Feature-Indistinguishable Attack to Circumvent Trapdoor-Enabled Defense

Chaoxiang He*
Huazhong Univ. of Science and Tech.
hechaoxiang@hust.edu.cn

Bin Benjamin Zhu
Microsoft Research Asia
binzhu@microsoft.com

Xiaojing Ma†*
Huazhong Univ. of Science and Tech.
lindahust@hust.edu.cn

Hai Jin*
Huazhong Univ. of Science and Tech.
hjin@hust.edu.cn

Shengshan Hu*
Huazhong Univ. of Science and Tech.
hushengshan@hust.edu.cn

ABSTRACT

Deep neural networks (DNNs) are vulnerable to adversarial attacks. A great effort has been directed to developing effective defenses against adversarial attacks and finding vulnerabilities of proposed defenses. A recently proposed defense called *Trapdoor-enabled Detection* (*TeD*) [51] deliberately injects trapdoors into DNN models to trap and detect adversarial examples targeting categories protected by TeD. TeD can effectively detect existing state-of-the-art adversarial attacks. In this paper, we propose a novel black-box adversarial attack on TeD, called *Feature-Indistinguishable Attack* (*FIA*). It circumvents TeD by crafting adversarial examples indistinguishable in the feature (i.e., neuron-activation) space from benign examples in the target category. To achieve this goal, FIA jointly minimizes the distance to the expectation of feature representations of benign samples in the target category and maximizes the distances to positive adversarial examples generated to query TeD in the preparation phase. A constraint is used to ensure that the feature vector of a generated adversarial example is within the distribution of feature vectors of benign examples in the target category. Our extensive empirical evaluation with different configurations and variants of TeD indicates that our proposed FIA can effectively circumvent TeD. FIA opens a door for developing much more powerful adversarial attacks. The FIA code is available at: https://github.com/CGCL-codes/FeatureIndistinguishableAttack.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Artificial intelligence**; **Machine learning**;

*Chaoxiang He, Xiaojing Ma, Hai Jin, Shengshan Hu are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Hubei Engineering Research Center on Big Data Security, Hubei key Laboratory of Distributed System Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology.

†Xiaojing Ma is the corresponding author.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3485378

KEYWORDS

Neural Networks; Adversarial examples; Adversarial attacks; Trapdoor enabled defense; Feature-indistinguishable attack.

ACM Reference Format:
Chaoxiang He, Bin Benjamin Zhu, Xiaojing Ma, Hai Jin, and Shengshan Hu. 2021. Feature-Indistinguishable Attack to Circumvent Trapdoor-Enabled Defense. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3460120.3485378

1 INTRODUCTION

Deep neural networks (DNNs) have been proved to be effective at many difficult machine-learning tasks. However, they are found to be vulnerable to adversarial attacks [56], wherein an example correctly predicted by a deep learning model is intentionally modified slightly, usually undetectable by humans, to cause the model to make an incorrect prediction. These slightly modified examples are called *adversarial examples*. They are carefully crafted counterfactual examples with the aim to deceive the model. Adversarial examples can be classified into *targeted* or *untargeted* adversarial examples. A targeted adversarial example causes the model to misclassify it into a specific (i.e., target) category different from the original one, while an untargeted adversarial example causes the model to misclassify it into any category different from the original one. Targeted adversarial examples are generally harder to craft than untargeted adversarial examples.

Adversarial attacks are proved effective in deceiving deep learning models of different deep neural networks for different tasks [3, 10, 12, 14, 16, 17, 62] including real-world application scenarios [33, 34, 52]. Adversarial attacks have raised a serious concern on the security and reliability of deploying a deep learning model in real-world applications, esp. security-critical applications such as traffic-sign identification, face recognition, malware detection, etc.

Existence of adversarial examples has inspired significant research activities on both defenses against increasingly more powerful adversarial attacks and adversarial attacks to circumvent more and more sophisticated defenses. Many adversarial attacks have been developed since the first adversarial attack FGSM [20] was introduced, such as state-of-the-art attacks PGD [33, 34], C&W [10], and Elastic Net [11] that rely on computing gradients of the model in crafting adversarial examples, and BPDA [1] and SPSA [58] that bypass computing gradients of the model in crafting adversarial

examples. These adversarial attacks can be used to generate either targeted or untargeted adversarial examples. At the same time, various defense methods have also been proposed. They aim to disrupt computation of gradients, improve model's robustness against adversarial examples, or detect adversarial examples at inference time. Most of them are proved later to be vulnerable to more powerful adversarial attacks.

A new defense, called *Trapdoor-enabled Detection* (*TeD*) [51], was proposed recently to detect targeted adversarial examples at inference time. TeD deliberately injects one or more defensive trapdoors into a DNN model to protect one or more categories through backdoor attack techniques [23]. When crafting adversarial examples targeting a protected category of a trapdoored model, the optimization process of an adversarial attack gravitates towards trapdoors, leading to generated adversarial examples similar to trapdoors in the feature (i.e., neuron-activation) space. By comparing neuron activation signatures of inputs with those of trapdoors at a latent layer, referred to as the *detection layer*, adversarial examples can be detected. TeD can effectively detect state-of-the-art adversarial attacks such as PGD, C&W, Elastic Net, BPDA, and SPSA, with negligible impact on the normal classification accuracy [51].

Two attacks [6, 51] have been proposed to evade TeD, one is a white-box attack on TeD by assuming that adversaries know the trapdoor signatures used in detection, while the other is a grey-box attack by assuming that adversaries know some characteristics of the trapdoored defense, such as the number of trapdoors and the layer to detect. These methods craft adversarial examples by maximizing the distance to the known or estimated trapdoor signatures while minimizing the cross-entropy to the target category. They can evade TeD's baseline detection but their success rates are significantly reduced when TeD reinforces its detection with randomly sampled neurons and multiple trapdoors [51].

In this paper, we propose a novel targeted adversarial attack, called *Feature-Indistinguishable Attack* (*FIA*), to circumvent TeD. FIA is a black-box attack on TeD (and white-box on the model): adversaries have no knowledge of TeD or its characteristics. It relies only on the distribution of benign samples in the target category, referred to as *benign target samples*. FIA exploits the facts that most benign target samples should be undetected (i.e., negative) in a practical deployment of TeD and that a useful model is desirably well behaved (i.e., non-overfitting). Inspired by recently proposed adversarial attacks [29–31] that aim to strengthen adversarial transferability by optimizing at a latent layer in the feature space instead of the commonly used cross entropy, we minimize the distance to the expectation of feature representations of benign target examples at a latent layer, referred to as the *generation layer*, and ensure that generated adversarial examples are within a specific thresholding distance to the expectation in the feature space. We call this scheme the basic FIA scheme.

Adversarial examples generated with the basic scheme may still be distinguishable from benign target samples at the detection layer in the feature space due to mismatch between the generation layer and the detection layer, irregular undetectable boundaries of the trapdoored defense, etc. To improve indistinguishability of generated adversarial examples, FIA contains a preparation phase in which the basic scheme is used to generate a small number of adversarial examples to query the trapdoored defense to determine an appropriate generation layer and other generation parameters. FIA also uses the detected (i.e. positive) adversarial examples in this phase to generate adversarial examples by maximizing the distances to these positive adversarial examples while minimizing the distance to the expectation of benign target samples, both in the feature space.

We conduct an extensive empirical evaluation of our proposed attack on different configurations and variants of the trapdoored defense, including an improved variant called *Projection-based TeD* (*P-TeD*) that we propose in this paper, and compare it with existing state-of-the-art adversarial attacks. Our experimental study indicates that, while TeD and P-TeD can effectively detect existing state-of-the-art adversarial attacks, our proposed FIA can effectively circumvent the trapdoored defense of both TeD and P-TeD.

This paper includes the following major contributions:

- We propose a novel adversarial attack on DNN models that aims to generate adversarial examples indistinguishable in the feature space from benign examples of the target category. To the best of our knowledge, this is the first adversarial attack to evade detection by pursuing indistinguishability from benign samples.
- We propose a variant of TeD with an improved detection performance.
- We present an extensive empirical evaluation of our proposed adversarial attack on different configurations and variants of the trapdoored defense.
- Our proposed adversarial attack can effectively circumvent TeD and its variants that existing stat-of-the-art adversarial attacks cannot evade.

The remaining paper is organized as follows. We present the background and related work in Section 2, describe briefly TeD and its improved variant in Section 3, provide an overview of our proposed FIA in Section 4, and describe the detail of FIA in Section 5. Our empirical evaluation is presented in Section 6, and discussion is provided in Section 7. We conclude this paper with Section 8.

The FIA code is available at: https://github.com/CGCL-codes/FeatureIndistinguishableAttack.

2 BACKGROUND AND RELATED WORK

2.1 Adversarial Attacks against DNNs

A deep neural network (DNN) can be viewed as a function $\mathcal{F}_\theta$ that maps the input space $\mathcal{X}$ to the set of classification labels $\mathcal{Y}$, where θ represents the parameters of the network. An adversarial attack aims to craft a small perturbation ϵ for a normal input x such that the target model $\mathcal{F}_\theta$ will misclassify adversarial example $x + \epsilon$: $\mathcal{F}_\theta(x + \epsilon) \neq \mathcal{F}_\theta(x)$.

An adversarial attack can be either *targeted attack* or *untargeted attack*. The former attack aims to craft an adversarial example misclassified into a specific (i.e., target) category C_t: $\mathcal{F}_\theta(x + \epsilon) = C_t$, while the latter attack aims to craft an adversarial example misclassified into any category different from the original category. A targeted adversarial attack is generally harder to achieve than an untargeted adversarial attack. Adversarial attacks can also be classified into *white-box*, *grey-box*, and *black-box* attacks, according to adversary's level of accessibility to and knowledge of the model to be attacked: full, partial, and none, respectively.

Many white-box adversarial attacks have been proposed. Typical adversarial attacks are briefly described here. *Fast Gradient Sign Method* (*FGSM*) [20] is the first adversarial attack. To generate an untargeted adversarial example $x+\epsilon$ for input x, it crafts adversarial perturbation ϵ with a single step of value η along the direction of the gradient of the model's loss function to maximize the loss function to the original category C_o of x:

$$\epsilon = \eta \cdot sign(\nabla_x \ell(\mathcal{F}_\theta(x), C_o)),$$

where $\ell(\mathcal{F}_\theta(x), C_o)$ is the loss function to category C_o. It can also be used to generate a targeted adversarial example by minimizing the loss function to the target category C_t:

$$\epsilon = -\eta \cdot sign(\nabla_x \ell(\mathcal{F}_\theta(x), C_t)).$$

Like FGSM, other adversarial attack methods can also be used to generate both targeted and untargeted adversarial examples with similar relationships. Since the trapdoored defense aims to detect targeted adversarial examples, we describe adversarial attacks to craft targeted adversarial examples in the following description.

Projected Gradient Descent (*PGD*) [33, 34] extends FGSM to multiple iterations with double clipping. In each iteration,

$$x_{n+1} = Clip_\delta(x_n - \eta \cdot sign(\nabla_x \ell(\mathcal{F}_\theta(x_n), C_t)))$$

where $x_0 = x$, and $Clip_\delta$ operates on each pixel to clip its value within $\pm\delta$ of its original value in x and within the valid range. PGD is a much more powerful adversarial attack than FGSM.

Calini and Wangner Attack (*C&W*) [10] is considered one of the most powerful adversarial attacks. It aims to generate adversarial examples with minimized perturbations:

$$\min_\epsilon \|\epsilon\|_p + c \cdot \ell(\mathcal{F}_\theta(x+\epsilon), C_t),$$

where c is a parameter to weight the two objectives. Its optimal value can be found with a binary search. *Elastic Net* [11] is a variant of C&W that minimizes both L_1 and L_2 norms of adversarial perturbation ϵ together with minimizing the loss function to the target category.

