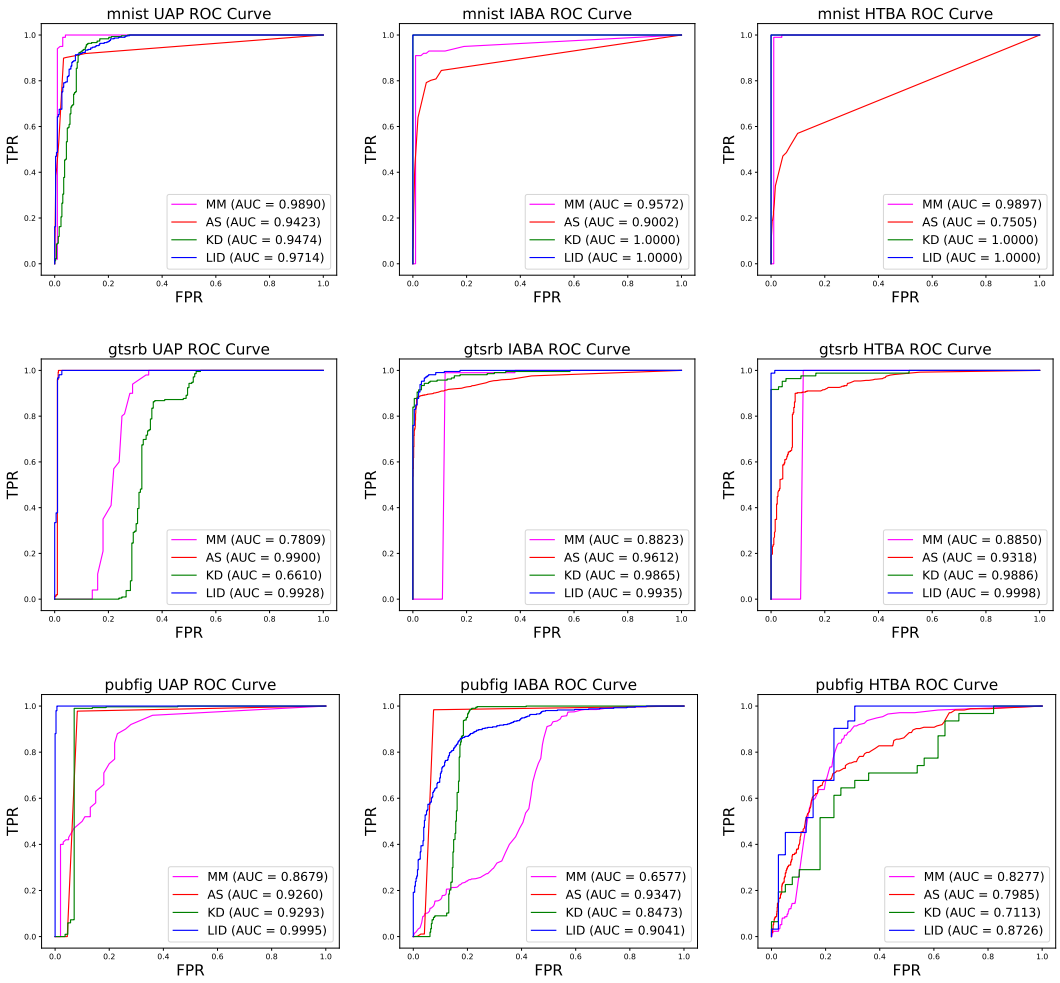


Fig. 14: ROC curves for detecting UAP, IAB and HTB attacks.

