# A Tale of Evil Twins: Adversarial Inputs versus Poisoned Models

Ren Pang[1] Hua Shen[1] Xinyang Zhang[1] Shouling Ji[2,3] Yevgeniy Vorobeychik[4]
Xiapu Luo[5] Alex Liu[3] Ting Wang[1]
[1]Pennsylvania State University [2]Zhejiang University [3]Ant Financial
[4]Washington University in St. Louis [5]Hong Kong Polytechnic University

## Abstract

Despite their tremendous success in a range of domains, deep learning systems are inherently susceptible to two types of manipulations: adversarial inputs – maliciously crafted samples that deceive target deep neural network (DNN) models, and poisoned models – adversely forged DNNs that misbehave on pre-defined inputs. While prior work has intensively studied the two attack vectors in parallel, there is still a lack of understanding about their fundamental connections: what are the dynamic interactions between the two attack vectors? what are the implications of such interactions for optimizing existing attacks? what are the potential countermeasures against the enhanced attacks? Answering these key questions is crucial for assessing and mitigating the holistic vulnerabilities of DNNs deployed in realistic settings.

Here we take a solid step towards this goal by conducting the first systematic study of the two attack vectors within a unified framework. Specifically, (i) we develop a new attack model that jointly optimizes adversarial inputs and poisoned models; (ii) with both analytical and empirical evidence, we reveal that there exist intriguing "mutual reinforcement" effects between the two attack vectors – leveraging one vector significantly amplifies the effectiveness of the other; (iii) we demonstrate that such effects enable a large design spectrum for the adversary to enhance the existing attacks that exploit both vectors (e.g., backdoor attacks), such as maximizing the attack evasiveness with respect to various detection methods; (iv) finally, we discuss potential countermeasures against such optimized attacks and their technical challenges, pointing to several promising research directions.

## 1 INTRODUCTION

The abrupt advances in deep learning have led to breakthroughs in a number of long-standing machine learning tasks (e.g., image classification [14], natural language processing [42], and even playing Go [45]), enabling scenarios previously considered strictly experimental. However, it is now well known that deep learning systems are inherently vulnerable to adversarial manipulations, which significantly hinders their use in security-critical domains, such as autonomous driving, video surveillance, web content filtering, and biometric authentication.

Two primary attack vectors have been considered in the literature. (i) Adversarial inputs – typically through perturbing a benign input $x$, the adversary crafts an adversarial version $x_*$ which deceives the target DNN $f$ at inference time [7, 20, 40, 48]. (ii) Poisoned models – during training, the adversary builds malicious functions into $f$, such that the poisoned DNN $f_*$ misbehaves on one (or more) pre-defined input(s) $x$ [24, 25, 44, 47]. As illustrated in Figure 1, the two attack vectors share the same aim of forcing the DNN to misbehave on pre-defined inputs, yet through different routes: one perturbs the input and the other modifies the model. There are attacks (e.g., backdoor attacks [21, 32]) that leverage the two attack vectors simultaneously: the adversary modifies $f$ to be sensitive to pre-defined trigger patterns (e.g., specific watermarks) during training and then generates trigger-embedded inputs at inference time to cause the poisoned model $f_*$ to malfunction.

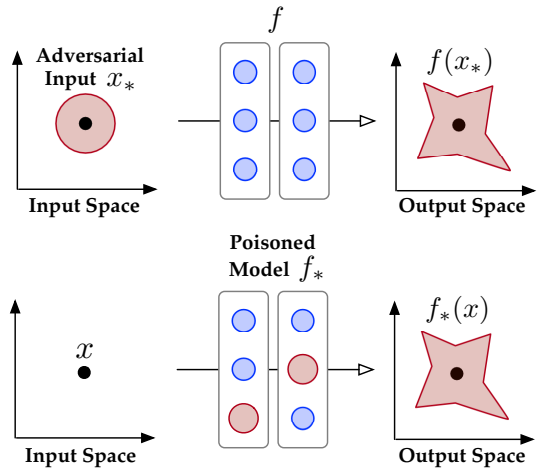


**Figure 1: "Duality" of adversarial inputs and poisoned models.**

Prior work has intensively studied the two attack vectors separately [7, 20, 24, 25, 40, 44, 47, 48]; yet, there is still a lack of understanding about their fundamental connections. First, it remains unclear what the vulnerability to one attack implies for the other. Revealing such implications is important for developing effective defenses against both attacks. Further, the adversary may exploit the two vectors together (e.g., backdoor attacks [21, 32]), or multiple adversaries may collude to perform coordinated attacks. It is unclear how the two vectors may interact with each other and how their interactions may influence the attack dynamics. Understanding such interactions is critical for building effective defenses against coordinated attacks. Finally, studying the two attack vectors within a unified framework is essential for assessing and mitigating the holistic vulnerabilities of DNNs deployed in practice, in which multiple attacks may be launched simultaneously.

More specifically, in this paper, we seek to answer the following research questions.

- RQ1 – *What are the fundamental connections between adversarial inputs and poisoned models?*

- RQ2 – *What are the dynamic interactions between the two attack vectors if they are applied together?*
- RQ3 – *What are the implications of such interactions for the adversary to optimize the attack strategies?*
- RQ4 – *What are the potential countermeasures to defend against such enhanced attacks?*

**Our Work.** This work represents a solid step towards answering the key questions above. We cast adversarial inputs and poisoned models within a unified framework, conduct a systematic study of their interactions, and reveal the implications for DNNs' holistic vulnerabilities, leading to the following interesting findings.

RA1 – We develop a new attack model that jointly optimizes adversarial inputs and poisoned models. With this framework, we show that there exists an intricate "duality" relationship between the two attack vectors. Specifically, they represent different routes to achieve the same aim (i.e., misclassification of the target input): one perturbs the input at the cost of "fidelity" (whether the attack retains the original input's perceptual quality), while the other modifies the DNN at the cost of "specificity" (whether the attack influences non-target inputs).

RA2 – Through empirical studies on benchmark datasets and in security-critical applications (e.g., skin cancer screening [16]), we reveal that the interactions between the two attack vectors demonstrate intriguing "mutual-reinforcement" effects: when launching the unified attack, leveraging one attack vector significantly amplifies the effectiveness of the other (i.e., "the whole is much greater than the sum of its parts"). We also provide analytical justification for such effects under a simplified setting.

RA3 – Further, we demonstrate that the mutual reinforcement effects entail a large design spectrum for the adversary to optimize the existing attacks that exploit both attack vectors (e.g., backdoor attacks). For instance, leveraging such effects, it is possible to enhance the attack evasiveness with respect to multiple defense mechanisms (e.g., adversarial training [34]), which are designed to defend against adversarial inputs or poisoned models alone; it is also possible to enhance the existing backdoor attacks (e.g., [21, 32]) with respect to both human vision (in terms of trigger size and transparency) and automated detection methods (in terms of input and model anomaly).

RA4 – Finally, we demonstrate that to effectively defend against such optimized attacks, it is necessary to investigate the attacks from multiple complementary perspectives (i.e., fidelity and specificity) and carefully account for the mutual reinforcement effects in applying the mitigation solutions, which point to a few promising research directions.

To our best knowledge, this work represents the first systematic study of adversarial inputs and poisoned models within a unified framework. We believe our findings deepen the holistic understanding about the vulnerabilities of DNNs in practical settings and shed light on developing more effective countermeasures.[1]

[1]The source code and data are released at https://github.com/alps-lab/imc.

| Notation | Definition |
|---|---|
| $x_\circ$, $x_*$ | benign, adversarial inputs |
| $\theta_\circ$, $\theta_*$ | benign, poisoned DNNs |
| $t$ | adversary's target class |
| $\kappa$ | misclassification confidence threshold |
| $\mathcal{D}$, $\mathcal{R}$, $\mathcal{T}$ | training, reference, target sets |
| $\ell$, $\ell_\mathrm{s}$, $\ell_\mathrm{f}$ | attack efficacy, specificity, fidelity losses |
| $\phi$ | leverage effect coefficient |
| $\alpha$ | learning rate |
| $\epsilon$, $\delta$ | thresholds of input, model perturbation |

**Table 1. Symbols and notations.**

**Roadmap.** The remainder of the paper proceeds as follows. § 2 introduces fundamental concepts; § 3 presents a new attack framework that unifies adversarial inputs and poisoned models; § 4 reveals the inherent connections between the two attack vectors; § 5 studies the implications of such connections for optimizing existing attacks and discusses potential countermeasures; § 6 surveys relevant literature; the paper is concluded in § 7.

## 2 PRELIMINARIES

We begin by introducing a set of fundamental concepts and assumptions. Table 1 summarizes the important notations in the paper.

### 2.1 Deep Neural Networks

Deep neural networks (DNNs) represent a class of machine learning models to learn high-level abstractions of complex data using multiple processing layers in conjunction with non-linear transformations. We primarily consider a predictive setting, in which a DNN $f$ (parameterized by $\theta$) encodes a function $f\colon \mathcal{X} \to \mathcal{Y}$. Given an input $x \in \mathcal{X}$, $f$ predicts a nominal variable $f(x;\theta)$ ranging over a set of pre-defined classes $\mathcal{Y}$.

We consider DNNs obtained via supervised learning. To train a model $f$, the training algorithm uses a training set $\mathcal{D}$, of which each instance $(x, y) \in \mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$ comprises an input $x$ and its ground-truth class $y$. The algorithm determines the best parameter configuration $\theta$ for $f$ via optimizing a loss function $\ell(f(x;\theta), y)$ (e.g., the cross entropy of $y$ and $f(x;\theta)$), which is typically implemented using stochastic gradient descent or its variants [55].

### 2.2 Attack Vectors

DNNs are inherently susceptible to malicious manipulations. In particular, two primary attack vectors have been considered in the literature, namely, adversarial inputs and poisoned models.

**Adversarial Inputs.** Adversarial inputs are maliciously crafted samples to deceive target DNNs at inference time. An adversarial input $x_*$ is typically generated by perturbing a benign input $x_\circ$ to change its classification to a target class $t$ desired by the adversary (e.g., pixel perturbation [34] or spatial transformation [1]). To ensure the attack evasiveness, the perturbation is often constrained to a *feasible set* (e.g., a norm ball $\mathcal{F}_\epsilon(x_\circ) = \{x \,|\, \|x - x_\circ\|_\infty \leq \epsilon\}$). Formally, the attack is formulated as the optimization objective:

$$x_* = \arg \min_{x \in \mathcal{F}_\epsilon(x_\circ)} \ell(x, t; \theta_\circ) \tag{1}$$

where the loss function measures the difference between $f$'s prediction $f(x;\theta_\circ)$ and the adversary's desired classification $t$.

Eqn (1) can be solved in many ways. For instance, FGSM [20] uses one-step descent along $\ell$'s gradient sign direction, PGD [34] applies a sequence of projected gradient descent steps, while C&W [7] solves Eqn (1) with iterative optimization.

**Poisoned Models.** Poisoned models are adversely forged DNNs that are embedded with malicious functions (i.e., misclassification of target inputs) during training.

This attack can be formulated as perturbing a benign DNN $\theta_\circ$ to a poisoned version $\theta_*$.[2] To ensure its evasiveness, the perturbation is often constrained to a feasible set $\mathcal{F}_\delta(\theta_\circ)$ to limit the impact on non-target inputs. For instance, $\mathcal{F}_\delta(\theta_\circ) = \{\theta | \mathbb{E}_{x\in\mathcal{R}}[|f(x;\theta) - f(x;\theta_\circ)|] \leq \delta\}$ specifies that the expected difference between $\theta_\circ$ and $\theta_*$'s predictions regarding the inputs in a reference set $\mathcal{R}$ stays below a threshold $\delta$. Formally, the adversary attempts to optimize the objective function:

$$\theta_* = \arg \min_{\theta\in\mathcal{F}_\delta(\theta_\circ)} \mathbb{E}_{x_\circ\in\mathcal{T}}\left[\ell(x_\circ, t_{x_\circ}; \theta)\right] \tag{2}$$

where $\mathcal{T}$ represents the set of target inputs, $t_{x_\circ}$ denotes $x_\circ$'s classification desired by the adversary, and the loss function is defined similarly as in Eqn (1).