Jacobian-based Saliency Map Attack (*JSMA*) [46] uses a Jacobian-based saliency map to model the impact of each pixel on the resulting classification and applies a greedy algorithm to iteratively modify the most influential pixel in crafting adversarial examples.

To deal with gradient obfuscation defenses (see Section 2.2), *Backward Pass Differentiable Approximation* (*BPDA*) [1] computes gradients using differentiable approximation, expectation, or reparameterization to overcome different types of obfuscated gradients, while *Simultaneous Perturbation Stochastic Approximation* (*SPSA*) [58] avoids computation of gradients by using stochastic sampling to find the global minimum in solving the optimization problem of deriving adversarial perturbations.

While most adversarial attacks are white-box attacks, black-box adversarial attacks have also been proposed [4, 44, 45, 55]. A practical attack proposed in [45] trains a substitute model with a synthetic dataset and uses the substitute model to craft adversarial examples. Since both models have similar decision boundaries, adversarial examples crafted with the substitute model are likely transferable to the black-box target model. Adversarial perturbations can also be crafted using differential evolution [55] or greedy local search [44].

When facing the trapdoor-enabled detection [51], commonly used and state-of-the-art adversarial attacks like FGSM, PGD, C&W, Elastic Net, BPDA, and SPSA are all ineffective [51].

2.2 Defenses against Adversarial Attacks

Along with the development of different adversarial attacks, many defense methods have also been developed. Early attempts to secure neural networks against adversarial attacks include removing adversarial perturbations with denoising autoencoders [22], increasing local stability [49] or robustness of DNNs via robustness metrics [2] or adversarial training [28, 41, 57, 64, 65], and defensive distillation [47] that uses the distillation technique [27] to retrain the same network with category probabilities predicted by the original network. These defenses fail or are significantly weakened when facing stronger adversarial attacks or high-confidence adversarial examples [7–9, 26].

Since most adversarial attacks, such as FGSM [20], PGD [33, 34], C&W [10], and Elastic Net [11], rely on computing gradients in crafting adversarial examples, many defenses [5, 15, 24, 40, 48, 53, 61] apply gradient masking (i.e., reducing useful gradients) to disrupt computation of gradients to thwart adversarial attacks. These gradient-masking defenses are proved vulnerable to black-box attacks [45, 57], gradient-approximation attacks like BPDA [1], and adversarial attacks without using gradients like SPSA [58].

Defenses focusing on detecting adversarial examples at inference time are also proposed. Many utilize statistical tests to differentiate adversarial examples from benign examples [19, 21, 38, 63]. These defenses fail to detect more powerful adversarial attacks such as C&W [10]. *Magnet* [42] uses one or more detection networks and a reformer network. The detection networks learn to distinguish between normal and adversarial examples, while the reformer network is used to remove any remaining minor adversarial nature in examples classified as benign by the detectors. However, Magnet is found vulnerable to an attack that adversaries train their own copy of the defense and generate transferable adversarial examples [9]. *Latent Intrinsic Dimensionality* (*LID*) [40] exploits the difference of a model's internal dimensionality characteristics between normal and adversarial examples to detect adversarial examples. LID is found [1] to be vulnerable to high confidence adversarial examples of C&W. *Neural-network Invariant Checking* (*NIC*) [39] extends LID by using one-class Support Vector Machines (SVM) to model the benign distribution of latent activation within and across layers to detect adversarial examples without using any prior knowledge of adversarial attacks.

A recently proposed detection method, *Trapdoor-enabled Detection* (*TeD*) [51], deliberately injects one or more trapdoors into DNN models through embedding defensive backdoors to trap and detect adversarial examples. When an adversarial attack searches for minimum perturbations to a target category protected by TeD in crafting adversarial examples, it is likely trapped into a shortcut created by the backdoored model, resulting in crafted adversarial examples likely detected with signatures of backdoor samples extracted from their latent representations. More details are provided in Section 3. TeD can detect, with high accuracy, adversarial examples generated by state-of-the-art attacks such as PGD, C&W, Elastic Net, and BPDA, with negligible impact on normal classification [51].

While the above defenses are all best-effort approaches, an emerging approach called *randomized smoothing* is proposed [13, 36, 37] to transform the original classifier into a smoothed classifier, which is used to return the category with the highest probability by querying isotropic Gaussian $N(x, \sigma^2 I)$ around an input x. Randomized smoothing provides certifiable robustness against adversarial examples within an L_2 ball around any input x, a desirable property that best-effort approaches lack. However, current randomized smoothing methods may not be practical for many applications due to their reduced accuracy [43] and significant inference overhead [13]. Best-effort defenses are still needed before randomized smoothing becomes more practical.

2.3 Related Work

2.3.1 Activation Attack and Its Variants. Our FIA is related to *Activation Attack* (*AA*) [31], a transfer-based black-box targeted adversarial attack. It uses a white-box model to generate adversarial examples to attack a black-box model. To strengthen transferability of adversarial examples, AA crafts adversarial examples by minimizing the Euclidean distance to a target example at some latent layer in the feature space. The target example is the one with the largest Euclidean distance to the feature vector of the source example of the current adversarial example among a small set of benign examples randomly sampled from the target category.

The same team has developed several variants [29, 30] with the same goal: strengthening transferability of adversarial examples. *Feature Distribution Attack* (*FDA*) [29] captures layer-wise and category-wise feature distributions of the white-box model with binary neural networks and generates adversarial examples by maximizing the target category probability at a latent layer, optionally minimizing the source category probability or maximizing the distance of the perturbed features from the original features at the same layer simultaneously. FDA is extended in [30] to multiple layers by optimizing these layers simultaneously and also by adding the cross-entropy loss function to optimize.

Our FIA differs from AA and its variants by pursuing a different goal. We aim to generate adversarial examples indistinguishable from benign examples of the target category while AA and its variants aim to strengthen transferability of adversarial examples from the white-box model to the black-box model. The differences in the goal to pursue result in the following major differences between our FIA and AA and its variants. First, the target to drive adversarial examples to is different. The target in AA is the example with the furthest distance to the source example in the feature space among a small set of benign examples randomly sampled from the target category, while the target in FIA is the expectation of feature vectors of benign examples in the target category to maximize the chance that a generated adversarial example is indistinguishable from benign examples in the target category. Second, our generated adversarial examples are guaranteed, according to our indistinguishability metric, to be within the distribution of and thus indistinguishable from benign examples in the target category (otherwise the adversarial example generation fails), while adversarial examples generated by AA and its variants may not be indistinguishable from benign examples of the target category since the focus is on strengthening adversarial transferability. Third, FIA has an additional term to optimize that AA and its variants lack: maximizing the distances to positive examples in the feature space.

2.3.2 Existing Attacks on Trapdoor-enabled Detection. Carlini [6] introduces two advanced attacks, *Oracle Signature Attack* (*OSA*) and *Trapdoor Vault Attack* (*TVA*), that can partially break the strengthened trapdoor-enabled detection. Oracle Signature Attack [6, 51] is a white-box attack for both the model and the trapdoored defense: adversaries have the knowledge of precise values of the trapdoor signature(s). It optimizes for both maximum cosine distance to the known trapdoor signature and minimal cross-entropy to the target category. It can evade the trapdoor-enabled detection with a success rate nearly 90% on MNIST when randomized neuron sampling is not used, but the attack success rate reduces to below 40% after 5% of randomly sampled neurons and multiple trapdoors are used [51].

Trapdoor Vault Attack [6, 51] is a weaker attack than Oracle Signature Attack. In this attack, adversaries know the basic setting of the trapdoor-enabled detection, such as the number of trapdoors used in the trapdoored defense, but have no knowledge of any trapdoor signature. The attack estimates the trapdoor signature(s) from adversarial examples generated with a conventional method such as PGD. For the case that N trapdoors are used in the trapdoored defense, it uses a clustering approach to approximate neuron signatures for each of the N trapdoors. The attack then uses the estimated trapdoor signature(s) with the same approach as Oracle Signature Attack to generate adversarial examples. Its success rate is lower than that of Oracle Signature Attack. When 5% of randomly sampled neurons and multiple trapdoors are used, its success rate is around 20% on MNIST [51].

Our FIA differs from both of Carlini's attacks in several ways. First, FIA is a black-box attack on the trapdoored defense (but white-box on the model): adversaries have no knowledge of the characteristics of the the trapdoored defense (e.g., how many trapdoors or which layer's signatures are used). FIA does not need in general to know or estimate trapdoor signatures used in the trapdoored defense. Second, the optimization goal is different. FIA minimizes the distance to the expectation of benign examples in the target category at some latent layer and ensures that the feature vector of a generated adversary example at that latent layer is indistinguishable from those of benign samples in the target category, while Carlini's attacks minimize the conventional adversarial loss (i.e., the cross entropy to the target category). Third, Carlini's attacks maximize the distance to the known or estimated trapdoor signatures (and thus adversaries need to know the detection layer and the number of trapdoor signatures used in the trapdoored defense), while FIA maximizes distances to positive adversarial examples it queries the trapdoored defense at the preparation phase. According to our ablation study presented in Section 6.7, maximizing distances to positive adversarial examples plays a minor role in FIA. In many cases, adversarial examples generated with the basic FIA scheme are all negative (i.e. undetected) for TeD in the preparation phase, and thus FIA actually uses the basic scheme (which does not have the maximizing term) to generate adversarial examples (see Section 5.2 for details).

2.4 Notation

The notation used in this paper is summarized in Table 1.

Table 1: Notation.

Notation	Definition
$\mathcal{F}_\theta$	DNN model with parameters θ.
$\mathcal{X}, \mathcal{Y}$	Input space and output space of model $\mathcal{F}_\theta$.
$\mathcal{F}_L(x)$	Feature representation of $x \in \mathcal{X}$ at layer L of $\mathcal{F}_\theta$.
C_t	Target category t
$\mathcal{D}_t$	Feature representation distribution of category t
ϵ	Adversarial perturbation
δ	Pixel-wise bound on adversarial perturbations
$\mathbf{D}(\cdot)$	Distance function
$\mathbf{E}(\cdot)$	Expectation function
$\cos(x, y)$	Cosine similarity between x and y.

3 TRAPDOOR-ENABLED DETECTION

3.1 Backdoor Attacks

Trapdoor-enabled Detection (TeD) [51] exploits DNN's vulnerability to backdoor attacks to train defensive backdoored DNN models to detect adversarial examples. Backdoor attacks [23] are another well-known vulnerability of DNNs. While adversarial attacks are passive attacks that don't change the target model, backdoor attacks are active attacks that require modification of the target model to inject one or multiple backdoors into the target model. In a backdoor attack, the adversary selects a target category and a special pattern, called *backdoor trigger*, and injects the backdoor into a DNN model through poisoning training data. A backdoored model behaves normally and has similar accuracy as a clean model when the backdoor trigger is not applied. When the backdoor trigger is applied on an arbitrary normal example of any category, the backdoored model will misclassify it into the target category.

3.2 Trapdoor-enabled Detection (TeD)

TeD deliberately injects "trapdoors" into a DNN model through embedding one or multiple defensive backdoors to trap and detect adversarial examples. Since targeted adversarial attacks essentially search for a minimum perturbation to the target category with an optimization algorithm in crafting adversarial examples, generated adversarial examples will be most likely trapped into a shortcut created by the backdoor and thus can be detected with signatures of backdoor samples extracted from their latent representations.

