Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks

Ambra Demontis, Marco Melis, and Maura Pintor, *University of Cagliari, Italy;*
Matthew Jagielski, *Northeastern University;* Battista Biggio, *University of Cagliari, Italy,
and Pluribus One;* Alina Oprea and Cristina Nita-Rotaru, *Northeastern University;*
Fabio Roli, *University of Cagliari, Italy, and Pluribus One*

This paper is included in the Proceedings of the 28th USENIX Security Symposium.

August 14–16, 2019 • Santa Clara, CA, USA

978-1-939133-06-9

Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks

Ambra Demontis[†], Marco Melis[†], Maura Pintor[†], Matthew Jagielski[*], Battista Biggio[†,‡], Alina Oprea[*], Cristina Nita-Rotaru[*], and Fabio Roli[†,‡]

[†]Department of Electrical and Electronic Engineering, University of Cagliari, Italy
[‡]Pluribus One, Italy
[*]Northeastern University, Boston, MA, USA

Abstract

Transferability captures the ability of an attack against a machine-learning model to be effective against a different, potentially unknown, model. Empirical evidence for transferability has been shown in previous work, but the underlying reasons why an attack transfers or not are not yet well understood. In this paper, we present a comprehensive analysis aimed to investigate the transferability of both test-time evasion and training-time poisoning attacks. We provide a unifying optimization framework for evasion and poisoning attacks, and a formal definition of transferability of such attacks. We highlight two main factors contributing to attack transferability: the intrinsic *adversarial vulnerability* of the target model, and the *complexity* of the surrogate model used to optimize the attack. Based on these insights, we define three metrics that impact an attack's transferability. Interestingly, our results derived from theoretical analysis hold for both evasion and poisoning attacks, and are confirmed experimentally using a wide range of linear and non-linear classifiers and datasets.

1 Introduction

The wide adoption of machine learning (ML) and deep learning algorithms in many critical applications introduces strong incentives for motivated adversaries to manipulate the results and models generated by these algorithms. Attacks against machine learning systems can happen during multiple stages in the learning pipeline. For instance, in many settings training data is collected online and thus can not be fully trusted. In *poisoning availability attacks*, the attacker controls a certain amount of training data, thus influencing the trained model and ultimately the predictions at testing time on most points in testing set [4, 18, 20, 28–30, 34, 36, 41, 48]. *Poisoning integrity attacks* have the goal of modifying predictions on a few targeted points by manipulating the training process [20, 41]. On the other hand, *evasion attacks* involve small manipulations of testing data points that results in misprediction at testing time on those points [3, 8, 10, 14, 32, 38, 42, 45, 49].

Creating poisoning and evasion attack points is not a trivial task, particularly when many online services avoid disclosing information about their machine learning algorithms. As a result, attackers are forced to craft their attacks in *black-box* settings, against a surrogate model instead of the real model used by the service, hoping that the attack will be effective on the real model. The *transferability* property of an attack is satisfied when an attack developed for a particular machine learning model (i.e., a surrogate model) is also effective against the target model. Attack transferability was observed in early studies on adversarial examples [14, 42] and has gained a lot more interest in recent years with the advancement of machine learning cloud services. Previous work has reported empirical findings about the transferability of evasion attacks [3, 13, 14, 21, 26, 32, 33, 42, 43, 47] and, only recently, also on the transferability of poisoning integrity attacks [41]. In spite of these efforts, the question of *when and why do adversarial points transfer* remains largely unanswered.

In this paper we present the first comprehensive evaluation of transferability of evasion and poisoning availability attacks, understanding the factors contributing to transferability of both attacks. In particular, we consider attacks crafted with gradient-based optimization techniques (e.g., [4, 8, 23]), a popular and successful mechanism used to create attack data points. We unify for the first time evasion and poisoning attacks into an optimization framework that can be instantiated for a range of threat models and adversarial constraints. We provide a formal definition of transferability and show that, under linearization of the loss function computed under attack, several main factors impact transferability: the intrinsic *adversarial vulnerability* of the target model, the *complexity* of the surrogate model used to optimize the attacks, and its alignment with the target model. Furthermore, we derive a new poisoning attack for logistic regression, and perform a comprehensive evaluation of both evasion and poisoning attacks on multiple datasets, confirming our theoretical analysis.

In more detail, the contributions of our work are:

Optimization framework for evasion and poisoning attacks. We introduce a unifying framework based on gradient-

descent optimization that encompasses both evasion and poisoning attacks. Our framework supports threat models with different adversarial goals (integrity and availability), amount of knowledge available to the adversary (white-box and black-box), as well as different adversarial capabilities (causative or exploratory). Our framework generalizes existing attacks proposed by previous work for evasion [3, 8, 14, 23, 42] and poisoning [4, 18, 20, 24, 27, 48]. Under our framework, we derive a novel gradient-based poisoning availability attack against logistic regression. We remark here that poisoning attacks are more difficult to derive than evasion ones, as they require computing hypergradients from a bilevel optimization problem, to capture the dependency on how the machine-learning model changes while the training poisoning points are modified [4, 18, 20, 24, 27, 48].