In practice, Eqn (2) can be solved through either polluting training data [21, 44, 47] or modifying benign DNNs [24, 32]. For instance, StingRay [47] generates poisoning data by perturbing benign inputs close to $x_\circ$ in the feature space; PoisonFrog [44] synthesizes poisoning data close to $x_\circ$ in the feature space but perceptually belonging to $t$ in the input space; while ModelReuse [24] directly perturbs the DNN parameters to minimize $x_\circ$'s distance to a representative input from $t$ in the feature space.

## 2.3 Threat Models

We assume a threat model wherein the adversary is able to exploit both attack vectors. During training, the adversary forges a DNN embedded with malicious functions. This poisoned model is then incorporated into the target deep learning system through either system development or maintenance [24, 32]. At inference time, the adversary further generates adversarial inputs to trigger the target system to malfunction.

This threat model is realistic. Due to the increasing model complexity and training cost, it becomes not only tempting but also necessary to reuse pre-trained models [21, 24, 32]. Besides reputable sources (e.g., Google), most pre-trained DNNs on the market (e.g., [6]) are provided by untrusted third parties. Given the widespread use of deep learning in security-critical domains, adversaries are strongly incentivized to build poisoned models, lure users to reuse them, and trigger malicious functions via adversarial inputs during system use. The backdoor attacks [21, 32, 54] are concrete instances of this threat model: the adversary makes DNN sensitive to certain trigger patterns (e.g., watermarks), so that any trigger-embedded inputs are misclassified at inference. Conceptually, one may regard the trigger as a universal perturbation $r$ [36]. To train the poisoned model $\theta_*$, the adversary samples inputs $\mathcal{T}$ from the training set $\mathcal{D}$ and enforces the trigger-embedded input $(x_\circ + r)$ for each $x_\circ \in \mathcal{T}$ to be misclassified to the target class $t$. Formally, the adversary optimizes the objective function:

$$\min_{r\in\mathcal{F}_\epsilon, \theta\in\mathcal{F}_\delta(\theta_\circ)} \mathbb{E}_{x_\circ\in\mathcal{T}}\left[\ell(x_\circ + r, t; \theta)\right] \tag{3}$$

where both the trigger and poisoned model need to satisfy the evasiveness constraints. Nonetheless, in the existing backdoor attacks, Eqn (3) is often solved in an ad hoc manner, resulting in suboptimal triggers and/or poisoned models. For example, TrojanNN [32] pre-defines the trigger shape (e.g., Apple logo) and determines its pixel values in a preprocessing step. We show that the existing attacks can be significantly enhanced within a rigorous optimization framework (details in § 5).

# 3 A UNIFIED ATTACK FRAMEWORK

Despite their apparent variations, adversarial inputs and poisoned models share the same objective of forcing target DNNs (modified or not) to misclassify pre-defined inputs (perturbed or not). While intensive research has been conducted on the two attack vectors in parallel, little is known about their fundamental connections.

## 3.1 Attack Objectives

To bridge this gap, we study the two attack vectors using *input model co-optimization* (IMC), a unified attack framework. Intuitively, within IMC, the adversary is allowed to perturb each target input $x_\circ \in \mathcal{T}$ and/or to poison the original DNN $\theta_\circ$, with the objective of forcing the adversarial version $x_*$ of each $x_\circ \in \mathcal{T}$ to be misclassified to a target class $t_{x_\circ}$ by the poisoned model $\theta_*$.

Formally, we define the unified attack model by integrating the objectives of Eqn (1), Eqn (2), and Eqn (3):

$$\boxed{\min_{\theta\in\mathcal{F}_\delta(\theta_\circ)} \mathbb{E}_{x_\circ\in\mathcal{T}}\left[\min_{x\in\mathcal{F}_\epsilon(x_\circ)} \ell\left(x, t_{x_\circ}; \theta\right)\right]} \tag{4}$$

where the different terms define the adversary's multiple desiderata:

- The loss $\ell$ quantifies the difference of the model prediction and the classification desired by the adversary, which represents the attack *efficacy* – whether the attack successfully forces the DNN to misclassify each input $x_\circ \in \mathcal{T}$ to its target class $t_{x_\circ}$.
- The constraint $\mathcal{F}_\epsilon$ bounds the impact of input perturbation on each target input, which represents the attack *fidelity* – whether the attack retains the perceptual similarity of each adversarial input to its benign counterpart.
- The constraint $\mathcal{F}_\delta$ bounds the influence of model perturbation on non-target inputs, which represents the attack *specificity* – whether the attack precisely directs its influence to the set of target inputs $\mathcal{T}$ only.

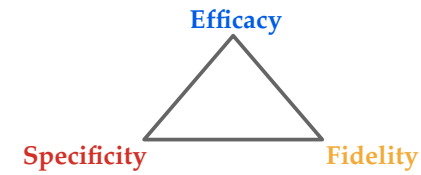


**Figure 2: Adversary's multiple objectives.**

This formulation subsumes many attacks in the literature. Specifically, (i) in the case of $\delta = 0$ and $|\mathcal{T}| = 1$, Eqn (4) is instantiated as

[2] Note that below we use $\theta_\circ$ ($\theta_*$) to denote both a DNN and its parameter configuration. Also note that the benign model $\theta_\circ$ is independent of the target benign input $x_\circ$.

the adversarial attack; (ii) in the case of $\epsilon = 0$, Eqn (4) is instantiated as the poisoning attack, which can be either a single target ($|\mathcal{T}| = 1$) or multiple targets ($|\mathcal{T}| > 1$); and (iii) in the case that a universal perturbation ($x - x_\circ$) is defined for all the inputs $\{x_\circ \in \mathcal{T}\}$ and all the target classes $\{t_{x_\circ}\}$ are fixed as $t$, Eqn (4) is instantiated as the backdoor attack. Also note that this formulation does not make any assumptions regarding the adversary's capability or resource (e.g., access to the training or inference data), while it is solely defined in terms of the adversary's objectives.

Interestingly, the three objectives are tightly intertwined, forming a triangle structure, as illustrated in Figure 2. We have the following observations.

- It is impossible to achieve all the objectives simultaneously. To attain attack efficacy (i.e., launching a successful attack), it requires either perturbing the input (i.e., at the cost of fidelity) or modifying the model (i.e., at the cost of specificity).
- It is feasible to attain two out of the three objectives at the same time. For instance, it is trivial to achieve both attack efficacy and fidelity by setting $\epsilon = 0$ (i.e., only model perturbation is allowed).
- With one objective fixed, it is possible to balance the other two. For instance, with fixed attack efficacy, it allows to trade between attack fidelity and specificity.

Next, by casting the attack vectors of adversarial inputs and poisoned models within the IMC framework, we reveal their inherent connections and explore the dynamic interactions among the attack efficacy, fidelity, and specificity.

## 3.2 Attack Implementation

Recall that IMC is formulated in Eqn (4) as optimizing the objectives over both the input and model. While it is impractical to exactly solve Eqn (4) due to its non-convexity and non-linearity, we reformulate Eqn (4) to make it amenable for optimization. To ease the discussion, in the following, we assume the case of a single target input $x_\circ$ in the target set $\mathcal{T}$ (i.e., $|\mathcal{T}| = 1$), while the generalization to multiple targets is straightforward. Further, when the context is clear, we omit the reference input $x_\circ$, benign model $\theta_\circ$, and target class $t$ to simplify the notations.

*3.2.1* **Reformulation.** The constraints $\mathcal{F}_\epsilon(x_\circ)$ and $\mathcal{F}_\delta(\theta_\circ)$ in Eqn (4) essentially bound the fidelity and specificity losses. The fidelity loss $\ell_\mathrm{f}(x)$ quantifies whether the perturbed input $x$ faithfully retains its perceptual similarity to its benign counterpart $x_\circ$ (e.g., $\|x - x_\circ\|$); the specificity loss $\ell_\mathrm{s}(\theta)$ quantifies whether the attack impacts non-target inputs (e.g., $\mathbb{E}_{x \in \mathcal{R}}\left[|f(x;\theta) - f(x;\theta_\circ)|\right]$). According to optimization theory [5], specifying the bounds $\epsilon$ and $\delta$ on the input and model perturbation amounts to specifying the hyper-parameters $\lambda$ and $\nu$ on the fidelity and specificity losses (the adversary is able to balance different objectives by controlling $\lambda$ and $\nu$). Eqn (4) can therefore be re-formulated as follows:

$$\min_{x,\theta} \ell(x;\theta) + \lambda \ell_\mathrm{f}(x) + \nu \ell_\mathrm{s}(\theta) \tag{5}$$

Nonetheless, it is still difficult to directly optimize Eqn (5) given that the input $x$ and the model $\theta$ are mutually dependent on each other. Note that however $\ell_\mathrm{f}$ does not depend on $\theta$ while $\ell_\mathrm{s}$ does not depend on $x$. We thus further approximate Eqn (5) with the following bi-optimization formulation:

$$\begin{cases} x_* = \arg\min_x \ell(x;\theta_*) + \lambda \ell_\mathrm{f}(x) \\ \theta_* = \arg\min_\theta \ell(x_*;\theta) + \nu \ell_\mathrm{s}(\theta) \end{cases} \tag{6}$$

*3.2.2* **Optimization.** This formulation naturally leads to an optimization procedure that alternates between updating the input $x$ and updating the model $\theta$, as illustrated in Figure 3. Specifically, let $x^{(k)}$ and $\theta^{(k)}$ be the perturbed input and model respectively after the $k$-th iteration. The $(k+1)$-th iteration comprises two operations.

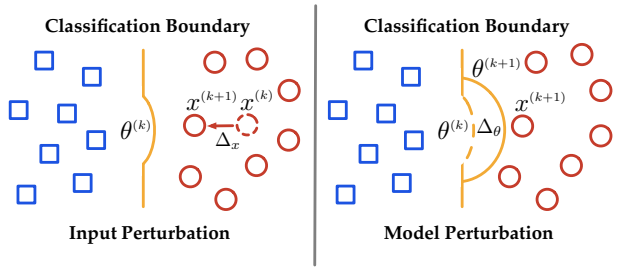


**Figure 3: IMC alternates between two operations: (i) input perturbation to update the adversarial input $x$, and (ii) model perturbation to update the poisoned model $\theta$.**

*Input Perturbation* – In this step, with the model $\theta^{(k)}$ fixed, it updates the perturbed input by optimizing the objective:

$$x^{(k+1)} = \arg\min_x \ \ell(x;\theta^{(k)}) + \lambda \ell_\mathrm{f}(x) \tag{7}$$

In practice, this step can be approximated by applying an off-the-shelf optimizer (e.g., Adam [26]) or solved partially by applying gradient descent on the objective function. For instance, in our implementation, we apply projected gradient descent (PGD [34]) as the update operation:

$$x^{(k+1)} = \Pi_{\mathcal{F}_\epsilon(x_\circ)}\left(x^{(k)} - \alpha \operatorname{sgn}\left(\nabla_x \ell\left(x^{(k)};\theta^{(k)}\right)\right)\right)$$

where $\Pi$ is the projection operator, $\mathcal{F}_\epsilon(x_\circ)$ is the feasible set (i.e., $\{x | \|x - x_\circ\| \le \epsilon\}$), and $\alpha$ is the learning rate.