TeD selects a latent layer, called *detection layer* in this paper, to detect adversarial examples. A detection layer should be a layer late in the forward pipeline of a DNN model, usually the penultimate layer (i.e., the last layer before the output softmax layer). After training trapdoored model $\mathcal{F}_\theta$ with a trapdoor trigger Δ for target category C_t, TeD calculates trapdoor signature S_Δ as follows:

$$S_\Delta = \mathbf{E}_{x \notin C_t} \mathcal{F}_L(x + \Delta), \tag{1}$$

where $\mathbf{E}(\cdot)$ is the expectation function, $\mathcal{F}_L(x)$ is the feature representation of input $x \in \mathcal{X}$ at detection layer L of model $\mathcal{F}_\theta$. To build this signature in practice, the model owner computes and records neuron activation vectors of many inputs containing trigger Δ.

To determine if input $x \in \mathcal{X}$ is an adversarial example or not, TeD calculates the cosine similarity between the feature representation of x at detection layer L, $\mathcal{F}_L(x)$, and trapdoor signature S_Δ. If the similarity exceeds a preset threshold ϕ_t, input x is determined as adversarial (i.e., positive). Otherwise it is determined as normal (i.e., negative). Threshold ϕ_t is chosen as the k-th percentile value of the statistical distribution of cosine similarity between benign samples and S_Δ, with $1 - \frac{k}{100}$ as the desired false positive rate.

To thwart the two attacks proposed by Carlini [6] to circumvent TeD, TeD can be enhanced by injecting multiple trapdoors to protect a single category C_t. Each trapdoor is associated with a trapdoor signature at the detection layer. In this case, TeD calculates the cosine similarity of input x with each of the trapdoor signatures. If any one exceeds the preset threshold, it determines that the input is adversarial. In addition, TeD can detect adversarial examples using the activation of a subset of neurons, for example, 5% or 10% of randomly sampled neurons at detection layer L.

On a subset of randomly sampled neurons, it is possible that the distribution of benign samples in target category C_t and trapdoored samples are not well separated, resulting in a high false positive rate if the subset is used in detection. The authors of TeD haven't provided details on selecting or using subsets of neurons in their paper [51] or their released code [50]. In our experimental evaluation (Section 6), we use the following way to select a subset of neurons at the detection layer: given a number of neurons such as 5% to select, we randomly sample neurons to form a subset of neurons. Then we use the expectation of trapdoored samples on the subset, say S^1_Δ, as the detection signature, calculate the cosine similarity distributions of both trapdoored samples and benign samples in target category C_t with S^1_Δ, and determine k_Δ-th percentile value v_{k_Δ} of the trapdoored sample distribution and k_{C_t}-th percentile value $v_{k_{C_t}}$ of the benign sample distribution. In our evaluation, we set $k_\Delta = 25$ and $k_{C_t} = 75$. If $v_{k_\Delta} - v_{k_{C_t}}$ is larger than a preset threshold, the subset can be used; otherwise it is not used. We select N useful subsets in this way. The selected subsets are used in the same way as in the case of multiple trapdoors: if any of the subsets signals positive, the input sample is determined to be adversarial, otherwise it is determined to be benign.

3.3 Projection-based TeD (P-TeD)

In our experimental evaluation of TeD, using the proposed signature S_Δ may not lead to a good separation between trapdoored samples and benign samples in the category protected by the trapdoor. This is because they may have similar values on some neurons. To increase the detection sensitivity, we have tried several ways to increase the separation of trapdoored samples and benign target samples: ordering neurons according to the separation and selecting a subset of neurons with top separation values, projecting S_Δ to the expectation of benign samples in C_t, etc. We have found that the projection method produces the best detection results. We call this TeD variant as *Projection-based Trapdoor-enabled Detection* (*P-TeD*). The detail for using a single trapdoor will be described. The cases using multiple trapdoors and sampling neurons can be done in a similar manner.

When a single trapdoor is used to protect a target category C_t, we calculate trapdoor signature S_Δ with Eq. 1 and the expectation of feature representations of benign samples in C_t as the benign signature:

$$S_{C_t} = \mathbf{E}_{x \in C_t} \mathcal{F}_L(x). \tag{2}$$

Then we remove the projection of S_Δ on S_{C_t} from S_Δ to get its component perpendicular to S_{C_t}:

$$S_\Delta^\perp = S_\Delta - \frac{S_\Delta \cdot S_{C_t}}{\|S_{C_t}\|_2^2} S_{C_t}, \tag{3}$$

where $A \cdot B$ means the inner product of A and B. $S_\Delta^\perp$ is used as the signature to detect adversarial examples in the same way as S_Δ is used in TeD.

4 FIA OVERVIEW

4.1 Threat Model

We assume that the trapdoored DNN model is white-box while the trapdoored defense is black-box. More specifically, we assume that adversaries have full access to the trapdoored model, including its architecture and internal parameter values, and are aware of protection by the trapdoored defense but have no knowledge of its characteristics (i.e., the number of trapdoors, trapdoor signatures, detection layer, etc.). The trapdoored defense behaves like an oracle to adversaries: when an example is received, it responds with the detection result, either positive or negative, that the detector has determined. No information on the distance to any trapdoor signature is provided.

In addition, we assume that adversaries have no access to any training data or backdoor triggers used to train a trapdoored model and cannot apply any reverse-engineering technique such as *Neural Cleanse* [59] to estimate the number of trapdoors, backdoor triggers, or trapdoor signatures used in the defense, either due to unknown characteristics of the trapdoored defense or other reasons. This restriction of no access to any reverse-engineering technique ensures that the white-box access to the trapdoored model can only be used to search for adversarial examples like conventional white-box adversarial attacks. It cannot be used to derive the characteristics of the trapdoored defense.

On the other hand, we assume that adversaries have access to a small set of benign data independent of the training data used in training a trapdoored model.

4.2 Intuition behind FIA

TeD relies on the following two key factors to work: one is that searching for targeted adversarial examples is likely trapped into shortcuts created by trapdoors of a trapdoored model, and the other is that trapdoor signatures should be significantly different from those of benign examples of the target category. The former ensures that adversarial examples are likely detected by the trapdoor signatures, while the latter ensures a low false positive rate, which is critical to deploying any defense in real-world applications. In addition, along its forward path, a DNN such as CNN extracts features from low-level to high-level, and distinguishability of different categories in the feature space gradually improves. For good detectability, the detection layer L of TeD should be close to the end of the forward pipeline, usually the penultimate layer.

To circumvent the trapdoored detection, we aim to make adversarial examples indistinguishable from benign target examples in the feature space. Once an adversarial example is indistinguishable from benign target examples at an appropriate latent layer, it is unlikely detectable by the trapdoored defense unless the trapdoored defense is impractical with a high false positive rate for benign target samples. To achieve this goal, we need to craft adversarial examples with a loss function that can drive adversarial examples into the distribution of benign target samples in the feature space. The conventional loss function used in crafting adversarial examples is the cross entropy, which has no control on the feature vector of a generated adversarial example at a given latent layer. Fortunately, Inkawhich et al. [29–31] proposed recently to use a loss function in the feature space to craft more transferable adversarial examples so that adversarial examples generated on a white-box model can be transferred to attack a black-box model. We can adopt their approach to circumvent the trapdoored detection.

Another obstacle we need to overcome is to find a metric to measure distinguishability of an example from a set of examples in the feature space. If we assume that the trapdoored model and the trapdoored detection are both stable, we expect that an example with its feature vector located within a small dense region of benign target examples at an appropriate latent layer is likely classified and detected like its nearby benign target examples. This stability is generally required for a practical DNN model with good behaviors. With this assumption, we can simply find a target inside the dense region, minimize the distance to the target in the feature space, and check if such a distance is within an appropriate threshold at the end of crafting. This fulfills the basic scheme of our proposed FIA.

Adversarial examples generated with the basic scheme may still be detectable due to mismatch between the generation layer and the the detection layer, irregular undetectable boundaries of the trapdoored detection, over-simplified modeling of indistinguishability, etc. FIA contains a preparation phase wherein the basic scheme is used to generate a small number of adversarial examples to query the trapdoored detection for determining an appropriate generation layer and other generation parameters. It also adds a loss term to the basic scheme to maximize distances to detected adversarial examples in the preparation phase to avoid regions where trapdoor signatures may be located.

5 FEATURE-INDISTINGUISHABLE ATTACK

Our proposed Feature-Indistinguishable Attack (FIA) is described in detail in this section. In addition to the basic scheme and the complete scheme, we also present a variant of FIA.

5.1 Basic Scheme

5.1.1 Optimization Problem. For input $x \in \mathcal{X}$ that is not in target category C_t, $x \notin C_t$, and a latent representation distribution $\mathcal{D}_t$ of target category C_t at a selected latent layer L, called *generation layer*, the basic scheme of our proposed FIA minimizes the distance of the latent representation $\mathcal{F}_L(x+\epsilon)$ of $x+\epsilon$ at layer L to a target representation $\mathcal{F}_L^{tgt} \in \mathcal{D}_t$ on the condition that $\mathcal{F}_L(x+\epsilon)$ is within the latent representation distribution $\mathcal{D}_t$ of target category C_t:

$$\begin{aligned} &\min_{\epsilon} \mathbf{D}(\mathcal{F}_L(x+\epsilon), \mathcal{F}_L^{tgt}), \\ &\text{s.t. } \mathcal{F}_L(x+\epsilon) \in \mathcal{D}_t \end{aligned} \tag{4}$$

where $\mathbf{D}$ is a distance function. The constraint in Eq. 4 ensures that a generated adversarial example is indistinguishable from benign examples in target category C_t.

FIA uses two distance loss functions in Eq. 4 to minimize both distances simultaneously. One is an L_2 distance like that in Activation Attack [31]. Its goal is to drive adversarial examples into target category C_t. The other is the cosine similarity with the target representation. Its goal is to ensure that an adversarial example has a feature vector along a direction similar to that of the target representation so that TeD, which relies on cosine similarity with trapdoor signatures in the feature space, unlikely detects it. The optimization problem becomes:

$$\min_{\epsilon}\{-\cos(\mathcal{F}_L(x+\epsilon), \mathcal{F}_L^{tgt}) + \lambda \cdot \|\mathcal{F}_L(x+\epsilon) - \mathcal{F}_L^{tgt}\|_2\}, \quad \text{s.t. } \mathcal{F}_L(x+\epsilon) \in \mathcal{D}_t \tag{5}$$

where $\lambda \geq 0$ is a weighting parameter. Note that there is a negative sign before the cosine similarity since we want to maximize the cosine similarity with the target representation.

5.1.2 Basic Scheme. To ensure the constraint in Eq. 5, we need to estimate the feature-representation distribution $\mathcal{D}_t$ of target category C_t. Although it is possible to use neural networks to model the feature-representation distribution of target category C_t like in [29], FIA adopts a less accurate but much simpler approach: we assume that feature representations of benign examples in target category C_t can form a convex region[1]. With this assumption, we can choose the expectation of feature representations of benign examples in C_t, $\mathcal{F}_L^{C_t}$, as the target representation $\mathcal{F}_L^{tgt}$:

$$\mathcal{F}_L^{tgt} = \mathcal{F}_L^{C_t} \equiv \mathbf{E}_{x \in C_t} \mathcal{F}_L(x), \tag{6}$$

where $\mathbf{E}(\cdot)$ is the expectation function, and use the cosine similarity distribution of benign examples in C_t with $\mathcal{F}_L^{C_t}$ as an approximation of distribution $\mathcal{D}_t$.

Since expectation is sensitive to outliers, we determine and remove outliers with DBSCAN [18] before calculating the expectation in Eq. 6: a certain percentage (10% in our evaluation) of benign target samples located in the lowest-density regions in the feature space are considered as outliers and removed. We then calculate a threshold c_p to be the smallest cosine similarity between the expectation and survived benign target samples. We require that the cosine similarity of an adversarial example with the expectation $\mathcal{F}_L^{C_t}$ is within c_p.