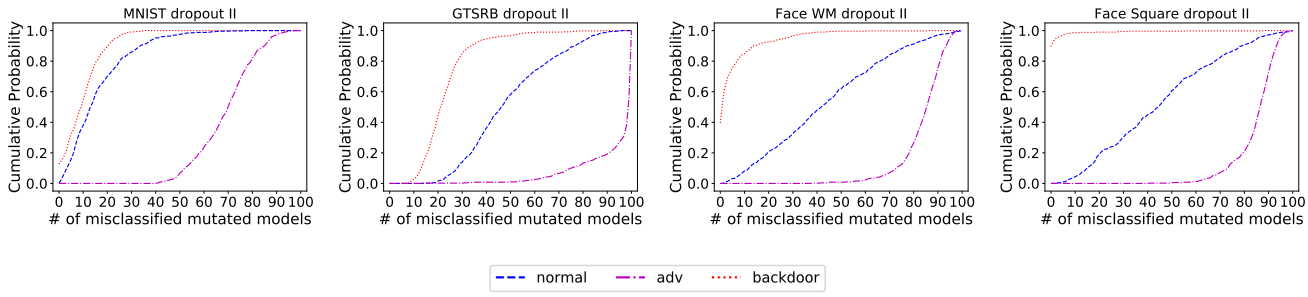


Fig. 15: Cumulative Distribution Function of three samples on the BU method.

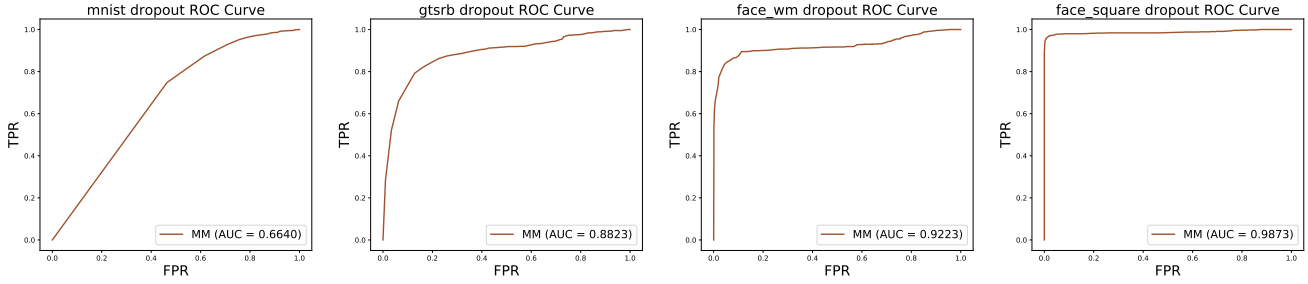
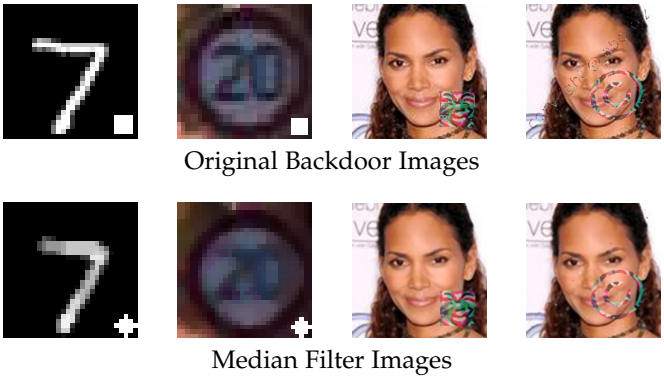


Fig. 16: ROC curve with BU method.



Original Backdoor Images

Median Filter Images

Fig. 17: Transformation with median filter. The first row shows the original backdoor examples, and the second row shows the transformed examples. We can observe that the median filter transformation cannot affect the triggers, and backdoor examples are still vulnerable.

their corresponding defense solutions. However, the connections between these threats are not well investigated, although they share certain similarities. There are only two works [40], [41] exploring the relationships between adversarial and backdoor examples, from the perspective of attacks. We present the first study towards the defenses of these threats. We believe our work reveals the common features of adversarial and backdoor attacks, which can facilitate the design of defenses. In particular, our findings on Activation Space and Feature Space for detecting malicious examples can be effective for building secure deep learning applications and systems. The Activation Clustering method [34] for detecting poisoning data can also help to improve adversarial defenses such as adversarial training [74], [75].

6 CONCLUSION

In this paper, we identify the connections between adversarial examples and backdoor examples in model sensitivity, feature space and activation space. Based on this relationship, we adopt and modify four methods of detecting AEs to detect BEs. Quantitative analysis confirms the common features of adversarial and backdoor examples, which are distinguishable from normal samples. Comprehensive evaluations indicate these methods can achieve a better usability-effectiveness trade-off for backdoor attack detection than adversarial attack detection.