*Model Perturbation* – In this step, with the input $x^{(k+1)}$ fixed, it searches for the model perturbation by optimizing the objective:

$$\theta^{(k+1)} = \arg\min_\theta \ \ell(x^{(k+1)};\theta) + \nu \ell_\mathrm{s}(\theta) \tag{8}$$

In practice, this step can be approximated by running re-training over a training set that mixes the original training data $\mathcal{D}$ and $m$ copies of the current adversarial input $x^{(k+1)}$. In our implementation, $m$ is set to be half of the batch size.

Algorithm 1 sketches the complete procedure. By alternating between input and model perturbation, it finds approximately optimal adversarial input $x_*$ and poisoned model $\theta_*$. Note that designed to study the interactions of adversarial inputs and poisoned models (§ 4), Algorithm 1 is only one possible implementation of Eqn (4) under the setting of a single target input and both input and model perturbation. To implement other attack variants, one can adjust Algorithm 1 accordingly (§ 5). Also note that it is possible to perform multiple input (or model) updates per model (or input) update to accommodate their different convergence rates.

**Algorithm 1:** IMC Attack

**Input:** benign input – $x_\circ$; benign model – $\theta_\circ$; target class – $t$; hyper-parameters – $\lambda$, $\nu$
**Output:** adversarial input – $x_*$; poisoned model – $\theta_*$

// initialization
$x^{(0)}, \theta^{(0)}, k \leftarrow x_\circ, \theta_\circ, 0$;
// optimization
**while** *not converged yet* **do**
  // input perturbation
  $x^{(k+1)} = \arg\min_x \ell(x; \theta^{(k)}) + \lambda \ell_{\mathrm{f}}(x)$;
  // model perturbation
  $\theta^{(k+1)} = \arg\min_\theta \ell(x^{(k+1)}; \theta) + \nu \ell_{\mathrm{s}}(\theta)$;
  $k \leftarrow k + 1$;
**return** $(x^{(k)}, \theta^{(k)})$;

*3.2.3* **Analysis.** Next we provide analytical justification for Algorithm 1. As Eqn (5) is effectively equivalent to Eqn (4), Algorithm 1 approximately solves Eqn (5) by alternating between (i) input perturbation – searching for $x_* = \arg\min_{x \in \mathcal{F}_\epsilon(x_\circ)} \ell(x; \theta_*)$ and (ii) model perturbation – searching for $\theta_* = \arg\min_{\theta \in \mathcal{F}_\delta(\theta_\circ)} \ell(x_*; \theta)$. We now show that this implementation effectively solves Eqn (5) (proof deferred to Appendix A).

**Proposition 1.** Let $x_* \in \mathcal{F}_\epsilon(x_\circ)$ be a minimizer of the function $\min_x \ell(x; \theta)$. If $x_*$ is non-zero, then $\nabla_\theta \ell(x_*; \theta)$ is a proper descent direction for the objective function of $\min_{x \in \mathcal{F}_\epsilon(x_\circ)} \ell(x; \theta)$.

Thus, we can conclude that Algorithm 1 is an effective implementation of the IMC attack framework. It is observed in our empirical evaluation that Algorithm 1 typically converges within less than 20 iterations (details in § 4).

# 4 MUTUAL REINFORCEMENT EFFECTS

Next we study the dynamic interactions between adversarial inputs and poisoned models. With both empirical and analytical evidence, we reveal that there exist intricate "mutual reinforcement" effects between the two attack vectors: (i) leverage effect – with fixed attack efficacy, at slight cost of one metric (i.e., fidelity or specificity), one can disproportionally improve the other metric; (ii) amplification effect – with one metric fixed, at minimal cost of the other, one can greatly boost the attack efficacy.

## 4.1 Study Setting

**Datasets.** To factor out the influence of specific datasets, we primarily use 4 benchmark datasets:

- CIFAR10 [27] – It consists of $32 \times 32$ color images drawn from 10 classes (e.g., 'airplane');
- Mini-ImageNet – It is a subset of the ImageNet dataset [14], which consists of $224 \times 224$ (center-cropped) color images drawn from 20 classes (e.g., 'dog');
- ISIC [16] – It represents the skin cancer screening task from the ISIC 2018 challenge, in which given $600 \times 450$ skin lesion images are categorized into a 7-disease taxonomy (e.g., 'melanoma');
- GTSRB [46] – It consists of color images of size ranging from $29 \times 30$ to $144 \times 48$, each representing one of 43 traffic signs.

Note that among these datasets, ISIC and GTSRB in particular represent security-sensitive tasks (i.e., skin cancer screening [16] and traffic sign recognition [4]).

**DNNs.** We apply ResNet18 [23] to CIFAR10, GTSRB and ImageNet and ResNet101 to ISIC as the reference DNN models. Their top-1 accuracy on the testset of each dataset is summarized in Table 2. Using two distinct DNNs, we intend to factor out the influence of individual DNN characteristics (e.g., network capacity).

| | CIFAR10 | ImageNet | ISIC | GTSRB |
|---|---|---|---|---|
| Model | ResNet18 | ResNet18 | ResNet101 | ResNet18 |
| Accuracy | 95.23% | 94.56% | 88.18% | 99.12% |

**Table 2. Accuracy of benign DNNs on reference datasets.**

**Attacks.** Besides the IMC attack in § 3.2, we also implement two variants of IMC (with the same hyper-parameter setting) for comparison: (i) input perturbation only, in which IMC is instantiated as the adversarial attack (i.e., PGD [34]), and (ii) model perturbation only, in which IMC is instantiated as the poisoning attack. The implementation details are deferred to Appendix B.

**Measures.** We quantify the attack objectives as follows.

*Efficacy* – We measure the attack efficacy by the misclassification confidence, $f_t(x_*; \theta_*)$, which is the probability that the adversarial input $x_*$ belongs to the target class $t$ as predicted by the poisoned model $\theta_*$. We consider the attack successful if the misclassification confidence exceeds a threshold $\kappa$.

*Fidelity* – We measure the fidelity loss by the $L_p$-norm of the input perturbation $\ell_{\mathrm{f}}(x_*) \triangleq \|x_* - x_\circ\|_p$. Following previous work on adversarial attacks [7, 20, 34], we use $p = \infty$ by default in the following evaluation.

*Specificity* – Further, we measure the specificity loss using the difference of the benign and poisoned models on classifying a reference set $\mathcal{R}$. Let $\mathbb{I}_z$ be the indicator function that returns 1 if $z$ is true and 0 otherwise. The specificity loss can be defined as:

$$\ell_{\mathrm{s}}(\theta_*) \triangleq \sum_{x \in \mathcal{R}} \frac{\mathbb{I}_{f(x;\theta_\circ) \neq f(x;\theta_*)}}{|\mathcal{R}|} \tag{9}$$

With fixed attack efficacy $\kappa$, let $(x_*, \theta_*)$ be the adversarial input and poisoned model generated by IMC, and $\bar{x}_*$ and $\bar{\theta}_*$ be the adversarial input and poisoned model given by the adversarial and poisoning attacks respectively. Because the adversarial and poisoning attacks are special variants of IMC, we have $x_* = \bar{x}_*$ if $\theta_* = \theta_\circ$ and $\theta_* = \bar{\theta}_*$ if $x_* = x_\circ$. Thus, in the following, we normalize the fidelity and specificity losses as $\ell_{\mathrm{f}}(x_*)/\ell_{\mathrm{f}}(\bar{x}_*)$ and $\ell_{\mathrm{s}}(\theta_*)/\ell_{\mathrm{s}}(\bar{\theta}_*)$ respectively, both of which are bound to $[0, 1]$. For reference, the concrete specificity losses $\ell_{\mathrm{s}}(\bar{\theta}_*)$ (average accuracy drop) caused by the poisoning attack on each dataset are summarized in Table 3.

| $\kappa$ | CIFAR10 | ImageNet | ISIC | GTSRB |
|---|---|---|---|---|
| 0.75 | 0.22% | 1.94% | 1.62% | 0.14% |
| 0.9 | 0.25% | 1.96% | 1.63% | 0.20% |

**Table 3. Specificity losses (average accuracy drop) caused by poisoning attacks on reference datasets.**

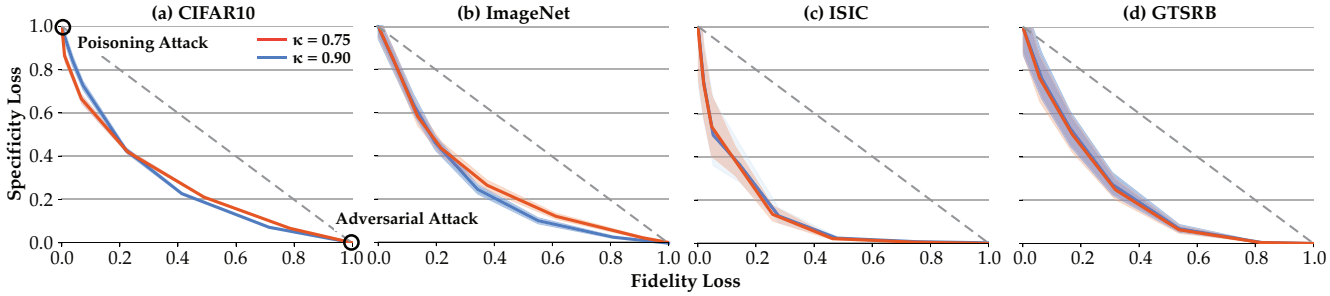


**Figure 4: Disproportionate trade-off between attack fidelity and specificity.**

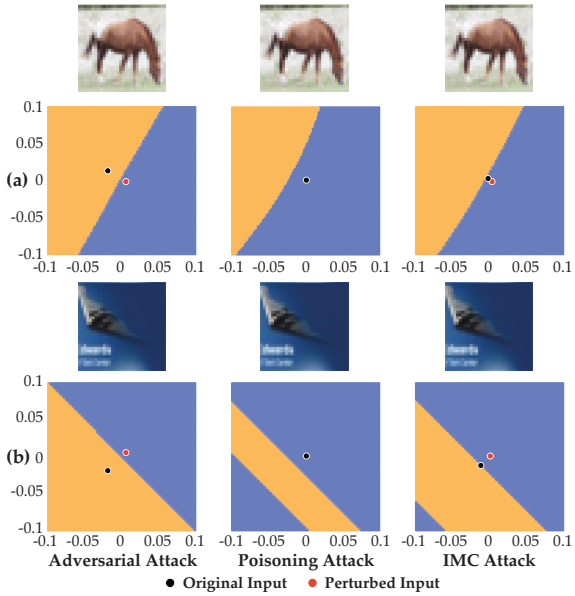


**Figure 5: Sample inputs and surrounding classification boundaries under various attacks on CIFAR10. The output of the DNN's penultimate layer is considered as the feature space, in which two random, orthogonal directions are selected as the x- and y-axis, while each colored region represents one distinct class. (a) 'horse' misclassified as 'deer'; (b) 'airplane' misclassified as 'bird'.**

## 4.2 Effect I: Leverage Effect

In the first set of experiments, we show that for fixed attack efficacy, with disproportionally small cost of fidelity, it is feasible to significantly improve the attack specificity, and vice versa.

*4.2.1* **Disproportionate Trade-off.** For each dataset, we apply the adversarial, poisoning, and IMC attacks against 1,000 inputs randomly sampled from the testset (as the target set $\mathcal{T}$), and use the rest as the reference set $\mathcal{R}$ to measure the specificity loss. For each input of $\mathcal{T}$, we randomly select its target class and fix the required attack efficacy (i.e., misclassification confidence $\kappa$). By varying IMC's hyper-parameters $\lambda$ and $\nu$, we control the importance of fidelity and specificity. We then measure the fidelity and specificity losses for all the successful cases. Figure 4 illustrates how IMC balances fidelity and specificity. Across all the datasets and models, we have the following observations.