With the above assumption and simplification, Eq. 5 becomes:

$$\min_{\epsilon}\{-\cos(\mathcal{F}_L(x+\epsilon), \mathcal{F}_L^{C_t}) + \lambda \cdot \|\mathcal{F}_L(x+\epsilon) - \mathcal{F}_L^{C_t}\|_2\}, \quad \text{s.t. } \cos(\mathcal{F}_L(x+\epsilon), \mathcal{F}_L^{C_t}) \geq c_p \tag{7}$$

5.1.3 Adaptive Iteration. Since the constraint in Eq. 7 is on the first loss term in Eq. 7, we can apply the following adaptive iteration to solve Eq. 7: We start to drive only the first loss term by setting $\lambda = 0$ until the constraint is met. We then activate λ by setting it to a non-zero value (1 in our evaluation) to minimize both loss terms simultaneously to drive $x + \epsilon$ to target category C_t while maintaining satisfying the constraint. When $x + \epsilon$ is classified into C_t and its softmax probability is larger than the next maximum softmax probability by a specific threshold, we return to drive only the first loss term again by setting $\lambda = 0$ as long as the softmax probability gap is maintained, otherwise we activate λ to drive both loss terms. The softmax gap is used to ensure that a generated adversarial example is robustly classified into target category C_t.

If we place an L_∞ bound δ on adversarial perturbation ϵ, Eq. 7 can be solved with a PGD-like iterative process with the above adaptive method:

$$x_0 = x, \quad x_{n+1} = Clip_\delta(x_n - \eta \cdot sign(\nabla_x \ell(\mathcal{F}_L(x_n), \mathcal{F}_L^{C_t}))) \tag{8}$$

where $\ell(\mathcal{F}_L(x_n), \mathcal{F}_L^{C_t}))$ is the loss function of the basic scheme:

$$\ell(\mathcal{F}_L(x), \mathcal{F}_L^{C_t})) = -\cos(\mathcal{F}_L(x), \mathcal{F}_L^{C_t}) + \lambda \cdot \|\mathcal{F}_L(x) - \mathcal{F}_L^{C_t}\|_2 \tag{9}$$

The iterative process can stop early if x_n is classified into target category, $x_n \in C_t$, and $\cos(\mathcal{F}_L(x_n), \mathcal{F}_L^{C_t}) \geq c_p$. Otherwise a preset number of steps is executed before it stops. In our empirical evaluation, stopping early is used for the basic scheme used in the preparation phase (see Section 5.2.1), and a fixed number of steps is executed for the complete scheme to generate adversarial examples.

By the end of the iterative process, the generation of an adversarial example is said successful if the resulting example is classified into target category C_t by the model and the constraint in Eq. 7 is satisfied. Otherwise the generation is a failure.

5.2 Complete Scheme

Adversarial examples generated with the basic scheme may still fail to circumvent the trapdoored detection due to several reasons: mismatch between the generation layer and the unknown detection layer of TeD, irregular undetectable boundaries of the trapdoored defense, over-simplified convex-region assumption for the feature-representation distribution of the target category and the subsequent simplification from Eq. 5 to Eq. 7.

The second reason can be explained as follows. Suppose the generation layer and the detection layer are the same latent layer. Since TeD uses cosine similarity with trapdoor signatures to detect adversarial examples, examples with the same cosine similarity value with the expectation vector may have different detectability: those close to the trapdoor signatures have a higher chance to be detected than those far away from the trapdoor signatures. Unfortunately, the basic scheme treats these examples identically.

To address these issues, we enhance the basic scheme with two additions. One addition is a preparation phase to use the basic scheme to generate some adversarial examples to query the trapdoored defense to determine an appropriate generation layer and other generation parameters. The other addition is an additional term in the loss function to direct adversarial examples away from bad regions where detected adversarial examples in the preparation phase stay. These positive adversarial examples are a rough estimate of trapdoor signatures[2].

5.2.1 Preparation Phase. In this phase, we use the basic scheme to generate a small number of adversarial examples to query the trapdoored detector. The query result is used to determine first an appropriate generation layer and then generation parameters.

[1]This assumption is generally over-simplified for a DNN model, esp. when a middle latent layer is used as the generation layer. See Section 6.5 for more information.

[2]Conventional adversarial attacks such as PGD should produce a more accurate estimate of trapdoor signatures than detected adversarial examples in the preparation phase (see Section 6.5 for more information). Since this driving-away loss term plays a minor role in generating adversarial examples (see Section 6.7), this rough estimate should be sufficient.

To determine an appropriate generation layer, we generate a small number of adversarial examples with the basic scheme for each of potential generation layers, starting with the penultimate layer and moving backward, to query the trapdoored defense. If the detection rate is below a threshold (i.e., when we consider an appropriate generation layer is found), we stop and choose the layer with the smallest detection rate as the generation layer. Since the penultimate layer is generally used as the detection layer in the trapdoored defense and the basic scheme works reasonably well (see Section 6.7), an appropriate generation layer can be found quickly without searching many layers.

Once the generation layer is determined, we use the query result of the layer to adjust target $\mathcal{F}_L^{C_t}$ and constraint threshold c_p in Eq. 7 to be used in the generation phase. If the generated adversarial examples are all negative, the basic scheme with the original $\mathcal{F}_L^{C_t}$ and c_p will be used to generate adversarial examples. If the detection rate is high enough (above a preset threshold), which occurs typically when TeD has a high false positive rate for benign target samples (see Section A.3), we query the TeD detector with benign target samples until we have collected enough negative samples (10 in our experimental evaluation). In this case, the benign target samples that are negative and untested are used to calculate a weighted average, with more weight (double in our evaluation) for the negative benign examples. The detected benign target examples (i.e. false positive samples) are excluded from the calculation. This weighted average replaces $\mathcal{F}_L^{C_t}$ used in the basic scheme.

When some generated adversarial examples are detected, target $\mathcal{F}_L^{C_t}$ is adjusted to move away from them as follows:

$$\begin{aligned} \Xi &\equiv \mathcal{F}_L^{C_t} - \mathbf{E}_{x \in S_{pae}} \mathcal{F}_L(x), \\ \mathcal{F}_L^{C_t} &\leftarrow \mathcal{F}_L^{C_t} + \gamma d_r \frac{\Xi}{\|\Xi\|_2}, \end{aligned} \tag{10}$$

where S_{pae} is the set of positive adversarial examples, d_r is the detection rate of the generated adversarial examples, and γ is a positive weighting parameter (0.1 in our evaluation). $\mathcal{F}_L^{C_t}$ on the right side of Eq. 10 is the weighted average calculated above or the original target used in the basic scheme if the weighted average is not calculated. $\mathbf{E}_{x \in S_{pae}} \mathcal{F}_L(x)$ in Eq. 10 is the average of detected adversarial examples. Negative adversarial examples are not used in Eq. 10 since their average may be on the same side as the average of positive adversarial examples. This modified target will be used in the generation phase. We can see from Eq. 10 that there is no change to $\mathcal{F}_L^{C_t}$ if the detection rate is 0 ($d_r = 0$).

Once the new target $\mathcal{F}_L^{C_t}$ is determined with Eq. 10, we can determine a new constraint boundary c_p on the cosine similarity to include negative examples and exclude positive examples as many as possible. More specifically, we calculate the cosine similarity distributions of both positive examples and negative examples (including benign target samples) queried in this phase with new target $\mathcal{F}_L^{C_t}$. We preset a range of percentiles, $[k_{nl}, k_{nh}]$-th ($[10, 50]$-th in our evaluation) percentiles to specify a permissible range of percentages of negative samples outside the boundary, and a threshold k_p-th (90th in our evaluation) percentile to specify the minimum percentage of positive samples outside the boundary. We find the corresponding values $v_{k_{nl}}$ and $v_{k_{nh}}$ of k_{nl}-th and k_{nh}-th percentiles from the distribution of negative examples and v_{k_p} corresponding to k_p-th percentile from the distribution of the positive examples. The new constraint boundary c_p is determined as follows,

$$c_p = \min(v_{k_{nh}}, \max(v_{k_{nl}}, v_{k_p})). \tag{11}$$

Eq. 11 determines a boundary c_p inside the permissible range $[v_{k_{nl}}, v_{k_{nh}}]$, aiming to exclude at least k_p-th percentile of positive examples. This new c_p will be used to generate adversarial examples in the generation phase.

5.2.2 Generation Phase. In generating adversarial examples, we add the following drive-away loss, ℓ_{away}, to the loss function of the basic scheme (Eq. 9) to minimize the cosine similarity with the positive adversarial examples in the preparation phase:

$$\ell_{away}(\mathcal{F}_L(x)) = \sum_{a \in S_{pae}} \cos(\mathcal{F}_L(x), \mathcal{F}_L(a)) \tag{12}$$

Adding this drive-away loss to Eq. 9, the loss function becomes:

$$\begin{aligned} \ell(\mathcal{F}_L(x), \mathcal{F}_L^{C_t})) = &- \cos(\mathcal{F}_L(x+\epsilon), \mathcal{F}_L^{C_t}) + \lambda_1 \cdot \|\mathcal{F}_L(x) - \mathcal{F}_L^{C_t}\|_2 \\ &+ \lambda_2 \cdot \sum_{a \in S_{pae}} \cos(\mathcal{F}_L(x), \mathcal{F}_L(a)), \end{aligned} \tag{13}$$

where λ_1 and λ_2 are two non-negative weighting parameters. Target $\mathcal{F}_L^{C_t}$ in Eq. 13 is the one calculated with Eq. 10. The same constraint as in Eq. 7, except that c_p now is calculated with Eq. 11, is used with Eq. 13 in generating adversarial examples.

The iteration to crafting an adversarial example is similar to that in the basic scheme. At the beginning, we drive only the first term in Eq. 13 by setting $\lambda_1 = \lambda_2 = 0$ until the constraint is satisfied. Then we activate λ_1 (set to 1 in our experimental evaluation) and λ_2 to drive x into target category C_t and away from positive examples in the feature space. When x falls into target category C_t, λ_1 is deactivated (set to 0). When λ_2 is activated, its value is adapted by multiplying a factor (1.2 in our evaluation) if the first term in Eq. 13 decreases when compared with the last iteration or dividing by another factor (1.3 in our evaluation) if the first increases.

5.3 FIA for Enhanced TeD

Both the basic and complete schemes can be modified slightly to deal with the enhanced version of TeD with randomly sampled neurons and multiple trapdoors. When multiple trapdoors are used, we hope to activate all effective trapdoor signatures[3] in querying the trapdoored defense in the preparation phase. The query described in Section 5.2.1 may not fulfill this goal since feature vectors of generated adversarial examples may be close to each other and thus can activate only a portion of effective trapdoor signatures.

To address this problem, we aim to activate all effective trapdoor signatures during the preparation phase. This can be achieved by applying FIA to query the trapdoored defense multiple rounds until the detection rate is below a threshold or does not decrease, each round with a small number of adversarial examples. More specifically, the first round is executed as before: we use the basic scheme to generate a small number of adversarial examples to query the trapdoored defense. If the detection rate is above the threshold, we

[3]Some trapdoor signatures may be redundant due to being too close to other trapdoor signatures when multiple trapdoors are used.

apply the complete scheme to generate another round of adversarial examples to query the trapdoored defense. Since they are generated by maximizing distances to positive adversarial examples in previous queries (i.e., they try to stay away from trapdoor signatures activated in previous queries), newly generated adversarial examples should activate remaining inactivated trapdoor signatures. This process is repeated until all effective trapdoor signatures have been activated. In each round, the drive-away loss ℓ_{away} in Eq. 12 includes all positive adversarial examples found in previous queries. When all trapdoor signatures have been activated, most generated adversarial examples should be undetectable.