Although the connection between adversarial examples and backdoor attacks were preliminarily explored in [40], [41] from the attack behaviors, this paper presents the first study towards such connection from the perspective of detection. We identify eight remarks, which can shed light on the design of more advanced defense solutions against backdoor attacks. In the future, we will extend our work with the following three directions: (1) we will focus on unifying other detection methods, and other types of defenses (e.g., removing perturbation via input preprocessing, combining the activation space and feature space [75], [76]). (2) We will adopt the ensemble of multiple detection approaches for better accuracy. (3) We will also analyze and interpret the connection and unification of adversarial and backdoor examples in a theoretical way. (4) In this paper, we only evaluate the state-of-the-art backdoor attacks. In the future, we will consider adaptive attacks which can try to make them stealthy in the feature space and activation space to bypass our detectors.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[4] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[6] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[8] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1145–1153.

[9] M. Fan, J. Liu, X. Luo, K. Chen, Z. Tian, Q. Zheng, and T. Liu, "Android malware familial classification and representative sample selection via frequent subgraph analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 1890–1905, 2018.

[10] M. Fan, X. Luo, J. Liu, C. Nong, Q. Zheng, and T. Liu, "Ctdroid: leveraging a corpus of technical blogs for android malware analysis," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 124–138, 2019.

[11] M. Fan, J. Liu, X. Luo, K. Chen, T. Chen, Z. Tian, X. Zhang, Q. Zheng, and T. Liu, "Frequent subgraph based familial classification of android malware," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 24–35.

[12] M. Fan, X. Luo, J. Liu, M. Wang, C. Nong, Q. Zheng, and T. Liu, "Graph embedding based familial analysis of android malware using unsupervised learning," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 771–782.

[13] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[14] S. G. Finlayson, H. W. Chung, I. S. Kohane, and A. L. Beam, "Adversarial attacks against medical deep learning systems," *arXiv preprint arXiv:1804.05296*, 2018.

[15] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of supervised models through robust optimization," *Neurocomputing*, vol. 307, pp. 195–204, 2018.

[16] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *arXiv preprint arXiv:1511.03034*, 2015.

[17] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[18] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[19] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[20] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 135–147.

[21] H. Lee, S. Han, and J. Lee, "Generative adversarial trainer: Defense to adversarial perturbations with gan," *arXiv preprint arXiv:1705.03387*, 2017.

[22] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyJ7ClWCb

[23] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8571–8580.

[24] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk9yuql0Z

[25] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations*, 2018.

[26] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 196–204.

[27] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[28] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence. AAAI Press*, 2019, pp. 4658–4664.

[29] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," *arXiv preprint arXiv:1908.01763*, 2019.

[30] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 004–14 013.

[31] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," *arXiv preprint arXiv:1910.03137*, 2019.

[32] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[33] T. J. L. Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," *arXiv preprint arXiv:1905.13409*, 2019.

[34] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[35] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 8000–8010.

[36] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," *arXiv preprint arXiv:1911.07116*, 2019.

[37] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 113–125.

[38] E. Chou, F. Tramèr, G. Pellegrino, and D. Boneh, "Sentinet: Detecting physical attacks against deep learning systems," *arXiv preprint arXiv:1812.00292*, 2018.

[39] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[40] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins: Adversarial inputs versus poisoned models," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 85–99.

[41] C.-H. Weng, Y.-T. Lee, and S.-H. B. Wu, "On the trade-off between adversarial and backdoor robustness," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[42] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[43] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[44] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[45] Y. Dong, F. Liao, T. Pang, X. Hu, and J. Zhu, "Discovering adversarial examples with momentum," *arXiv preprint arXiv:1710.06081*, 2017.

[46] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[47] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[48] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[49] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[50] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[51] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[52] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2041–2055.

[53] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 182–199.

[54] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1000–1008.

[55] H. Hosseini and R. Poovendran, "Semantic adversarial examples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1614–1619.