First, with fixed attack efficacy (i.e., $\kappa = 0.75$), by sacrificing disproportionally small fidelity (i.e., input perturbation magnitude), IMC significantly improves the attack specificity (i.e., accuracy drop on non-target inputs), compared with required by the corresponding poisoning attack. For instance, in the case of ISIC (Figure 4 (c)), as the fidelity loss increases from 0 to 0.05, the specificity loss is reduced by more than 0.48.

Second, this effect is symmetric: a slight increase of specificity loss also leads to significant fidelity improvement, compared with required by the corresponding adversarial attack. For instance, in the case of CIFAR10 (Figure 4 (a)), as the specificity loss increases from 0 to 0.05, the specificity loss drops by 0.4.

Third, the fidelity-specifcity trade-off is not sensitive to the attack efficacy setting. Observe that across all the cases, the trade-offs show similar patterns for $\kappa = 0.75$ and 0.9 and differ only slightly in variance. Thus, we will fix $\kappa = 0.75$ in the following study.

Figure 5 showcases sample inputs and their surrounding classification boundaries in the feature space. Observe that IMC attains the same efficacy but with much less fidelity or specificity loss compared with the adversarial or poisoning attack.

**Leverage Effect**

There exists an intricate fidelity-specificity trade-off. At disproportionally small cost of fidelity, it is possible to significantly improve specificity, and vice versa.

*4.2.2* **Empirical Implications.** The leverage effect has profound implications. We show that it entails a large design spectrum for the adversary to optimize the attack evasiveness with respect to various detection methods (detectors). Note that here we do not consider the adversary's adaptiveness to specific detectors but rather focus on exposing the design spectrum enabled from a detection perspective. In § 5, we show that IMC also allows to enhance the attacks by adapting to specific detectors. To assess IMC's evasiveness, we consider three complementary detectors.

*Input Anomaly* – From the input anomaly perspective, we apply manifold transformation [35] as the detector. At a high level, it employs a reformer network to project given inputs to the manifold spanned by benign inputs and a detector network to differentiate benign and adversarial inputs. Besides, we apply randomized smoothing [11] as another detector, which transforms a given DNN into a "smoothed" model and considers a given input $x_*$ as adversarial if the probability difference of $x_*$'s largest and second largest classes exceeds a threshold.

*Model Anomaly* – From the model anomaly perspective, we apply curvature profiling [38] as the detector. Recall that the poisoning attack twists the classification boundary surrounding the target

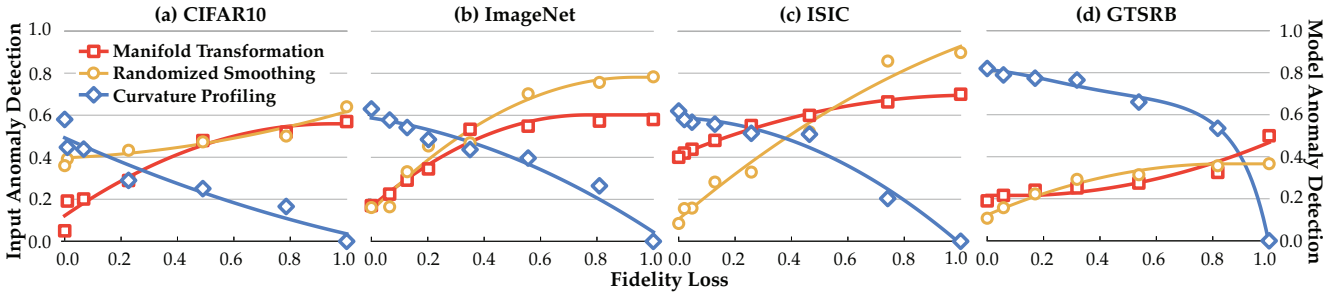


**Figure 6: Detection rates of input anomaly (by manifold projection [35]) and model anomaly (by curvature profile [38]).**

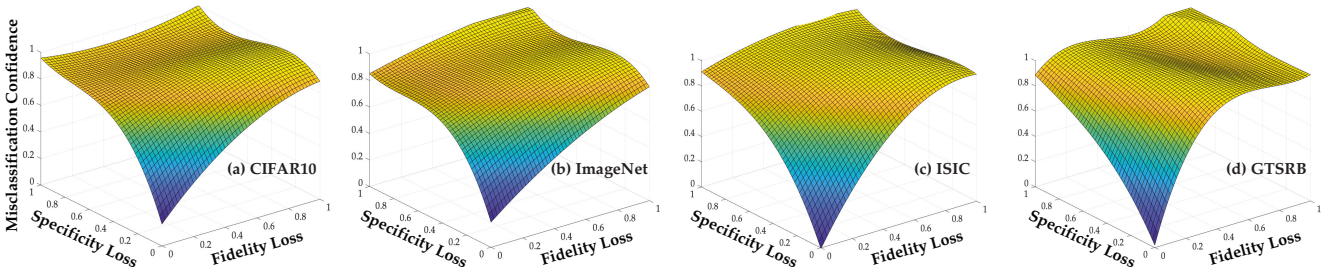


**Figure 7: Average misclassification confidence ($\kappa$) as a function of fidelity and specificity losses.**

input $x_*$; thus, the loss function tends to change abruptly in $x_*$'s vicinity. To quantify this property, we compute the eigenvalues of the Hessian $H_x(x_*) = \nabla_x^2 \ell(x_*)$. Intuitively, larger (absolute) eigenvalues indicate larger curvatures of the loss function. We define the average (absolute) value of the top-$k$ eigenvalues of $H_x(x_*)$ as $x_*$'s curvature profile: $\frac{1}{k}\sum_{i=1}^{k} |\lambda_i(H_x(x_*))|$, where $\lambda_i(M)$ is the $i$-th eigenvalue of matrix $M$ (details in Appendix B). We compare the curvature profiles of given inputs and benign ones, and use the Kolmogorov–Smirnov statistics to differentiate the two sets.

We apply the above detectors to the adversarial inputs and poisoned models generated by IMC under varying fidelity-specificity trade-off ($\kappa$ fixed as 0.75). Figure 6 measures the detection rates for different datasets. We have the following observations.

The detection rate of input anomaly grows monotonically with the fidelity loss (i.e., input perturbation magnitude); on the contrary, the detection rate of model anomaly drops quickly with the fidelity loss (i.e., disproportionate specificity improvement due to the leverage effect). For instance, in the case of ImageNet (Figure 6 (b)), as the fidelity loss varies from 0 to 0.35, the detection rate of input anomaly increases from 0.17 to 0.53 by manifold transformation and from 0.16 to 0.47 by randomized smoothing, while the detection rate of corresponding model anomaly drops from 0.63 to 0.44.

Moreover, across all the cases, IMC is able to balance fidelity and specificity, leading to high evasiveness with respect to multiple detectors simultaneously. For instance, in the case of CIFAR10 (Figure 6 (a)), with the fidelity loss set as 0.23, the detection rates of manifold transformation, randomized smoothing, and curvature profiling are reduced to 0.29, 0.43, and 0.29 respectively.

## 4.3 Effect II: Amplification Effect

Next we show that the two attack vectors are able to amplify each other and attain attack efficacy unreachable by each vector alone.

*4.3.1* **Mutual Amplification.** We measure the attack efficacy (average misclassification confidence) attainable by the adversarial, poisoning, and IMC attacks under varying fidelity and specificity losses. The results are shown in Figure 7 We have two observations.

First, IMC realizes higher attack efficacy than simply combining the adversarial and poisoning attacks. For instance, in the case of ISIC (Figure 7 (c)), with fidelity loss fixed as 0.2, the adversarial attack achieves $\kappa$ about 0.25; with specificity loss fixed as 0.2, the poisoning attack attains $\kappa$ around 0.4; while IMC reaches $\kappa$ above 0.8 under this setting. This is explained by that IMC employs a stronger threat model to jointly optimize the perturbations introduced at both training and inference.

Second, IMC is able to attain attack efficacy unreachable by using each attack vector alone. Across all the cases, IMC achieves $\kappa = 1$ under proper fidelity and specificity settings, while the adversarial (or poisoning) attack alone (even with fidelity or specificity loss fixed as 1) is only able to reach $\kappa$ less than 0.9.

**Amplification Effect**

Adversarial inputs and poisoned models amplify each other and give rise to attack efficacy unreachable by using each vector alone.

*4.3.2* **Empirical Implications.** This amplification effect entails profound implications for the adversary to design more effective attacks. Here we explore to use adversarial training [34, 43], one state-of-the-art defense against adversarial attacks [2], to cleanse poisoned models. Starting with the poisoned model, the re-training iteratively updates it with adversarial inputs that deceive its current configuration (i.e., adversarial "re-training").

We perform adversarial re-training on each poisoned model $\theta_*$ generated by IMC under varying fidelity-specificity trade-off (implementation details in Appendix B). We evaluate the re-trained model

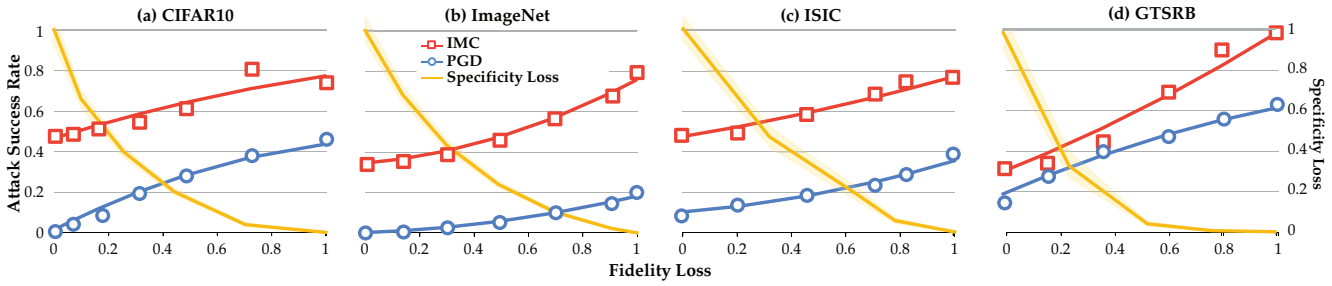


**Figure 8: Accuracy and robustness (with respect to PGD and IMC) of adversarially re-trained models.**

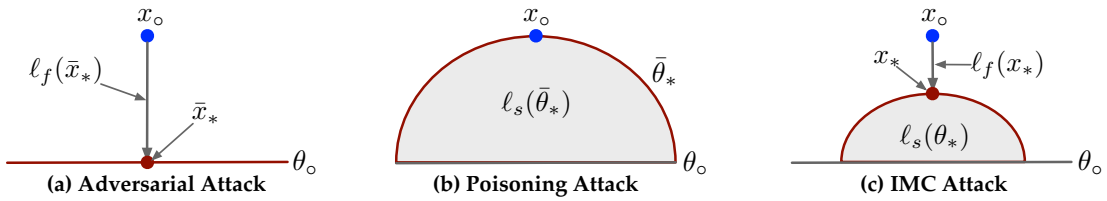


**Figure 9: Comparison of the adversarial, poisoning, and IMC attacks under fixed attack efficacy.**

| Dataset | Maximum Perturbation | |
|---|---|---|
| | PGD | IMC |
| CIFAR10 | $3 \times 10^{-2}$ | $2 \times 10^{-3}$ |
| ImageNet | $4 \times 10^{-3}$ | $1 \times 10^{-3}$ |
| ISIC | $2 \times 10^{-1}$ | $1 \times 10^{-3}$ |
| GTSRB | $1 \times 10^{-1}$ | $2 \times 10^{-2}$ |

**Table 4. Maximum input perturbation magnitude for PGD and IMC.**

$\tilde{\theta}_*$ in terms of (i) the attack success rate of PGD (i.e., $\tilde{\theta}_*$'s robustness against regular adversarial attacks), (ii) the attack success rate of $\theta_*$'s corresponding adversarial input $x_*$ (i.e., $\tilde{\theta}_*$'s robustness against IMC), and (iii) $\tilde{\theta}_*$'s overall accuracy over benign inputs in the testset. Note that in order to work against the re-trained models, PGD is enabled with significantly higher perturbation magnitude than IMC. Table 4 summarizes PGD and IMC's maximum allowed perturbation magnitude (i.e., fidelity loss) for each dataset.