To deal with randomly sampled neurons, the indistinguishability constraint also checks cosine similarities in subsets of randomly sampled neurons to ensure indistinguishability in these subsets too. Since we have no idea which subsets of neurons are used in the trapdoored detection, we just randomly select a group of subsets of neurons at the generation layer and ensure that our generated adversarial examples are indistinguishable in these selected subsets. They are likely indistinguishable in the subsets used in the detection too. For each selected subset, its threshold c_p is determined in the same way as the case using a single trapdoor signature with all neurons.

In addition to the above passive subset checking to ensure indistinguishability on each of randomly selected subsets of neurons at the generation layer, we can also actively drive adversarial examples to reach indistinguishability on each of these subsets. This can be achieved by adding two loss terms for selected subsets. One loss term is the sum of cosine similarity of the adversarial example with the expectation of benign examples in C_t on each selected subset, which we want to maximize. The other loss term is the sum of cosine similarity of the adversarial example with those examples in S_{pae} (i.e., detected adversarial examples during the preparation phase) on each selected subset, which we want to minimize.

6 EXPERIMENTAL EVALUATION

We empirically evaluate the performance of our proposed Feature-Indistinguishable Attack (FIA) against different configurations and variants of the trapdoored defense, including defending single category and all categories, multiple trapdoors and random sampling of neurons, and Projection-based Trapdoor-enabled Detection (P-TeD) described in Section 3.3, etc. Several popular datasets are used in our empirical evaluation. We report the empirical study and its results in this section.

6.1 Experimental Setup

6.1.1 Datasets and DNN models. The same datasets and deep neural networks used in [51] are used in our empirical evaluation. These datasets and neural networks are described in Appendix A.1. We perform handwritten digit recognition with MNIST [35] and image classification with CIFAR10 [32] as they are the most popularly adopted benchmarks. We carry out traffic sign recognition with GTSRB [54] and face recognition with YouTube Face [60] as they stand for two of the most security-critical scenarios (autonomous-driving and biometrics identification) where deep learning applies broadly. The YouTube Face dataset consists of 440K facial images of 224×224 pixels, belonging to 1,283 different people taken from YouTube videos. We use it to evaluate our attack on large-scale datasets. Four convolutional networks including ResNet20 and ResNet50 [25] are used to evaluate our attack on both small and large scale networks.

In our empirical evaluation, training data is used to train a trapdoored model and determine trapdoored defense parameters (e.g., trapdoor signatures), while test data is used in attacking the trapdoored defense. This ensures that defenses and attacks use disjoint data and adversaries have no access to the data used for training a trapdoored model and determining the TeD characteristics.

6.1.2 Trapdoor Settings in Trapdoor-enabled Detection. We evaluate our attack on TeD and P-TeD with different configurations of trapdoors, including single-category (single-label in [51]), where a single trapdoor is injected into a model to protect a specific category, and all categories (all-label in [51]) with both single and multiple trapdoors per category, where multiple trapdoors are injected into a model to protect all categories, with one or multiple trapdoors per category.

Trapdoors are injected with the same parameter settings as in [51]. The detail is presented in Appendix A.2. Under these settings, effective trapdoors can be inserted (each injected trapdoor has a success rate > 97%) with similar accuracy performance as a clean model (accuracy degradation < 2%).

6.1.3 Configurations of Adversarial Attacks. - We evaluate the performance of our proposed attack against TeD and P-TeD with similar experiments as the authors of TeD to evaluate the detection performance of TeD in their paper [51], and compare our attack with existing state-of-the-art adversarial attacks. More specifically, in evaluating our attack against TeD and P-TeD when a single or all categories are protected, we choose two state-of-the-art white-box adversarial attacks, PGD [33, 34] and C&W [10], as baseline attacks. When evaluating our attack against Ted and P-TeD when randomized neuron signatures and multiple trapdoors per category are used, we choose Oracle Signature Attack (OSA) [6] as the baseline attack. OSA is the white-box version of the two attacks [6] specifically designed to circumvent TeD. OSA has a better attacking performance than its grey-box counterpart. The configurations of our proposed attack and the baseline adversarial attacks are summarized in Table 2. Unless stated otherwise, these configurations were used in our experimental evaluation, and the experimental results were obtained by setting the penultimate layer as the detection layer (and the generation layer was also determined to be the penultimate layer) and by setting the false positive rate (FPR) to 5% for benign samples in the target category.

As shown in Table 2, our proposed FIA generally requires a higher bound than that of PGD on a trapdoored model. This is because a conventional adversarial attack like PGD is likely trapped into a shortcut created by the trapdoored defense, and thus requires a bound much lower than the same adversarial attack on a clean model without injecting any trapdoor. For example, it requires $\delta = 64$ in general for PGD to achieve a good attack success rate on a clean MNIST model, but the bound reduces to 8 for PGD to achieve a similar attack success rate on a trapdoored MNIST model. FIA avoids falling into traps in a trapdoored model and thus requires a higher bound, such as 64 on a trapdoored MNIST model. We note that FIA's bound is similar to that of PGD on a clean version (i.e., without the trapdoored defense) of the same DNN model.

Table 2: The configurations of our FIA and the baseline attacks. FIA's four bound values (δ) are for the four datasets listed in Table 11 of Appendix A.1, respectively.

Attack Method	Attack Configuration
PGD	δ=8, n_{iter}=100, η=0.1
C&W	binary step size=9, η=0.05, max iteration=1000, confidence=10.0
Our FIA	δ=64/16/8/16, n_{iter}=5000, η=0.05
OSA	δ=64, n_{iter}=5000, η=0.05

The perceptual quality of adversarial examples crafted with FIA on TeD-protected models is compared with those crafted with PGD on clean models (i.e., without TeD protection) in Appendix A.5. FIA's perceptual quality is the same as or a little better than PGD when the same bound is used.

The above baseline adversarial attacks are chosen in our empirical evaluation for an additional purpose: to verify correctness of our implementation of TeD (and P-TeD) by comparing our attack results with those reported in [51] since the released TeD code [50] is incomplete for conducting our experiments. Our experimental results confirm the effectiveness of the trapdoored defense against existing state-of-the-art L_∞ and L_2 adversarial attacks.

6.2 TeD - Single and All Categories

We first evaluate the attack performances of our FIA and baseline attacks PGD and C&W against TeD when it is used to defend a specific category C_t with a trapdoor (single category) and all categories with one trapdoor per category (all categories). The detection rates of both single category and all categories are reported in Table 3. Their ROC curves are presented in Appendix A.4. For all datasets except YouTube Face, each value in Table 3 is the average over all categories by attacking each category. For YouTube Face, each value in Table 3 is the average over 50 randomly selected categories by attacking each of the selected category.

We can see from the table that the detection rates of FIA are low ($\leq$ 2.0%) except the all categories on the YouTube Face dataset, for which the detection rate is 6.1%. On the other hand, the detection rates of PGD and C&W are generally high: most are above 90%, some are in the range between 70% and 90%, and only two have low detection rates: the detection rate of C&W against the single category is 34.3% on GTSRB, and that of PGD against the all categories is 30.7% on the same dataset. Both values are much higher than the highest detection rate, 6.1%, of FIA reported in the table.

Many detection rates of PGD and C&W reported in Table 3 are significantly lower than those reported in [51]. Our investigation indicates that the results in [51] are likely obtained with FPR calculated with benign samples of categories other than the target category. We present the experimental results with this setting in Appendix A.3, which agree well with those reported in [51].

6.3 P-TeD - Single and All Categories

6.3.1 Detection Rate. The same experiment presented in Section 6.2 is also conducted on P-TeD. The detection rates of the single category and the all categories are reported in Table 4. Their ROC curves are presented in Appendix A.4. From the table, we can see

Table 3: Detection rates at 5% FPR of benign target samples when TeD defends single category and all categories.

	Single Category			All Categories		
	FIA	**PGD**	**C&W**	**FIA**	**PGD**	**C&W**
MNIST	2.0%	100%	99.7%	1.0%	92.8%	98.2%
CIFAR10	0.2%	98.7%	72.7%	0.0%	85.1%	95.2%
GTSRB	0.5%	85.8%	34.3%	0.3%	30.7%	71.8%
YtbFace	1.6%	78.5%	95.1%	6.1%	100%	99.8%

Table 4: Detection rates at 5% FPR of benign target samples when P-TeD defends single category and all categories.

	Single Category			All Categories		
	FIA	**PGD**	**C&W**	**FIA**	**PGD**	**C&W**
MNIST	5.2%	100%	100%	2.5%	100%	99.8%
CIFAR10	0.0%	98.8%	92.2%	0.0%	100%	95.6%
GTSRB	3.1%	97.5%	98.5%	0.4%	96.9%	99.4%
YtbFace	5.9%	94.9%	99.6%	14.3%	100%	99.8%

Table 5: Adversarial generation success rates when P-TeD defends single category and all categories at 5% FPR of benign target samples.

	Single Category			All Categories		
	FIA	**PGD**	**C&W**	**FIA**	**PGD**	**C&W**
MNIST	96.2%	94.9%	100%	99.5%	100%	99.6%
CIFAR10	100%	100%	100%	100%	100%	100%
GTSRB	97.1%	100%	100%	100%	100%	100%
YtbFace	81.6%	91.8%	100%	79.3%	98.9%	100%

that the detection rates of P-TeD are very high (above 94%) for both PGD and C&W attacks on all datasets. These results confirm that the trapdoored defense has a high chance to detect adversarial examples generated with existing state-of-the-art white-box adversarial attacks. On the other hard, the detection rates for FIA are still low, below 5.9% except the all categories on the YouTube Face dataset, for which the detection rate is 14.3%. The experimental results indicate that FIA can effectively circumvent P-TeD too.

6.3.2 Adversarial Generation Success Rate. An adversarial attack may fail in generating an adversarial example at the end of its iterative crafting process. The success rate to generate adversarial examples is also an important metric to measure the performance of an adversarial attack. Table 5 reports the adversarial generation success rates of FIA, PGD, and C&W on P-TeD protecting single category and all categories at 5% FPR of benign target samples.

From Table 5, we can see that the generation success rates of both PGD and C&W are very high on all the tested datasets, above 91% for PGD and above 99% for C&W. FIA's generation success rates are similar to PGD and C&W except on the YouTube Face dataset. On the YouTube Face dataset, FIA has reasonable generation success rates (above 79%), but these rates are significantly lower than their counterparts of both PGD and C&W. This is because the bound of 16 (i.e., $\delta = 16$) used by FIA on the YouTube Face dataset in our

evaluation is a little tight for the dataset. If we relax the bound to 32, FIA has a higher generation success rate and a lower detection rate on the YouTube Face dataset, with both rates similar to their counterparts on the other three tested datasets. This is also true for TeD protecting single category and all categories reported in Table 3 and Section 6.2.

6.4 P-TeD Detection with Randomly Sampled Neurons and Multiple Trapdoors

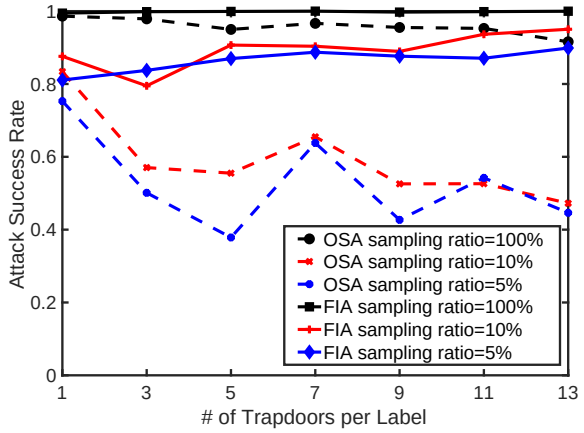

Figure 1: Attack success rates of FIA and OSA when P-TeD protects all categories with 1 to 13 trapdoors per category and with randomly sampled neurons at 100%, 10%, and 5%.