[56] A. Bhattad, M. J. Chong, K. Liang, B. Li, and D. A. Forsyth, "Unrestricted adversarial examples via semantic manipulation," *arXiv preprint arXiv:1904.06347*, 2019.

[57] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[58] A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *arXiv preprint arXiv:2010.08138*, 2020.

[59] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[60] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.

[61] Z. Katzir and Y. Elovici, "Detecting adversarial perturbations through spatial behavior in activation spaces," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–9.

[62] S. Tian, G. Yang, and Y. Cai, "Detecting adversarial examples through image transformation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[63] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed System Security Symposium*, 2018.

[64] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd*

Annual Computer Security Applications Conference, 2017, pp. 278–287.

[65] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1245–1256.

[66] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[67] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang, "Deepsec: A uniform platform for security analysis of deep learning model," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 673–690.

[68] IBM, "Adversarial robustness toolbox (art) v1.2," https://github.com/IBM/adversarial-robustness-toolbox.

[69] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[70] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[71] N. Pinto, Z. Stone, T. Zickler, and D. Cox, "Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook," in *CVPR 2011 WORKSHOPS*. IEEE, 2011, pp. 35–42.

[72] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[73] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 957–11 965.

[74] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *International Conference on Learning Representations*, 2019.

[75] Y. Bai, Y. Zeng, Y. Jiang, S.-T. Xia, X. Ma, and Y. Wang, "Improving adversarial robustness via channel-wise activation suppressing," *arXiv preprint arXiv:2103.08307*, 2021.

[76] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," *arXiv preprint arXiv:2101.05930*, 2021.

Kaidi Jin received the BS degree in computer science from Chongqing University, China, in 2018. He is currently working toward the PhD degree in the School of Cyber Science and Engineering, Xi'an Jiaotong University, China. His research interests include the robustness of machine learning models against adversarial attacks and backdoor attacks.

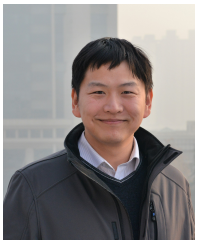
Tianwei Zhang is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelor's degree at Peking University in 2011, and the Ph.D degree in at Princeton University in 2017.

Shen Chao received the B.S. degree in Automation from Xi'an Jiaotong University, China in 2007; and the Ph.D. degree in Control Theory and Control Engineering from Xi'an Jiaotong University, China in 2014. He is currently a Professor in the Faculty of Electronic and Information Engineering, Xi'an Jiaotong University of China. He serves as the Associate Dean of School of Cyber Security of Xi'an Jiaotong University. He is also with the Ministry of Education Key Lab for Intelligent Networks and Network Security. His current research interests include AI Security, insider/intrusion detection, behavioral biometrics, and measurement and experimental methodology.

Yufei Chen received the B.S. degree in Electrical Engineering from Xi'an Jiaotong University, China in 2016. He is currently a Joint Ph.D. student with Xi'an Jiaotong University and City University of Hong Kong. His current research interests include AI security and behavioral biometrics.

Ming Fan received his B.S. and Ph.D. degrees in computer science and technology from Xi'an Jiaotong University, China, in 2013 and 2019, respectively. He received his Ph.D. degree in computing from The Hong Kong Polytechnic University in 2019. He is currently an associate professor in the School of Cyber Science and Engineering at Xi'an Jiaotong University, China. His research interests include trustworthy AI and Android malware analysis.

Chenhao Lin received the B.E. degree in automation from Xi'an Jiongtong University in 2011, the M.Sc. degree in electrical engineering from Columbia University, in 2013 and the Ph.D. degree from The Hong Kong Polytechnic University, in 2018. He is currently a Research Fellow in Xi'an Jiongtong University of China. His research interests are in artificial intelligence security, adversarial attack and robustness, biometric authentication and interpretable machine learning.



Ting Liu received his BS degree in information engineering and PhD degree in system engineering from Xi'an Jiaotong University, Xi'an, China, in 2003 and 2010, respectively. Currently, he is a professor in Xi'an Jiaotong University. His research interests include CPS, and AI software. He is a member of the the IEEE.