Observe that adversarial re-training greatly improves the robustness against PGD, which is consistent with prior work [34, 43]. Yet, due to the amplification effect, IMC retains its high attack effectiveness against the re-trained model. For instance, in the case of ISIC (Figure 8 (c)), even with the maximum perturbation, PGD attains less than 40% success rate; in comparison, with two orders of magnitude lower perturbation, IMC succeeds with close to 80% chance. This also implies that adversarial re-training is in general ineffective against IMC. Also observe that by slightly increasing the input perturbation magnitude, IMC sharply improves the specificity of the poisoned model (e.g., average accuracy over benign inputs), which is attributed to the leverage effect. Note that while here IMC is not adapted to adversarial re-training, it is possible to further optimize the poisoned model by taking account of this defense during training, similar to [54].

## 4.4 Analytical Justification

We now provide analytical justification for the empirical observations regarding the mutual reinforcement effects.

*4.4.1* **Loss Measures.** Without loss of generality, we consider a binary classification setting (i.e., $\mathcal{Y} = \{0, 1\}$), with $(1 - t)$ and $t$ being the benign input $x_\circ$'s ground-truth class and the adversary's target class respectively. Let $f_t(x; \theta)$ be the model $\theta$'s predicted probability that $x$ belongs to $t$. Under this setting, we quantify the set of attack objectives as follows.

*Efficacy* – The attack succeeds only if the adversarial input $x_*$ and poisoned model $\theta_*$ force $f_t(x_*; \theta_*)$ to exceed 0.5 (i.e., the input crosses the classification boundary). We thus use $\kappa \triangleq f_t(x_\circ; \theta_\circ) - 0.5$ to measure the current gap between $\theta_\circ$'s prediction regarding $x_\circ$ and the adversary's target class $t$.

*Fidelity* – We quantify the fidelity loss using the $L_p$-norm of the input perturbation: $\ell_f(x_*) = \|x_* - x_\circ\|_p$. For two adversarial inputs $x_*, x'_*$, we say $x_* < x'_*$ if $\ell_f(x_*) < \ell_f(x'_*)$. For simplicity, we use $p = 2$, while the analysis generalizes to other norms as well.

As shown in Figure 9 (a), in a successful adversarial attack (with the adversarial input $\bar{x}_*$), if the perturbation magnitude is small enough, we can approximate the fidelity loss as $x_\circ$'s distance to the classification boundary [37]: $\ell_f(\bar{x}_*) \approx \kappa / \|\nabla_x \ell(x_\circ; \theta_\circ)\|_2$, where a linear approximation is applied to the loss function. In the following, we denote $h \triangleq \ell_f(\bar{x}_*)$.

*Specificity* – Recall that the poisoned model $\theta_*$ modifies $x_\circ$'s surrounding classification boundary, as shown in Figure 9 (b). While it is difficult to exactly describe the classification boundaries encoded by DNNs [17], we approximate the local boundary surrounding an input with the surface of a $d$-dimensional sphere, where $d$ is the input dimensionality. This approximation is justified as follows.

First, it uses a quadratic form, which is more expressive than a linear approximation [37]. Second, it reflects the impact of model complexity on the boundary: the maximum possible curvature of the boundary is often determined by the model's inherent complexity [17]. For instance, the curvature of a linear model is 0, while a one hidden-layer neural network with an infinite number of neurons is able to model arbitrary boundaries [12]. We relate the model's complexity to the maximum possible curvature, which corresponds to the minimum possible radius of the sphere.

The boundaries before and after the attacks are thus described by two hyper-spherical caps. As the boundary before the attack is

fixed, without loss of generality, we assume it to be flat for simplicity. Now according to Eqn (9), the specificity loss is measured by the number of inputs whose classifications are changed due to $\theta$. Following the assumptions, such inputs reside in a $d$-dimensional hyper-spherical cap, as shown in Figure 9 (b). Due to its minuscule scale, the probability density $p_{\text{data}}$ in this cap is roughly constant. Minimizing the specificity loss is thus equivalent to minimizing the cap volume [41], which amounts to maximizing the curvature of the sphere (or minimizing its radius). Let $r$ be the minimum radius induced by the model. We quantify the specificity loss as:

$$\ell_{\text{s}}(\theta) = p_{\text{data}} \frac{\pi^{\frac{d-1}{2}} r^d}{\Gamma\left(\frac{d+1}{2}\right)} \int_0^{\arccos\left(1-\frac{h}{r}\right)} \sin^d(t)\,\mathrm{d}t \tag{10}$$

where $\Gamma(z) \triangleq \int_0^\infty t^{z-1}e^{-t}\,\mathrm{d}t$ is the Gamma function.

*4.4.2* **Mutual Reinforcement Effects.** Let $\bar{x}_*, \bar{\theta}_*$ be the adversarial input and poisoned model given by the adversarial and poisoning attacks respectively, and $(x_*, \theta_*)$ be the adversarial input and poisoned model generated by IMC. Note that for fixed attack efficacy, $x_* = \bar{x}_*$ if $\theta_* = \theta_\circ$ and $\theta_* = \bar{\theta}_*$ if $x_* = x_\circ$.

*Leverage Effect* – We now quantify the leverage effect in the case of trading fidelity for specificity, while the alternative case can be derived similarly. Specifically, this effect is measured by the ratio of specificity "saving" versus fidelity "cost", which we term as the *leverage effect coefficient*:

$$\phi(x_*, \theta_*) \triangleq \frac{1 - \ell_{\text{s}}(\theta_*)/\ell_{\text{s}}(\bar{\theta}_*)}{\ell_{\text{f}}(x_*)/\ell_{\text{f}}(\bar{x}_*)} \tag{11}$$

Intuitively, the numerator is the specificity "saving", while the denominator is the fidelity "cost". We say that the trade-off is significantly disproportionate, if $\phi(x_*, \theta_*) \gg 1$, i.e., the saving dwarfs the cost. It is trivial to verify that if $\phi(x_*, \theta_*) \gg 1$ then the effect of trading specificity for fidelity is also significant $\phi(\theta_*, x_*) \gg 1$.[3]

Consider the IMC attack as shown in Figure 9 (c). The adversarial input $x_*$ moves towards the classification boundary and reduces the loss by $\kappa'(\kappa' < \kappa)$. The perturbation magnitude is thus at least $\kappa'/\|\nabla_x \ell(x_\circ; \theta_\circ)\|_2$. The relative fidelity loss is given by:

$$\ell_{\text{f}}(x_*)/\ell_{\text{f}}(\bar{x}_*) = \kappa'/\kappa \tag{12}$$

Below we use $z = \kappa'/\kappa$ for a short notation.

Meanwhile, it is straightforward to derive that the height of the hyper-spherical cap is $(1 - z)h$. The relative specificity loss is thus:

$$\ell_{\text{s}}(\theta_*)/\ell_{\text{s}}(\bar{\theta}_*) = \frac{\int_0^{\arccos\left(1-\frac{h}{r}+z\frac{h}{r}\right)} \sin^d(t)\,\mathrm{d}t}{\int_0^{\arccos\left(1-\frac{h}{r}\right)} \sin^d(t)\,\mathrm{d}t} \tag{13}$$

Instantiating Eqn (11) with Eqn (12) and Eqn (13), the leverage effect of trading fidelity for specificity is defined as:

$$\phi(x_*, \theta_*) = \frac{\int_{\arccos\left(1-\frac{h}{r}+z\frac{h}{r}\right)}^{\arccos\left(1-\frac{h}{r}\right)} \sin^d(t)\,\mathrm{d}t}{z\int_0^{\arccos\left(1-\frac{h}{r}\right)} \sin^d(t)\,\mathrm{d}t} \tag{14}$$

[3] If $(1 - x)/y \gg 1$ then $(1 - y)/x \gg 1$ for $0 < x, y < 1$.

The following proposition justifies the effect of trading fidelity for specificity (proof in Appendix A). A similar argument can be derived for trading specificity for fidelity.

**Proposition 2.** The leverage effect defined in Eqn (14) is strictly greater than 1 for any $0 < z < 1$.

Intuitively, to achieve fixed attack efficacy ($\kappa$), with a slight increase of fidelity loss $\ell_{\text{f}}(x_*)$, the specificity loss $\ell_{\text{s}}(\theta_*)$ is reduced super-linearly.

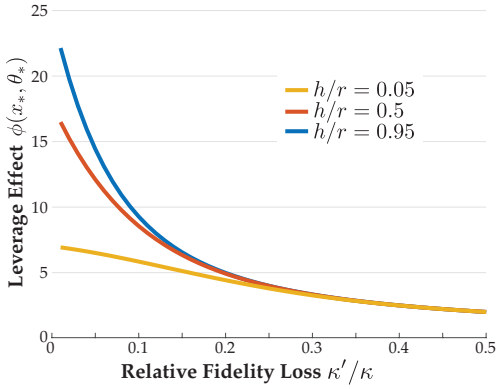


**Figure 10: Leverage effect with respect to the relative fidelity loss $z$ and the minimum radius $r$ (with $d = 50$).**

Figure 10 evaluates this effect as a function of relative fidelity loss under varying setting of $h/r$. Observe that the effect is larger than 1 by a large margin, especially for small fidelity loss $\kappa'/\kappa$, which is consistent with our empirical observation: with little fidelity cost, it is possible to significantly reduce the specificity loss.

*Amplification Effect* – From Proposition 2, we can also derive the explanation for the amplification effect.

Consider an adversarial input $x_*$ that currently achieves attack efficacy $\kappa'$ with relative fidelity loss $\kappa'/\kappa$. Applying the poisoned model $\theta_*$ with relative specificity loss $(1-\kappa'/\kappa)/\phi(x_*, \theta_*)$, the adversary is able to attain attack efficacy $\kappa$. In other words, the poisoned model $\theta_*$ "amplifies" the attack efficacy of the adversarial input $x_*$ by $\kappa/\kappa'$ times, with cost much lower than required by using the adversarial attack alone to reach the same attack efficacy (i.e., $1 - \kappa'/\kappa$), given that $\phi(x_*, \theta_*) \gg 1$ in Proposition 2.

## 5 IMC-OPTIMIZED ATTACKS

In this section, we demonstrate that IMC, as a general attack framework, can be exploited to enhance existing attacks with respect to multiple metrics. We further discuss potential countermeasures against such optimized attacks and their technical challenges.

### 5.1 Attack Optimization

*5.1.1* **Basic Attack.** We consider TrojanNN [32], a representative backdoor attack, as the reference attack model. At a high level, TrojanNN defines a specific pattern (e.g., watermark) as the trigger and enforces the poisoned model to misclassify all the inputs embedded with this trigger. As it optimizes both the trigger and poisoned model, TrojanNN enhances other backdoor attacks (e.g., BadNet [21]) that employ fixed trigger patterns.

Specifically, the attack consists of three steps. (i) First, the trigger pattern is partially defined in an ad hoc manner; that is, the watermark shape (e.g., square) and embedding position are pre-specified. (ii) Then, the concrete pixel values of the trigger are optimized to activate neurons rarely activated by benign inputs, in order to minimize the impact on benign inputs. (iii) Finally, the model is re-trained to enhance the effectiveness of the trigger pattern.