The strongest protection is provided with randomly sampled neurons and multiple trapdoors per category [51]. We conduct the same experiments as reported in [51] to evaluate FIA's performance against this strongest protection and compare it with the white-box attack, Oracle Signature Attack (OSA), the stronger version of the two attacks developed by Carlini [6] to attack the trapdoored defense. In our experiment, the number of trapdoors per category ranged from 1 to 13, and 100%, 10%, and 5% of randomly sampled neurons were used in detection. We set the same FPR for each subset and tried to maintain the total FPR (i.e., the detector's FPR of benign target samples) at 5%. The number of subsets and the FPR for each subset were adjusted to try to make OSA's performance curves similar to those reported in [51]. The same setting was used to evaluate FIA and OSA.

The attack success rates for P-TeD to protect all categories with different configurations are reported in Fig. 1. The resulting total FPR is around 7.0% for each configuration. The results were obtained by using one batch with a size of 16 to generate adversarial examples in each round of queries in the preparation phase. The resulting number of queries used in the preparation phase was small. For example, we used a total of 25.9 and 26.5 queries on average for 100% and 5% randomly sampled neurons, respectively, when 13 trapdoors were injected per category.

From Fig. 1, we can see that both FIA and OSA have high attack success rates when all neurons are used in the trapdoored detection: above 91% for OSA and nearly 100% for FIA. When 10% and 5% randomly sampled neurons are used, FIA maintains high attack success rates ($\geq$ 79.5%), while OSA's attack success rates reduce to 37.85% when the number of trapdoors per category increases to 13 and 5% randomly sampled neurons are used. This result indicates that FIA can effective circumvent the strongest protection provided by the trapdoored defense.

With this strongest trapdoored protection, the generation success rate of FIA is around 40%, much lower than that reported in Table 5 when weaker trapdoored defense is used. A higher bound is needed if a higher generation success rate or a higher attack success rate is required.

6.5 Different Generation Layers

Different latent layers except early layers in the forward pipeline can be used as the generation layer. This is because an adversarial example needs a sequence of layers to transition from the source category to the target category in the feature space. We have conducted experiments to study FIA using different layers as the generation layer. In these experiments, the detection layer was set the same as the generation layer. The second column in Table 6 shows the adversarial attack success rate (ASR) and the adversarial generation success rate (GSR) for P-TeD to protect a single category at 5% FPR of benign target samples on MNIST. Table 12 in Appendix A.1 describes the detail of the network used for MNIST.

From the second column of Table 6, we can see that the ASR is high (above 93%) and does not vary much, while GSR drops significantly from 96.2% to 7.3% when the generation layer moves backwards. The significant drop of ASR can be explained that it becomes increasingly harder to drive to the target when the generation layer moves backwards. A higher bound should be used to achieve a high ASR for an early generation layer, resulting in more noisy adversarial examples. For example, the third column (FIA×2) in Table 6 shows ASR and GSR when the bound is doubled to $\delta = 128$ for MNIST. We can see that the GSR is significantly improved, and GSR drops much slower when the generation layer moves backward. At the same time, ASR has also been improved, to nearly 100% for all the tested layers.

There is an alternative way to improve GSR when the generation layer is not close to the penultimate layer: we can use the basic scheme to drive at the penultimate layer simultaneously with the original driving at the generation layer. The penultimate-layer driving guides an adversarial example into the target category and thus improves GSR. This revised scheme is denoted as FIA+G in Table 6, and the fourth column of Table 6 shows the experimental results on MNIST when the original bound (i.e., $\delta = 64$) was used. We can see that GSR is significantly improved, but ASR drops from 93.6% at the penultimate layer (i.e., dense in Table 6) to 75.3% at conv_2 layer. This behavior is caused by the fact that positive adversarial examples found in the preparation phase deviate from trapdoor signatures much more at a middle layer than at a late layer. The penultimate guidance helps increases GSR, but more generated adversarial examples also spread more in the feature space. Since the constraint is checked at a middle layer (and thus not very accurate) in crafting adversarial examples, this spreading may increase the chance to be detected, resulting in an increased detection rate. This

Table 6: ASR and GSR of FIA using different generation layers and settings (see Section 6.5) on MNIST with single-category P-TeD at 5% FPR of benign target samples.

	ASR / GSR (%)			
$L_{Generation}$	**FIA**	**FIA×2**	**FIA+G**	**FIA+GP**
dense	94.8 / 96.2	100 / 100	93.6 / 96.3	98.6 / 96.1
max_pool_2	93.3 / 64.1	100 / 99.4	82.8 / 96.7	93.2 / 96.5
conv_2	93.9 / 36.3	100 / 90.5	75.3 / 96.7	94.0 / 96.6
max_pool_1	97.3 / 15.1	99.8 / 83.9	85.6 / 94.0	88.5 / 93.9
conv_1	95.3 / 7.3	99.5 / 62.8	83.9 / 91.5	85.5 / 91.7

Table 7: ASR and GSR of FIA with single-category P-TeD at 5% FPR when the generation layer ($L_{Generation}$) mismatches the detection layer ($L_{Detection}$).

Dataset	$L_{Detection}$	$L_{Generation}$	ASR	GSR
MNIST	dense	max_pool_1	95.0%	83.1%
CIFAR10	flatten_1	activation_18	93.8%	87.7%
GTSRB	dense_1	conv_5	99.1%	94.8%
	dense_1	max_pool_2	97.0%	91.0%
	max_pool_3	max_pool_2	88.1%	82.8%

issue can be alleviated by increasing the bound or using a more accurate estimate of trapdoor signatures.

Since PGD can be readily detected by the trapdoored defense, adversarial examples generated with PGD should have a more accurate estimate of trapdoor signatures than adversarial examples with FIA. We can replace the positive adversarial examples used in FIA+G with those generated by PGD (without querying the trapdoored defense since the detection rate is very high for PGD). This FIA variant is denoted as FIA+GP (FIA+G with PGD to generate positive adversarial examples) in Table 6. The rightmost column in Table 6 reports the experimental result for FIA+GP. We can see that ASR improves significantly over FIA+G for the last two layers tested but very small for the two early layers tested. This is because adversarial examples generated with PGD deviate from trapdoor signatures more and more when the generation layer (i.e., the detection layer) moves backward, eventually there is no difference between adversarial examples generated with PGD and FIA in estimating trapdoor signatures.

We note that the trapdoored defense becomes less effective in detecting adversarial examples when the detection layer is significantly away from the penultimate layer.

6.6 Layer Mismatch

In previous experiments, the generation layer and the detection layer are actually the same layer. We have conducted experiments to study FIA's performance when the generate layer mismatches the detection layer. In these experiments, we deliberately let the generation layer before the detection layer. Table 7 shows the experimental results on MNIST, CIFAR10, and GTSRB when P-TeD is used to protect a single category at 5% FPR. We can see from the table that FIA maintains a high ASR (above 88%) and a reasonably high GSR (above 82%) when mismatch occurs.

Table 8: ASR and GSR of FIA on MNIST with P-TeD protecting single category and all categories at 5% FPR when a single batch with different batch sizes is used in the preparation phase. The resulting average number of queries conducted in the preparation phase is reported in the bottom row.

	Single Category			All Categories		
Batch Size	**16**	**32**	**64**	**16**	**32**	**64**
ASR	94.0%	94.8%	95.6%	98.3%	97.5%	97.7%
GSR	96.2%	96.2%	96.3%	99.5%	99.5%	99.5%
# Queries	15.7	30.9	61.7	16.0	31.9	63.6

Table 9: FIA's ASR with and without the drive-away loss when P-TeD protects single category and all categories at 5% FPR.

	With / Without Drive-away Loss (%)	
	Single Category	**All Categories**
MNIST	94.8% / 92.4%	97.5% / 96.8%
GTSRB	96.9% / 82.3%	99.6% / 93.7%
CIFAR10	100% / 100%	100% / 100%

6.7 Ablation Study

We have also conducted an "ablation study" of the impact of hyper-parameters and the drive-away loss on FIA's performance. The parameters that are determined by experimental data, such as target $\mathcal{F}_L^{C_t}$, constraint boundary c_p, etc. are excluded from this study since they cannot be manually set. Only the hyper-parameters that might have a significant impact are reported in the paper. Hyper-parameters not reported can deviate from the empirical values we used in our experimental evaluation with a minor or negligible impact on FIA's performance.

Table 8 shows FIA's performance on MNIST with P-TeD protecting single category and all categories at 5% FPR when a single batch with different batch sizes is used in the preparation phase. The resulting average number of queries in this phase is also reported in the table. From the table, we can see that the batch size has negligible impact on FIA's performance. Similar results are also observed with other datasets. This result indicate that a small batch size can be used in the preparation to reduce the number of queries.

We have also studied the performance when the drive-away loss term is removed. Table 9 reports FIA's ASR with and without the drive-away loss when P-TeD protects single category and all categories at 5% FPR. We can see from the table that the drive-away loss has a small or negligible impact for MNIST and CIFAR10 (at or below +2.4%), but a reasonably significant impact for GTSRB (up to +14.6%). Nevertheless, the ASR without the drive-away loss is at least reasonably high for all the tested datasets.

7 DISCUSSION

7.1 Fortifying TeD?

A question naturally arises: is it possible to fortify TeD to thwart FIA? For the FIA described in Section 5, we can fortify TeD with random trapdoor signatures and detection layers: to protect a target

category, we inject multiple trapdoors and detect at multiple layers, each time using a trapdoor and a detection layer; and we change them from time to time. If a single trapdoor or a single layer is used when FIA queries TeD in the preparation phase, only the activated trapdoor signature or detection layer is explored by FIA. Adversarial examples crafted by FIA may be detected when an unexplored trapdoor or detection layer is used.

The above fortified TeD can be evaded with a fortified FIA scheme by querying TeD repeatedly using the complete FIA scheme until all trapdoor signatures and detection layers are explored, and then FIA applies all positive adversarial examples to drive simultaneously at all layers determined to be potentially used as the detection layer. This is similar to the FIA scheme to attack TeD with randomly sampled neurons and multiple trapdoor signatures per category that is described in Section 5.3.

We have conducted experiments to study the performance with fortified TeD and fortified FIA. Table 10 reports the experimental results on GTSRB when the fortified TeD uses two trapdoors at layer max_pool_3 or two detection layers (dense_1 and max_pool_2) to protect all categories at 5% FPR, with one used in FIA's preparation phase and the other used at detecting crafted adversarial examples. The results with the original TeD and FIA are also reported in Table 10. We can see from the table that the detection rate of the original TeD on the original FIA is 3.8%. The detection rate is boosted to 75.41% when two trapdoors at layer max_pool_3 are used in the fortified TeD: one trapdoor is used at FIA's preparation phase and the other is used at evaluating the detection rate on crafted adversarial examples. The detection rates of the original TeD at layer dense_1 and layer max_pool_2 are 0.44% and 6.11%, respectively. The detection rate is boosted to 99.98% when the fortified TeD uses layer dense_1 at FIA's preparation phase and layer max_pool_2 to evaluate the detection rate with crafted adversarial examples. When the fortified FIA is used to attack the fortified TeD, the detection rate reduces to 6.83% and 8.09% when the fortified TeD uses the two trapdoors or the two detection layers. From the experimental results, we believe that it is almost impossible to fortify the trapdoored defense to effectively detect FIA-like attacks.

Table 10: Detection rates of FIA and fortified FIA on GTSRB with P-TeD and fortified P-TeD (with two trapdoors or two detection layers) protecting all categories at 5% FPR.