Note that within TrojanNN, the operations of trigger optimization and model re-training are executed independently. It is thus possible that after re-training, the neurons activated by the trigger pattern may deviate from the originally selected neurons, resulting in suboptimal trigger patterns and/or poisoned models. Also note that TrojanNN works without access to the training data; yet, to make fair comparison, in the following evaluation, TrojanNN also uses the original training data to construct the backdoors.

*5.1.2* **Enhanced Attacks.** We optimize TrojanNN within the IMC framework. Compared with optimizing the trigger only [29], IMC improves TrojanNN in terms of both attack effectiveness and evasiveness. Specifically, let $r$ denote the trigger. We initialize $r$ with the trigger pre-defined by TrojanNN and optimize it using the co-optimization procedure. To this end, we introduce a mask $m$ for the given benign input $x_\circ$. For $x_\circ$'s $i$-th dimension (pixel), we define $m[i] = 1 - p$ ($p$ is the transparency setting) if $i$ is covered by the watermark and $m[i] = 0$ otherwise. Thus the perturbation operation is defined as $x_* = \psi(x_\circ, r; m) = x_\circ \odot (1 - m) + r \odot m$, where $\odot$ denotes element-wise multiplication. We reformulate the backdoor attack in Eqn (5) as follows:

$$\min_{\theta, r, m} \ \mathbb{E}_{x_\circ \in \mathcal{T}} \left[ \ell(\psi(x_\circ, r; m), t; \theta) \right] + \lambda \ell_{\mathrm{f}}(m) + \nu \ell_{\mathrm{s}}(\theta) \qquad (15)$$

where we define the fidelity loss in terms of $m$. Typically, $\ell_{\mathrm{f}}(m)$ is defined as $m$'s $L_1$ norm and $\ell_{\mathrm{s}}(\theta)$ is the accuracy drop on benign cases similar to Eqn (9).

Algorithm 2 sketches the optimization procedure of Eqn (15). It alternates between optimizing the trigger and mask (line 4) and optimizing the poisoned model (line 5). Specifically, during the trigger perturbation step, we apply the Adam optimizer [26]. Further, instead of directly optimizing $r$ which is bounded by $[0, 1]$, we apply change-of-variable and optimize over a new variable $w_r \in (-\infty, +\infty)$, such that $r = (\tanh(w_r) + 1)/2$ (the same trick is also applied on $m$). Note that Algorithm 2 represents a general optimization framework, which is adaptable to various settings. For instance, one may specify all the non-zero elements of $m$ to share the same transparency or optimize the transparency of each element independently (details in § 5.2 and § 5.3). In the following, we term the enhanced TrojanNN as TrojanNN*.

## 5.2 Optimization against Human Vision

We first show that TrojanNN* is optimizable in terms of its evasiveness with respect to human vision. The evasiveness is quantified by the size and transparency (or opacity) of trigger patterns. Without loss of generality, we use square-shaped triggers. The trigger size is measured by the ratio of its width over the image width.

Figure 11 illustrates TrojanNN*'s attack efficacy (average misclassification confidence of trigger-embedded inputs) under varying evasiveness constraints. Observe that the efficacy increases sharply as a function of the trigger size or opacity. Interestingly, the trigger

**Algorithm 2:** TrojanNN* Attack

**Input:** initial trigger mask – $m_\circ$; benign model – $\theta_\circ$; target class – $t$; hyper-parameters – $\lambda$, $\nu$

**Output:** trigger mask – $m$; trigger pattern – $r$, poisoned model – $\theta_*$

// initialization

1 $\theta^{(0)}, k \leftarrow \theta_\circ, 0$;

2 $m^{(0)}, r^{(0)} \leftarrow m_0$, TrojanNN($m_0$);

// optimization

3 **while** *not converged yet* **do**

// trigger perturbation

4 $r^{(k+1)}, m^{(k+1)} \leftarrow \arg\min_{r,m} \mathbb{E}_{x_\circ \in \mathcal{T}} [\ell(\psi(x_\circ, r; m), t; \theta^{(k)})] + \lambda \ell_{\mathrm{f}}(m)$;

// model perturbation

5 $\theta^{(k+1)} \leftarrow \arg\min_\theta \mathbb{E}_{x_\circ \in \mathcal{T}} [\ell(\psi(x_\circ, r^{(k)}; m^{(k)}), t; \theta)] + \nu \ell_{\mathrm{s}}(\theta)$;

6 $k \leftarrow k + 1$;

7 **return** $(m^{(k)}, r^{(k)}, \theta^{(k)})$;

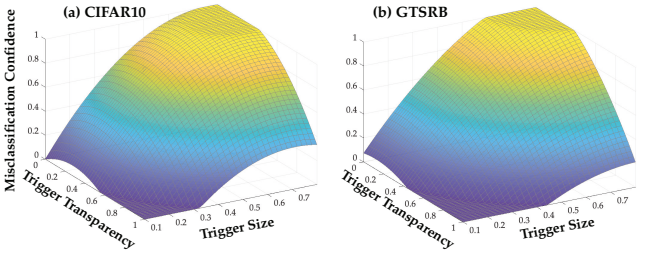


**Figure 11: Attack efficacy of TrojanNN* as a function of trigger size and transparency.**

size and opacity also demonstrate strong mutual reinforcement effects: (i) leverage - for fixed attack efficacy, by a slight increase in opacity (or size), it significantly reduces the size (or opacity); (ii) amplification - for fixed opacity (or size), by slightly increasing size (or opacity), it greatly boosts the attack efficacy.

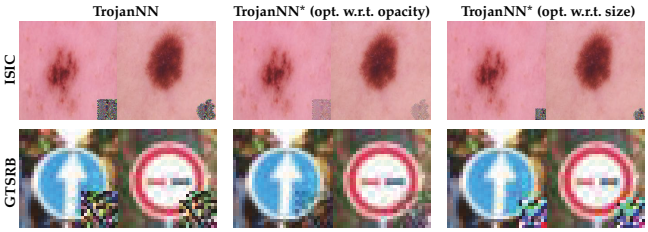


**Figure 12: Sample triggers generated by TrojanNN (a), TrojanNN* optimizing opacity (b) and optimizing size (c).**

To further validate the leverage effect, we compare the triggers generated by TrojanNN and TrojanNN*. Figure 12 shows sample triggers given by TrojanNN and TrojanNN* under fixed attack efficacy (with $\kappa = 0.95$). It is observed that compared with TrojanNN, TrojanNN* significantly increases the trigger transparency (under fixed size) or minimizes the trigger size (under fixed opacity).

To further validate the amplification effect, we measure the attack success rate (ASR) of TrojanNN and TrojanNN* under varying evasiveness constraints (with $\kappa = 0.95$), with results shown in Figure 13 and 14. It is noticed that across all the datasets, TrojanNN* outperforms TrojanNN by a large margin under given trigger size and opacity. For instance, in the case of GTSRB (Figure 13 (d)), with

trigger size and transparency fixed as 0.4 and 0.7, TrojanNN* outperforms TrojanNN by 0.39 in terms of ASR; in the case of CIFAR10 (Figure 14 (a)), with trigger size and transparency fixed as 0.3 and 0.2, the ASRs of TrojanNN* and TrojanNN differ by 0.36.

We can thus conclude that leveraging the co-optimization framework, TrojanNN* is optimizable with respect to human detection without affecting its attack effectiveness.

## 5.3 Optimization against Detection Methods

In this set of experiments, we demonstrate that TrojanNN* is also optimizable in terms of its evasiveness with respect to multiple automated detection methods.

*5.3.1* **Backdoor Detection.** The existing backdoor detection methods can be roughly classified in two categories based on their application stages and detection targets. The first class is applied at the model inspection stage and aims to detect suspicious models and potential backdoors [9, 31, 51]; the other class is applied at inference time and aims to detect trigger-embedded inputs [8, 10, 15, 18]. In our evaluation, we use NeuralCleanse [51] and STRIP [18] as the representative methods of the two categories. In Appendix C, we also evaluate TrojanNN and TrojanNN* against ABS [31], another state-of-the-art backdoor detector.

*NeuralCleanse* – For a given DNN, NeuralCleanse searches for potential triggers in every class. Intuitively, if a class is embedded with a backdoor, the minimum perturbation (measured by its $L_1$-norm) necessary to change all the inputs in this class to the target class is abnormally smaller than other classes. Empirically, after running the trigger search algorithm over 1,600 randomly sampled inputs for 10 epochs, a class with its minimum perturbation normalized by median absolute deviation exceeding 2.0 is considered to contain a potential backdoor with 95% confidence.

*STRIP* – For a given input, STRIP mixes it up with a benign input using equal weights, feeds the mixture to the target model, and computes the entropy of the prediction vector (i.e., self-entropy). Intuitively, if the input is embedded with a trigger, the mixture is still dominated by the trigger and tends to be misclassified to the target class, resulting in relatively low self-entropy; otherwise, the self-entropy tends to be higher. To reduce variance, for a given input, we average its self-entropy with respect to 8 randomly sampled benign inputs. We set the positive threshold as 0.05 and measure STRIP's effectiveness using F-1 score.

*5.3.2* **Attack Optimization.** We optimize TrojanNN* in terms of its evasiveness with respect to both NeuralCleanse and STRIP. Both detectors aim to detect anomaly under certain metrics, which we integrate into the loss terms in Algorithm 2.

Specifically, NeuralCleanse searches for potential trigger with minimum $L_1$-norm, which is related to the mask $m$. We thus instantiate the fidelity loss $\ell_{\mathrm{f}}(m)$ as $m$'s $L_1$-norm and optimize it during the trigger perturbation step. To normalize $\ell_{\mathrm{f}}(m)$ to an appropriate scale, we set the hyper-parameter $\lambda$ as the number of pixels covered by the trigger. Meanwhile, STRIP mixes adversarial and benign inputs and computes the self-entropy of the mixtures, which highly depends on the model's behaviors. We thus instantiate the specificity loss $\ell_{\mathrm{s}}(\theta)$ as $\mathbb{E}_{x,x'\in\mathcal{R}}\left[-H(f(\frac{x_*}{2}+\frac{x'}{2};\theta))\right]$, in which we randomly mix up an adversarial input $x_*$ (via perturbing a benign input $x$) and another benign input $x'$ and maximize the self-entropy of their mixture.

*5.3.3* **Detection Evasiveness.** We apply the above two detectors to detect TrojanNN and TrojanNN*, with results summarized in Figure 15. We have the following observations. First, the two detectors are fairly effective against TrojanNN. In comparison, TrojanNN* demonstrates much higher evasiveness. For instance, in the case of GTSRB (Figure 15 (b)), with trigger size fixed as 0.4, the anomaly measures of TrojanNN* and TrojanNN by NeuralCleanse differ by over 2, while the F-1 scores on TrojanNN* and TrojanNN by STRIP differ by more than 0.3. We thus conclude that TrojanNN* is optimizable in terms of evasiveness with respect to multiple detection methods simultaneously.

## 5.4 Potential Countermeasures

Now we discuss potential mitigation against IMC-optimized attacks and their technical challenges. It is shown above that using detectors against adversarial inputs or poisoned models independently is often insufficient to defend against IMC-optimized attacks, due to the mutual reinforcement effects. One possible solution is to build ensemble detectors that integrate individual ones and detect IMC-optimized attacks based on both input and model anomaly.

To assess the feasibility of this idea, we build an ensemble detector against TrojanNN* via integrating NeuralCleanse and STRIP. Specifically, we perform the following detection procedure: (i) applying NeuralCleanse to identify the potential trigger, (ii) for a given input, attaching the potential trigger to a benign input, (iii) mixing this benign input up with the given input under varying mixture weights, (iv) measuring the self-entropy of these mixtures, and (v) using the standard deviation of the self-entropy values to distinguish benign and trigger-embedded inputs.

Intuitively, if the given input is trigger-embedded, the mixture combines two trigger-embedded inputs and is thus dominated by one of the two triggers, regardless of the mixture weight, resulting in a low deviation of self-entropy. In comparison, if the given input is benign, the mixture is dominated by the trigger only if the weight is one-sided, resulting in a high deviation of self-entropy.

We compare the performance of the basic and ensemble STRIP against TrojanNN* (the detection against TrojanNN is deferred to Appendix C). As shown in Figure 16, the ensemble detector performs slightly better across all the cases, implying the effectiveness of the ensemble approach. However, the improvement is marginal (less than 0.2), especially in the case of small-sized triggers. This may be explained by the inherent challenges of defending against IMC-optimized attacks: due to the mutual reinforcement effects, TrojanNN* attains high attack efficacy with minimal input and model distortion; it thus requires to carefully account for such effects in order to design effective countermeasures.

# 6 RELATED WORK

With their increasing use in security-sensitive domains, DNNs are becoming the new targets of malicious manipulations [3]. Two primary attack vectors have been considered in the literature: adversarial inputs and poisoned models.

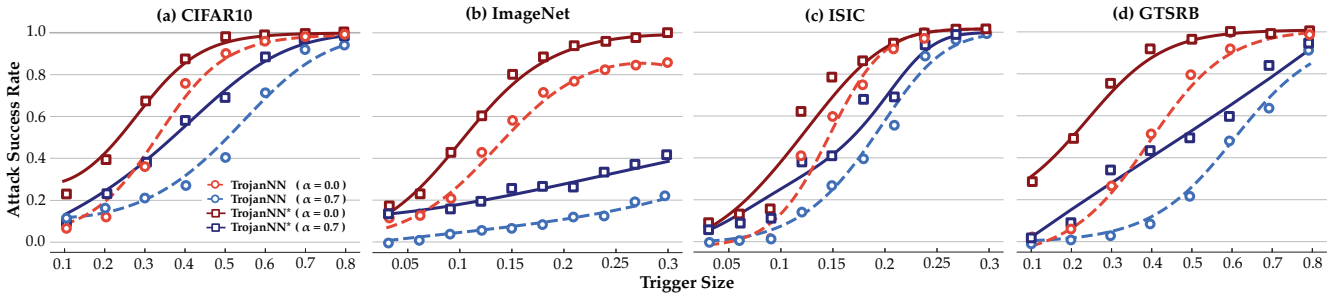


**Figure 13: ASR of TrojanNN and TrojanNN* as functions of trigger size.**

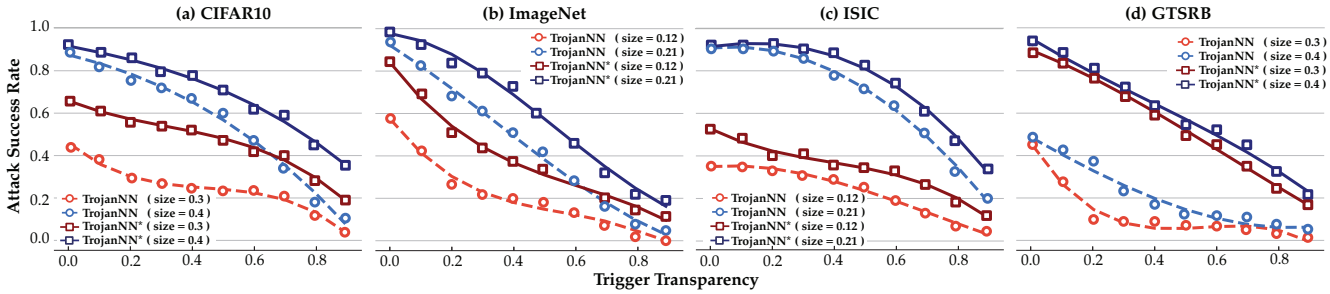


**Figure 14: ASR of TrojanNN and TrojanNN* as functions of trigger transparency.**

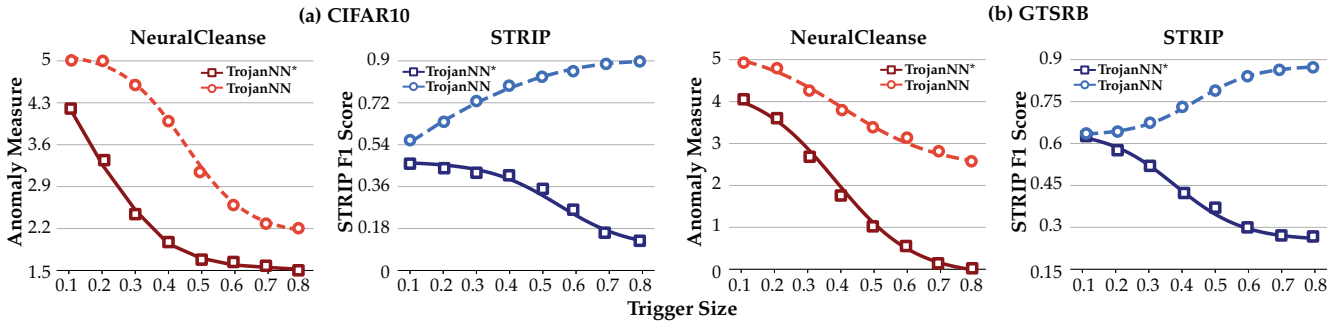


**Figure 15: Detection of TrojanNN and TrojanNN* by NeuralCleanse and STRIP on CIFAR10 and GTSRB.**

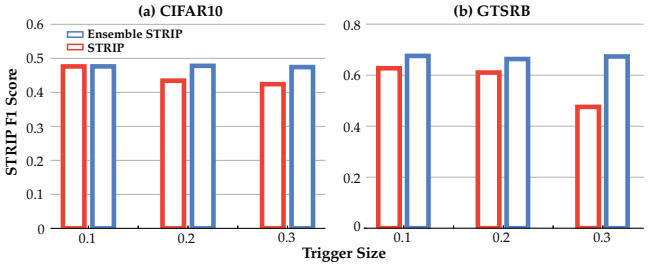


**Figure 16: Detection of basic and ensemble STRIP against TrojanNN* on CIFAR10 and GTSRB.**

**Adversarial Inputs.** The existing research on adversarial inputs is divided in two campaigns.

One line of work focuses on developing new attacks against DNNs [7, 20, 40, 48], with the aim of crafting adversarial samples to force DNNs to misbehave. The existing attacks can be categorized as untargeted (in which the adversary desires to simply force misclassification) and targeted (in which the adversary attempts to force the inputs to be misclassified into specific classes).

Another line of work attempts to improve DNN resilience against adversarial attacks by devising new training strategies (e.g., adversarial training) [22, 28, 39, 49] or detection mechanisms [19, 33, 35, 53]. However, the existing defenses are often penetrated or circumvented by even stronger attacks [2, 30], resulting in a constant arms race between the attackers and defenders.

**Poisoned Models.** The poisoned model-based attacks can be categorized according to their target inputs. In the poisoning attacks, the target inputs are defined as non-modified inputs, while the adversary's goal is to force such inputs to be misclassified by the poisoned DNNs [24, 25, 44, 47, 52]. In the backdoor attacks, specific trigger patterns (e.g., a particular watermark) are pre-defined, while the adversary's goal is to force any inputs embedded with such triggers to be misclassified by the poisoned models [21, 32]. Note that compared with the poisoning attacks, the backdoor attacks leverage both adversarial inputs and poisoned models.

The existing defense methods against poisoned models mostly focus on the backdoor attacks, which, according to their strategies, can be categorized as: (i) cleansing potential contaminated data at the training stage [50], (ii) identifying suspicious models during

model inspection [9, 31, 51], and (iii) detecting trigger-embedded inputs at inference time [8, 10, 15, 18].

Despite the intensive research on adversarial inputs and poisoned models in parallel, there is still a lack of understanding about their inherent connections. This work bridges this gap by studying the two attack vectors within a unified framework and providing a holistic view of the vulnerabilities of DNNs deployed in practice.

## 7 CONCLUSION

This work represents a solid step towards understanding adversarial inputs and poisoned models in a unified manner. We show both empirically and analytically that (i) there exist intriguing mutual reinforcement effects between the two attack vectors, (ii) the adversary is able to exploit such effects to optimize attacks with respect to multiple metrics, and (iii) it requires to carefully account for such effects in designing effective countermeasures against the optimized attacks. We believe our findings shed light on the holistic vulnerabilities of DNNs deployed in realistic settings.

This work also opens a few avenues for further investigation. First, besides the targeted, white-box attacks considered in this paper, it is interesting to study the connections between the two vectors under alternative settings (e.g., untargeted, black-box attacks). Second, enhancing other types of threats (e.g., latent backdoor attacks) within the input-model co-optimization framework is a direction worthy of exploration. Finally, devising a unified robustness metric accounting for both vectors may serve as a promising starting point for developing effective countermeasures.

## ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1910546, 1953813, and 1846151. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Shouling Ji was partly supported by NSFC under No. U1936215, 61772466, and U1836202, the National Key Research and Development Program of China under No. 2018YFB0804102, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, the Zhejiang Provincial Key R&D Program under No. 2019C01055, and the Ant Financial Research Funding. Xiapu Luo was partly supported by HK RGC Project (PolyU 152239/18E) and HKPolyU Research Grant (ZVQ8).

## REFERENCES

[1] Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. ADef: an Iterative Algorithm to Construct Adversarial Deformations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2018.

[3] Battista Biggio and Fabio Roli. Wild Patterns: Ten Years after The Rise of Adversarial Machine Learning. *Pattern Recognition*, 84:317–331, 2018.

[4] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars. *ArXiv e-prints*, 2016.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] BVLC. Model zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo, 2017.

[7] Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2017.

[8] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *ArXiv e-prints*, 2018.

[9] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2019.

[10] Edward Chou, Florian Tramer, Giancarlo Pellegrino, and Dan Boneh. SentiNet: Detecting Physical Attacks Against Deep Learning Systems. In *ArXiv e-prints*, 2018.

[11] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2019.

[12] G. Cybenko. Approximation by Superpositions of A Sigmoidal Function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[13] J.M. Danskin. *The Theory of Max-Min and Its Application to Weapons Allocation Problems*. Springer-Verlag, 1967.

[14] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-scale Hierarchical Image Database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[15] Bao Doan, Ehsan Abbasnejad, and Damith Ranasinghe. Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. In *ArXiv e-prints*, 2020.

[16] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks. *Nature*, 542(7639):115–118, 2017.

[17] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Stefano Soatto. Classification Regions of Deep Neural Networks. *ArXiv e-prints*, 2017.

[18] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith Ranasinghe, and Surya Nepal. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks. In *ArXiv e-prints*, 2019.

[19] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2018.

[20] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[21] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *ArXiv e-prints*, 2017.

[22] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering Adversarial Images Using Input Transformations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-Reuse Attacks on Deep Learning Systems. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2018.

[25] Yujie Ji, Xinyang Zhang, and Ting Wang. Backdoor Attacks against Learning Systems. In *Proceedings of IEEE Conference on Communications and Network Security (CNS)*, 2017.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[27] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. *Technical report, University of Toronto*, 2009.

[28] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[29] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor Embedding in Convolutional Neural Network Models via Invisible Perturbation. *ArXiv e-prints*, 2018.

[30] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.

[31] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2019.