Attack	FIA		Fortified-FIA
Defense	**TeD**	**Fortified-TeD**	**Fortified-TeD**
2 Trapdoors	3.80%	75.41%	6.83%
2 Layers	0.44% / 6.11%	99.98%	8.09%

7.2 FIA for Untargeted Adversarial Examples and Future Work

FIA is described under the context of crafting targeted adversarial examples. With minor modifications, it can also be used to craft untargeted adversarial examples. For example, to generate an untargeted adversarial example, we can compare the input with benign samples in different categories in the feature space to find a closest category to drive into. In general, untargeted adversarial examples are easier to craft than targeted adversarial examples. One can always use the aforementioned approach to convert a targeted adversarial attack into an untargeted adversarial attack.

The current FIA relies on an over-simplified assumption of convex region to simplify the FIA scheme. This assumption can be relaxed if we adopt the approach proposed in [30], which uses neural networks to model layer-wise and category-wise feature distributions. This approach should be much more general and powerful than the current FIA version, resulting in more powerful adversarial attacks with better indistinguishability from benign target samples. This will be our future work.

Unlike existing adversarial attacks, our proposed adversarial attack aims to make generated adversarial examples indistinguishable in the feature space from benign samples in the target category. We argue that this is a more powerful approach than existing adversarial attacks. This indistinguishability makes crafted adversarial examples likely much harder to detect than existing adversarial attacks. Our study proves the feasibility of this approach. We expect that FIA should be able to circumvent other powerful methods to detect adversarial examples. This will also be our future work.

With more powerful adversarial attacks like FIA, we need to find an effective defense to thwart them. Developing effective defense against FIA-like attacks will be our future work too.

8 CONCLUSION

The recently proposed *Trapdoor-enabled Detection* (*TeD*) [51] can effectively detect existing state-of-the-art adversarial attacks. To circumvent TeD, we present a novel black-box adversarial attack, called *Feature-Indistinguishable Attack* (*FIA*), which aims to generate adversarial examples indistinguishable in the feature space from benign examples in the target category. It jointly minimizes the distance to the expectation of feature vectors of benign examples in the target category and maximizes distances to positive adversarial examples generated to query TeD in the preparation phase. A constraint is used to ensure that the feature vector of a generated adversarial example is within the distribution of feature vectors of benign examples in the target category. Our extensive empirical evaluation indicates that our proposed FIA can effectively circumvent the strongest defense provided by TeD and its improved variant, *Projection-based Trapdoor-enabled Detection* (*P-TeD*), which is also proposed in this paper. To the best of our knowledge, FIA is the first adversarial attack that aims to craft adversarial examples indistinguishable from benign target examples in the feature space. Adversarial examples crafted with this approach should be much harder to detect. It opens a door for developing much more powerful adversarial attacks. It will also inspire researchers to develop more effective defense against FIA-like attacks.

ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China under Grant No. 61771211, Fundamental Research Funds for the Central Universities under Grant No. 2017KFYXJJ064, the Wuhan Applied Foundational Frontier Project under Grant No. 2020010601012188, and the Guangdong Provincial Key R&D Plan Project under Grant No. 2019B010139001.

REFERENCES

[1] 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, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 274–283. http://proceedings.mlr.press/v80/athalye18a.html

[2] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. 2016. Measuring neural net robustness with constraints. In *30th Conference on Neural Information Processing Systems (NIPS 2016)*.

[3] Avishek Joey Bose and Parham Aarabi. 2018. Adversarial attacks on face detectors using neural net based constrained optimization. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 1–6.

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

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

[6] Nicholas Carlini. 2020. A partial break of the honeypots defense to catch adversarial attacks. *arXiv preprint arXiv:2009.10975* (2020).

[7] Nicholas Carlini and David Wagner. 2016. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311* (2016).

[8] Nicholas Carlini and David 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.

[9] Nicholas Carlini and David Wagner. 2017. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478* (2017).

[10] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 39–57. https://doi.org/10.1109/SP.2017.49

[11] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 10–17. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16893

[12] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. 2018. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 52–68.

[13] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*. PMLR, 1310–1320.

[14] Francesco Croce and Matthias Hein. 2020. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*. PMLR, 2196–2205.

[15] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. 2018. Stochastic activation pruning for robust adversarial defense. (2018).

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *CVPR*.

[17] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. 2019. Evading Defenses to Transferable Adversarial Examples by Translation-Invariant Attacks. In *CVPR*.

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *kdd*, Vol. 96. 226–231.

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

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6572

[21] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[22] Shixiang Gu and Luca Rigazio. 2015. Towards deep neural network architectures robust to adversarial examples. In *International Conference on Learning Representations, (ICLR) Workshop*.

[23] 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. https://doi.org/10.1109/ACCESS.2019.2909068

[24] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. 2018. Countering adversarial images using input transformations. (2018).

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[26] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. 2017. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th USENIX workshop on offensive technologies (WOOT 17)*.

[27] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[28] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. 2016. Learning with a strong adversary. In *International Conference on Learning Representations (ICLR), 2016*.

[29] Nathan Inkawhich, Kevin J Liang, Lawrence Carin, and Yiran Chen. 2020. Transferable perturbations of deep feature distributions. In *International Conference on Learning Representations, ICLR 2020*.

[30] Nathan Inkawhich, Kevin J Liang, Binghui Wang, Matthew Inkawhich, Lawrence Carin, and Yiran Chen. 2020. Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability. *arXiv preprint arXiv:2004.14861* (2020).

[31] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. 2019. Feature space perturbations yield more transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7066–7074.

[32] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[33] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *5th International Conference on Learning Representations ICLR 2017, Workshop track*.

[34] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *5th International Conference on Learning Representations ICLR 2017*.

[35] Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective* (1995), 261–276.

[36] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 656–672.

[37] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. 2019. Certified adversarial robustness with additive noise. In *NeurIPS 2019*.

[38] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*. 446–454.

[39] Shiqing Ma and Yingqi Liu. 2019. Nic: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*.

[40] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. (2018).

[41] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJzIBfZAb

[42] Dongyu Meng and Hao Chen. 2017. MagNet: A Two-Pronged Defense against Adversarial Examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 135–147. https://doi.org/10.1145/3133956.3134057

[43] Jeet Mohapatra, Ching-Yun Ko, Sijia Liu, Pin-Yu Chen, Luca Daniel, et al. 2020. Rethinking randomized smoothing for adversarial robustness. *arXiv preprint arXiv:2003.01249* (2020).

[44] Nina Narodytska and Shiva Prasad Kasiviswanathan. 2017. Simple black-box adversarial perturbations for deep networks. In *CVPR Workshops*.

[45] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

[46] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.

[47] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 582–597.

[48] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-gan: Protecting classifiers against adversarial attacks using generative models. (2018).

[49] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust

optimization. *Neurocomputing* 307 (2018), 195–204.
[50] Shawn Shan. 2021. Gotta Catch'Em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks. https://github.com/Shawn-Shan/trapdoor, note = Accessed on May 1st, 2021. (2021).
[51] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. 2020. Gotta Catch'Em All: Using Honeypots to Catch Adversarial Attacks on Neural Networks. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*. Association for Computing Machinery, New York, NY, USA, 67–83. https://doi.org/10.1145/3372297.3417231
[52] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*. 1528–1540.
[53] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2018. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. (2018).
[54] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks* 32 (2012), 323–332.
[55] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.
[56] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. 2nd International Conference on Learning Representations (ICLR) 2014.
[57] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).
[58] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aäron van den Oord. 2018. Adversarial Risk and the Dangers of Evaluating Against Weak Attacks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 5032–5041. http://proceedings.mlr.press/v80/uesato18a.html
[59] 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 *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 707–723. https://doi.org/10.1109/SP.2019.00031
[60] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*. IEEE, 529–534.
[61] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2018. Mitigating adversarial effects through randomization. (2018).
[62] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving transferability of adversarial examples with input diversity. In *CVPR*.
[63] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-4_Xu_paper.pdf
[64] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 39–49.
[65] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*. 4480–4488.

A APPENDIX

A.1 Datasets and Networks

The datasets and neural networks used in our experimental evaluation are listed in Table 11. The detail of the networks used for MNIST, GTSRB, and CIFAR10 are listed in Tables 12, 13, and 14, respectively. The network for YouTube Face is the standard ResNet50 [25]. It is too long to list here.

A.2 Trapdoor Settings in TeD

In our experimental evaluation, trapdoors are injected with the same parameter settings as in [51]. For all datasets except YouTube Face, when only one trapdoor is injected into a DNN model, the trapdoor trigger is a 6×6 pattern placed at the right bottom corner of an image, and the trapdoor is injected with the injection ratio set to 0.1, i.e., the trapdoored model is trained with the contaminated data taking 10% of the total training data. When multiple trapdoors are injected into a DNN model, a trapdoor trigger consists of five 3×3 small pieces placed randomly across the whole image, and trapdoors are injected with the injection ratio set to 0.5. In both cases, the merge transparency is set to 0.1, which means that the pixel value ratio is $0.1 : 0.9$ for a trapdoor trigger and an input image when we merge the trapdoor with the input image to generate a contaminated image. These trapdooor settings are summarized in Table 15.

The YouTube Face dataset has a much larger image size and many more categories than other datasets. The above trapdoor parameters are slightly adjusted to fit the characteristics of the YouTube Face dataset: a trapdoor is a single 42×42 pattern placed at the right bottom corner or five 21×21 patterns placed randomly

Table 11: Datasets and networks used in our evaluation.

Task	Dataset	Model Arch.
Handwritten digit recognition	MNIST	2 Conv, 2 Dense
Traffic sign recognition	GTSRB	6 Conv, 2 Dense
Image classification	CIFAR10	ResNet20
Face Recognition	YouTube Face	ResNet50

Table 12: The model architecture for MNIST. Size stands for filter size (kernel size $\times$ # of filters) for Conv2D, downsampling size for MaxPooling2D, or # of hidden neurons for Dense layer.

Layer (type)	Size	Activation
conv_1 (Conv2D)	$5 \times 5 \times 16$	ReLU
max_pool_1 (MaxPooling2D)	2×2	-
conv_2(Conv2D)	$5 \times 5 \times 32$	ReLU
max_pool_2 (MaxPooling2D)	2×2	-
dense (Dense)	512	ReLU
dense_1 (Dense)	10	Softmax

Table 13: The model architecture for GTSRB. Size stands for filter size (kernel size $\times$ # of filters) for Conv2D, downsampling size for MaxPooling2D, or # of hidden neurons for Dense layer.