[32] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning Attack on Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

[33] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*,

2019.
[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.
[35] Dongyu Meng and Hao Chen. MagNet: A Two-Pronged Defense Against Adversarial Examples. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2017.
[36] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal Adversarial Perturbations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[37] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of Universal Adversarial Perturbations. *ArXiv e-prints*, 2017.
[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via Curvature Regularization, and Vice Versa. *ArXiv e-prints*, 2018.
[39] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2016.
[40] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *Proceedings of IEEE European Symposium on Security and Privacy (Euro S&P)*, 2016.
[41] A.D. Polyanin and A.V. Manzhirov. *Handbook of Mathematics for Engineers and Scientists*. Taylor & Francis, 2006.
[42] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.
[43] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial Training for Free! In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
[44] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
[45] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, (7587):484–489, 2016.
[46] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural Metworks*, pages 323–32, 2012.
[47] Octavian Suciu, Radu Mărginean, Yiğitcan Kaya, Hal Daumé, III, and Tudor Dumitraş. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *Proceedings of USENIX Security Symposium (SEC)*, 2018.
[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.
[49] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.
[50] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral Signatures in Backdoor Attacks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
[51] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.
[52] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal Security Analysis of Neural Networks Using Symbolic Intervals. In *Proceedings of USENIX Security Symposium (SEC)*, 2018.
[53] W. Xu, D. Evans, and Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.
[54] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent Backdoor Attacks on Deep Neural Networks. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2019.
[55] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized Stochastic Gradient Descent. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*. 2010.

# APPENDIX

## A. Proofs

**A0. Preliminaries.** In the following proofs, we use the following definitions for notational simplicity:

$$\boxed{\alpha \triangleq h/r}$$

$$\boxed{y \triangleq 1 - z}$$

Further we have the following result.

$$\boxed{\int_0^{\arccos(x)} \sin^d(t)\mathrm{d}t = \int_x^1 \left(1-t^2\right)^{\frac{d-1}{2}} \mathrm{d}t} \tag{16}$$

**A1. Proof of Proposition 1.**

*Proof:* Recall that $\mathcal{F}_\epsilon(x_\circ)$ represents a non-empty compact set, $\ell(x;\cdot)$ is differentiable for $x \in \mathcal{F}_\epsilon(x_\circ)$, and $\nabla_\theta \ell(x,\theta)$ is continuous over the domains $\mathcal{F}_\epsilon(x_\circ) \times \mathbb{R}^n$.

Let $\mathcal{F}_\epsilon^*(x_\circ) = \{\arg\min_{x\in\mathcal{F}_\epsilon(x_\circ)} \ell(x;\theta)\}$ be the set of minimizers and $\ell(\theta) \triangleq \min_{x\in\mathcal{F}_\epsilon(x_\circ)} \ell(x;\theta)$. The Danskin's theorem [13] states that $\ell(\theta)$ is locally continuous and directionally differentiable. The derivative of $\ell(\theta)$ along the direction $d$ is given by

$$\mathrm{D}_d \ell(\theta) = \min_{x\in\mathcal{F}_\epsilon^*(x_\circ)} d^\top \nabla_\theta \ell(x,\theta)$$

We apply the Danskin's theorem to our case. Let $x^* \in \mathcal{F}_\epsilon(x_\circ)$ be a minimizer of $\min_x \ell(x;\theta)$. Consider the direction $d = -\nabla_\theta \ell(x^*;\theta)$. We then have:

$$\begin{aligned}\mathrm{D}_d \ell(\theta) &= \min_{x\in\mathcal{F}_\epsilon^*(x_\circ)} d^\top \nabla_\theta \ell(x,\theta) \\ &\le -\|\nabla_\theta \ell(x^*,\theta)\|_2^2 \le 0\end{aligned}$$

Thus, it follows that $\nabla_\theta \ell(x^*;\theta)$ is a proper descent direction of $\min_{x\in\mathcal{F}_\epsilon(x_\circ)} \ell(x;\theta)$. □

Note that in the proof above, we ignore the constraint of $\theta \in \mathcal{F}_\delta(\theta_\circ)$. Nevertheless, the conclusion is still valid. With this constraint, instead of considering the global optimum of $\theta$, we essentially consider its local optimum within $\mathcal{F}_\delta(\theta_\circ)$. Further, for DNNs that use constructs such as ReLU, the loss function is not necessarily continuously differentiable. However, since the set of discontinuities has measure zero, this is not an issue in practice.

**A2. Proof of Proposition 2.**

*Proof:* Proving $\phi(x_*,\theta_*) > 1$ is equivalent to showing the following inequality:

$$\frac{\int_0^{\arccos(1-y\alpha)} \sin^d(t)\mathrm{d}t}{y} < \int_0^{\arccos(1-\alpha)} \sin^d(t)\mathrm{d}t$$

We define $f(y) = \frac{1}{y}\int_0^{\arccos(1-y\alpha)} \sin^d(t)\mathrm{d}t$. This inequality is equivalent to $f(y) < f(1)$ for $y \in (0,1)$. It thus suffices to prove $f'(y) > 0$.

Considering Eqn (16), we have $f(y) = \frac{1}{y}\int_{1-y\alpha}^1 \left(1-t^2\right)^{\frac{d-1}{2}} \mathrm{d}t$ and $f'(y) = g(y)/y^2$ where

$$g(y) = y\alpha\left(1-(1-y\alpha)^2\right)^{\frac{d-1}{2}} - \int_{1-y\alpha}^1 \left(1-t^2\right)^{\frac{d-1}{2}} \mathrm{d}t$$

Denote $x = 1 - y\alpha$. We have

$$g(x) = (1+x)^{\frac{d-1}{2}}(1-x)^{\frac{d+1}{2}} - \int_x^1 \left(1-t^2\right)^{\frac{d-1}{2}} \mathrm{d}t$$

Note that $g(1) = 0$. With $d > 1$, we have

$$g'(x) = -(d-1)\,x\,(1+x)^{\frac{d-3}{2}}(1-x)^{\frac{d-1}{2}} < 0$$

Therefore, $g(x) > 0$ for $x \in (0,1)$, which in turn implies $f'(y) > 0$ for $y \in (0,1)$. □

## B. Implementation Details

Here we elaborate on the implementation of attacks and defenses in this paper.

**B1. Parameter Setting.** Table 5 summarizes the default parameter setting in our empirical evaluation (§ 4).

| Attack/Defense | Parameter | Setting |
|---|---|---|
| IMC | perturbation threshold | $\epsilon = 2 \times 10^{-3}$ |
| | learning rate | $\alpha = 1 \times 10^{-4}$ |
| | maximum iterations | $n_{\text{iter}} = 20$ |
| PGD | PGD | $\epsilon = 8/255$ |
| | learning rate | $\alpha = 3/255$ |
| | maximum iterations | $n_{\text{iter}} = 20$ |
| Manifold Transformation | network structure | [3,'average',3] |
| | random noise std | $v_{\text{noise}} = 0.1$ |
| Adversarial Re-Training | optimizer | SGD |
| | hop steps | $m = 4$ |
| | learning rate | $\alpha = 0.015$ |
| | learning rate decay | $\gamma = 0.9$ per 10 epochs |
| TrojanNN | neuron number | $n_{\text{neuron}} = 2$ |
| | threshold | 5 |
| | target value | 10 |
| STRIP | number of tests | $n_{\text{test}} = 8$ |

**Table 5. Default Parameter Setting**

**B2. Curvature Profile.** Exactly computing the eigenvalues of the Hessian matrix $H_x = \nabla_x^2 \ell(x)$ is prohibitively expensive for high-dimensional data. We use a finite difference approximation in our implementation. For any given vector $z$, the Hessian-vector product $H_x z$ can be approximated by:

$$H_x z = \lim_{\Delta \to 0} \frac{\nabla_x \ell(x + \Delta z) - \nabla_x \ell(x)}{\Delta} \tag{17}$$

By properly setting $\Delta$, this approximation allows us to measure the variation of gradient in $x$'s vicinity, rather than an infinitesimal point-wise curvature[38]. In practice we set $z$ as the gradient sign direction to capture the most variation:

$$z = \frac{\operatorname{sgn}(\nabla \ell(x))}{\|\operatorname{sgn}(\nabla \ell(x))\|} \tag{18}$$

and estimate the magnitude of curvature as

$$\|\nabla_x \ell(x + \Delta z) - \nabla_x \ell(x)\|^2 \tag{19}$$

We use Eqn (19) throughout the evaluation to compute the curvature profiles of given inputs.

## C. Additional Experiments

Here we provide experiment results additional to § 4 and § 5.

**C1. Detection of TrojanNN and TrojanNN* by ABS.** In addition to STRIP and NeuralCleanse, here we also evaluate TrojanNN and TrojanNN* against ABS [31], another state-of-the-art backdoor detection method. As the optimization of TrojanNN* requires white-box access to ABS, we re-implement ABS according to [31][4]. In the evaluation, we set the number of seed images as 5 per class and the maximum trojan size as 400.

Similar to NeuralCleanse, ABS attempts to detect potential backdoors embedded in given DNNs during model inspection. In a nutshell, its execution consists of two steps: (i) inspecting the given DNN to sift out abnormal neurons with large elevation difference (i.e., highly active only with respect to one particular class), and (ii) identifying potential trigger patterns by maximizing abnormal neuron activation while preserving normal neuron behaviors.

To optimize the evasiveness of TrojanNN* with respect to ABS, we integrate the cost function (Algorithm 2 in [31]) into the loss terms $\ell_f$ and $\ell_s$ in Algorithm 2 and optimize the trigger $r$ and model $\theta$ respectively to minimize this cost function.

The detection of TrojanNN and TrojanNN* by ABS on CIFAR10 is shown in Figure 17. Observe that ABS detects TrojanNN with close to 100% accuracy, which is consistent with the findings in [31]. In comparison, TrojanNN* is able to effectively evade ABS especially when the trigger size is sufficiently large. For instance, the detection rate (measured by maximum re-mask accuracy) drops to 40% as the trigger size increases to 0.4. This could be explained by that larger trigger size entails more operation space for TrojanNN* to optimize the trigger to evade ABS.

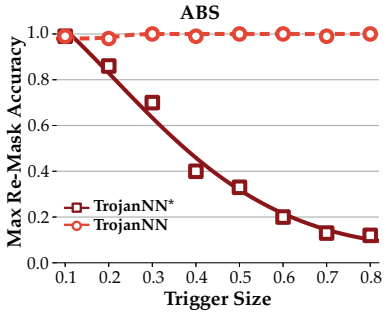


**Figure 17: Detection of TrojanNN and TrojanNN* by ABS on CIFAR10.**

**C2. Basic and Ensemble STRIP against TrojanNN.** Figure 18 below compares the performance of basic and ensemble STRIP in detecting TrojanNN. Interestingly, in contrary to the detection of TrojanNN* (Figure 16), here the basic STRIP outperforms the ensemble version. This may be explained as follows. As TrojanNN* is optimized to evade both STRIP and NeuralCleanse, to effectively detect it, ensemble STRIP needs to balance the metrics of both detectors; in contrast, TrojanNN is not optimized with respect to either detector. The compromise of ensemble STRIP results in its inferior performance compared with the basic detector.

[4]The re-implementation may have differences from the original ABS (https://github.com/naiyeleo/ABS).

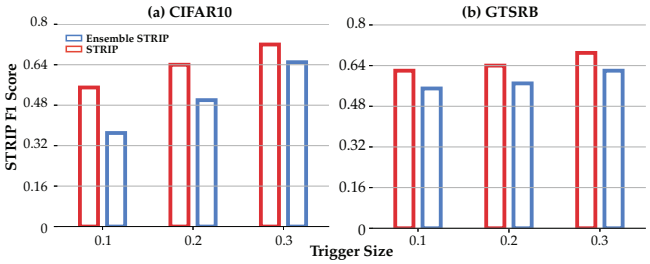


**Figure 18: Detection of basic and ensemble `STRIP` against `TrojanNN` on CIFAR10 and GTSRB.**