Layer (type)	Size	Activation
conv_1 (Conv2D)	$3 \times 3 \times 32$	ReLU
conv_2 (Conv2D)	$3 \times 3 \times 32$	ReLU
max_pool_1 (MaxPooling2D)	2×2	ReLU
conv_3 (Conv2D)	$3 \times 3 \times 64$	ReLU
conv_4 (Conv2D)	$3 \times 3 \times 64$	ReLU
max_pool_2 (MaxPooling2D)	2×2	ReLU
conv_5 (Conv2D)	$3 \times 3 \times 128$	ReLU
conv_6 (Conv2D)	$3 \times 3 \times 128$	ReLU
max_pool_3 (MaxPooling2D)	2×2	ReLU
dense_1 (Dense)	512	ReLU
dense_2 (Dense)	43	Softmax

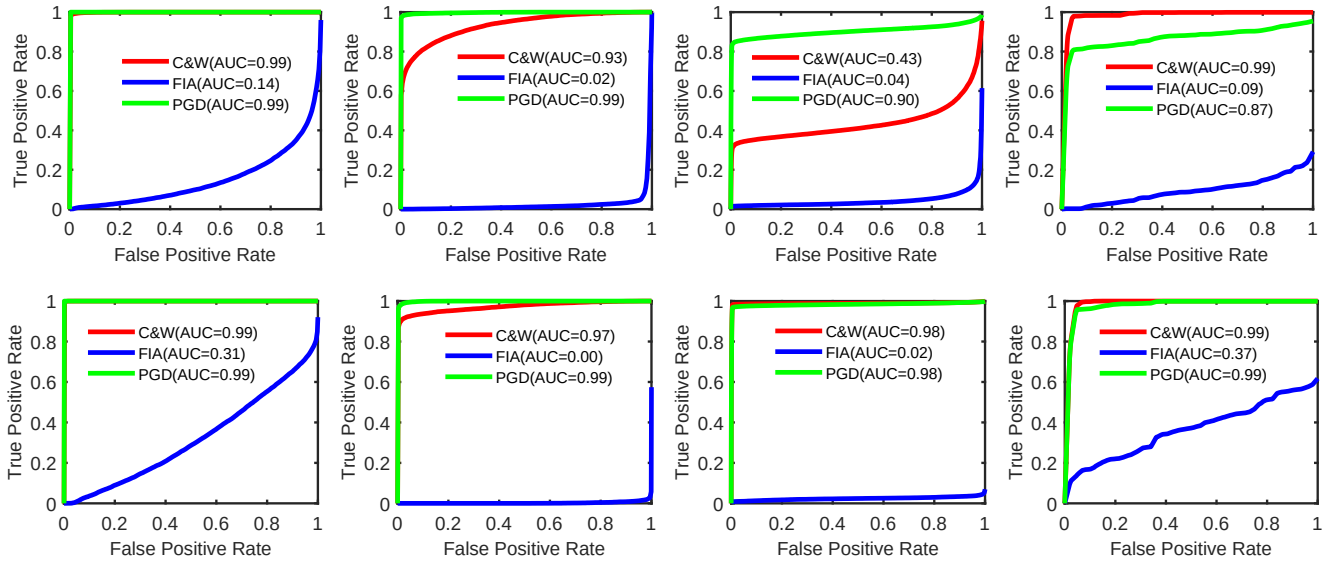

Figure 2: ROC curves for TeD and P-TeD to defend single category. Top row: TeD. Bottom row: P-TeD. Columns from left to right: MNIST, CIFAR10, GTSRB, YouTube Face.

across the whole image, with the injection ratio set to 0.01 or 0.5 when a single trapdoor or multiple trapdoors are injected into a DNN model. In both cases, the merge transparency is set to 0.2.

A.3 TeD - Single & All Categories (5% FPR of Others)

In evaluating the attack performance of baseline attacks PGD and C&W against the trapdoored defense when TeD is used to defend a specific category C_t with a trapdoor (single category) and all categories with one trapdoor per category (all categories), some of obtained detection rates for PGD and C&W are significantly lower than those reported in [51] (see Section 6.2). The authors of TeD don't describe clearly how their false positive rate is calculated in paper [51] when detection threshold ϕ_t is determined. By examining their released code [50], we find out that the released code uses benign samples of categories other than the target category, referred to as *benign non-target samples*, to calculate FPR to determine detection threshold ϕ_t at 5% FPR, which may lead to a high false positive rate for benign target samples, as explained next and conformed by our experimental results to be reported in this subsection. After changing to using benign non-target samples to calculate FPR, we can duplicate the attack performance of PGD and C&W reported in [51].

We believe it is a bug to use benign non-target examples to calculate FPR in their released code [50]. When a trapdoored model is used to protect a specific category C_t, TeD should be used to determine if an input is benign or adversarial only when the input is classified into C_t. It cannot be used to detect an input when it is classified into a category other than C_t. In other words, only samples classified into the category protected by the trapdoored model matters. Since trapdoored samples and benign target samples are all classified into C_t, while benign non-target samples are classified into categories other than C_t, benign non-target samples should have lower cosine similarity values with the trapdoor signature, which is the average of feature vectors of trapdoored samples, than those of benign target samples with the trapdoor signature. This means that detection threshold ϕ_t determined by 5% FPR of benign non-target samples would have a lower value than it should be, leading to many benign target samples falsely detected. As we will see from our experimental results reported in this subsection, all benign target samples are classified as adversarial for GTSRB, resulting in 100% FPR for benign target samples. In this case, FIA cannot craft any adversarial example since it cannot find any negative example in the preparation phase to determine proper target $\mathcal{F}_L^{C_t}$ and boundary c_p to craft adversarial examples.

For completeness, we report the attack performance of our FIA, PGD, and C&W for this setting in this subsection, although such detection setting is useless practically due to its high FPR for benign target samples. In our experimental evaluation for this setting, when the FPR for benign target samples is too high that we cannot find the required number of negative target samples as described in Section 5.2, we select a negative sample as the target in the feature space for FIA to drive to.

Table 16 shows the detection rates of TeD protecting single category and all categories on the four datasets when detection threshold ϕ_t is determined to be 5% FPR of benign non-target samples. We also report the corresponding false positive rates of benign target samples. We note that the detection rate for FIA on GTSRB is marked as N/A (not available) since all benign target samples are detected (i.e. 100% false positive rate), we cannot find any negative sample for FIA to drive to.

From Table 16, we can see that the trapdoored defense has a very high detection rate for both PGD and C&W, which agrees with the results reported in [51], but a very low detection rate for adversarial

Table 14: The model architecture of ResNet20 for CIFAR10. ResNet20 consists of residual blocks, where Conv2D layers are skip-connected with Add-layer.

Layer (type)	Activation	Connected to
conv_1 (Conv2D)	ReLU	-
conv_2 (Conv2D)	ReLU	conv_1
conv_3 (Conv2D)	-	conv_2
add_1 (Add)	ReLU	conv_3, conv_1
conv_4 (Conv2D)	ReLU	add_1
conv_5 (Conv2D)	-	conv_4
add_2 (Add)	ReLU	conv_5, add_1
conv_6 (Conv2D)	ReLU	add_2
conv_7 (Conv2D)	-	conv_6
add_3 (Add)	ReLU	conv_7, add_2
conv_8 (Conv2D)	ReLU	add_3
conv_9 (Conv2D)	-	conv_8
conv_10 (Conv2D)	-	add_3
add_4 (Add)	ReLU	conv_9, conv_10
conv_11 (Conv2D)	ReLU	add_4
conv_12 (Conv2D)	-	conv_11
add_5 (Add)	ReLU	conv_12, add_4
conv_13 (Conv2D)	ReLU	add_5
conv_14 (Conv2D)	-	conv_13
add_6 (Add)	ReLU	conv_14, add_5
conv_15 (Conv2D)	ReLU	add_6
conv_16 (Conv2D)	-	conv_15
conv_17 (Conv2D)	-	add_6
add_7 (Add)	ReLU	conv_16, conv_17
conv_18 (Conv2D)	ReLU	add_7
conv_19 (Conv2D)	-	conv_18
add_8 (Add)	ReLU	conv_19, add_7
conv_20 (Conv2D)	ReLU	add_8
conv_21 (Conv2D)	-	conv_20
add_9 (Add)	ReLU	conv_21, add_8
avg_pool (AveragePooling2D)	-	add_9
dense_1 (Dense)	Softmax	avg_pooling

Table 15: Trapdoor settings in TeD. Trans. and I.R. stand for merge transparency and injection ratio of a trapdoor, respectively. A trapdoor trigger is expressed as $n \times w \times h$, standing for the number of pieces, and the weight and height of each piece.

Defense Settings	Trigger	Trans.	I.R.
Single category single trapdoor	$1 \times 6 \times 6$	0.1	0.1
All categories single trapdoor	$5 \times 3 \times 3$	0.1	0.5
All categories multiple trapdoors	$5 \times 3 \times 3$	0.1	0.5

examples crafted with our proposed FIA. At the same time, the FPR of benign target samples is generally high. For GTSRB, this FPR is 100%, making FIA unable to craft any adversarial example since it cannot find a target to drive to.

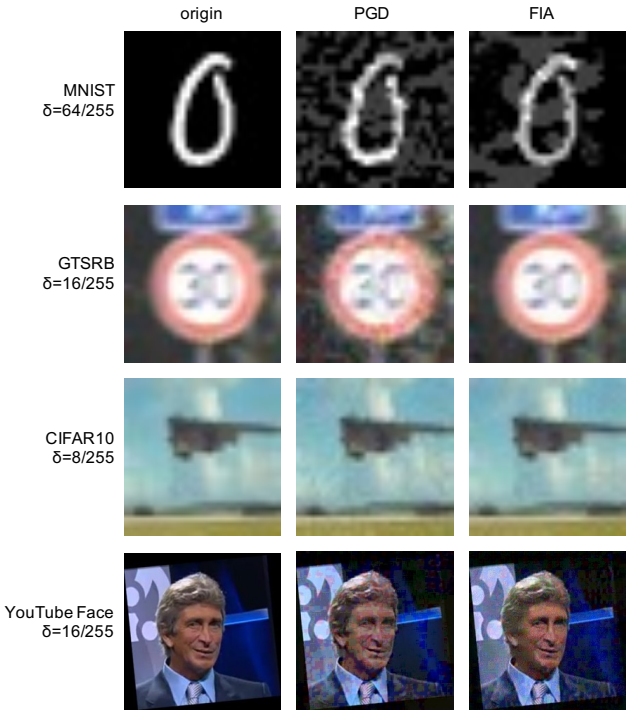

Figure 3: Comparison of perceptual quality of adversarial examples crafted with FIA on TeD-protected models and PGD on clean models using the same bound.

Table 16: Detection rates of TeD protecting both single category and all categories with 5% FPR of benign non-target samples and the corresponding FPR of benign samples in the target category (target FPR).

	Single Category / All Categories (%)			
	FIA	**PGD**	**C&W**	**Target FPR**
MNIST	2.8 / 3.0	100 / 98.9	100 / 99.4	51.7 / 27.5
CIFAR10	0.0 / 0.0	100 / 100	99.5 / 99.8	82.9 / 70.7
GTSRB	N/A	100 / 100	100 / 100	100 / 100
YtbFace	6.0 / 11.2	96.4 / 100	99.9 / 99.7	70.5 / 32.2

A.4 ROC Curves of TeD and P-TeD

To further compare detection performance of TeD and P-TeD on FIA, PGD and C&W, we report the ROC curves and AUC scores for TeD and P-TeD to defend single category in Fig. 2. From this figure, we can see that P-TeD has very high AUC scores for both PGD and C&W, at or above 0.97, on all the tested datasets, while TeD has very high AUC scores for both PGD and C&W on MNIST and CIFAR10, at or above 0.93. The AUC score of TeD for C&W is 0.43 on GTSRB, much lower than its AUC scores on other datasets. Further investigation reveals the cause: the cosine similarity distribution of adversarial examples with the trapdoor signature and that of benign target samples with the trapdoor signature in this case overlaps much more than other datasets. The detection performance of C&W

on GTSRB is significantly boosted when the projected signature is used: the AUC score of C&W on GTSRB increases to 0.98 for P-TeD.

On the other hand, Fig. 2 shows that the AUC scores of both TeD and P-TeD are low for FIA, all at or below 0.37. FIA's AUC scores on MNIST and YouTube Face are significantly higher than the other two datasets, indicating that it is more difficult for FIA to craft adversarial examples on MNIST and YouTube Face than the other two datasets. If we relax their bounds, FIA's attack performance on these two datasets can be significantly improved, at the cost of more noisy adversarial examples. For example, if the bound is increased from 16 to 32 for YouTube Face, the AUC scores reduce to 0.01 for TeD and 0.02 for P-TeD.

A.5 Perceptual Qualify of Adversarial Examples Crafted with FIA

Fig. 3 shows adversarial examples crafted with FIA on TeD-protected models and with PGD on clean models (i.e., without TeD protection) for different datasets. The same bound is used for FIA and PGD on a dataset. We can see from the figure that adversarial examples crafted with FIA look a little better than those crafted with PGD.