Transferability definition and theoretical bound. We give a formal definition of transferability of evasion and poisoning attacks, and an upper bound on a transfer attack's success. This allows us to derive three metrics connected to *model complexity*. Our formal definition unveils that transferability depends on: (1) the size of input gradients of the target classifier; (2) how well the gradients of the surrogate and target models align; and (3) the variance of the loss landscape optimized to generate the attack points.

Comprehensive experimental evaluation of transferability. We consider a wide range of classifiers, including logistic regression, SVMs with both linear and RBF kernels, ridge regression, random forests, and deep neural networks (both feed-forward and convolutional neural networks), all with different hyperparameter settings to reflect different model complexities. We evaluate the transferability of our attacks on three datasets related to different applications: handwritten digit recognition (MNIST), Android malware detection (DREBIN), and face recognition (LFW). We confirm our theoretical analysis for both evasion and poisoning attacks.

Insights into transferability. We demonstrate that attack transferability depends strongly on the *complexity* of the target model, i.e., on its inherent vulnerability. This confirms that reducing the size of input gradients, e.g., via regularization, may allow us to learn more robust classifiers not only against evasion [22, 35, 39, 44] but also against poisoning availability attacks. Second, transferability is also impacted by the surrogate model's alignment with the target model. Surrogates with better alignments to their targets (in terms of the angle between their gradients) are more successful at transferring the attack points. Third, surrogate loss functions that are stabler and have lower variance tend to facilitate gradient-based optimization attacks to find better local optima (see Figure 1). As less complex models exhibit a lower variance of their loss function, they typically result in better surrogates.

Organization. We discuss background on threat modeling against machine learning in Section 2. We introduce our unifying optimization framework for evasion and poisoning attacks,
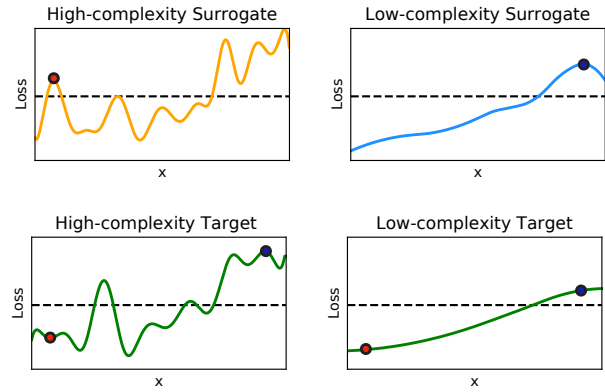

Figure 1: Conceptual representation of transferability. We show the loss function of the attack objective as a function of a single feature x. The top row includes 2 surrogate models (*high* and *low* complexity), while the bottom row includes both models as targets. The adversarial samples are represented as red dots for the high-complexity surrogate and as blue dots for the low-complexity surrogate. If the adversarial sample loss is below a certain threshold (i.e., the black horizontal line), the point is correctly classified, otherwise it is misclassified. The adversarial point computed against the high-complexity model (top left) lays in a local optimum due to the irregularity of the objective. This point is not effective even against the same classifier trained on a different dataset (bottom left) due to the variance of the high-complexity classifier. The adversarial point computed against the low complexity model (top right), instead, succeeds against both low and high-complexity targets (left and right bottom, respectively).

as well as the poisoning attack for logistic regression in Section 3. We then formally define transferability for both evasion and poisoning attacks, and show its approximate connection with the input gradients used to craft the corresponding attack samples (Section 4). Experiments are reported in Section 5, highlighting connections among regularization hyperparameters, the size of input gradients, and transferability of attacks, on different case studies involving handwritten digit recognition, Android malware detection, and face recognition. We discuss related work in Section 6 and conclude in Section 7.

2 Background and Threat Model

Supervised learning includes: (1) a training phase in which training data is given as input to a learning algorithm, resulting in a trained ML model; (2) a testing phase in which the model is applied to new data and a prediction is generated. In this paper, we consider a range of adversarial models against machine learning classifiers at both training and testing time. Attackers are defined by: (*i*) their goal or objective in attacking the system; (*ii*) their knowledge of the system; (*iii*) their capabilities in influencing the system through manipulation

of the input data. Before we detail each of these, we introduce our notation, and point out that the threat model and attacks considered in this work are suited to binary classification, but can be extended to multi-class settings.

Notation. We denote the sample and label spaces with $\mathcal{X}$ and $\mathcal{Y} \in \{-1, +1\}$, respectively, and the training data with $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^{n}$, where n is the training set size. We use $L(\mathcal{D}, \mathbf{w})$ to denote the *loss* incurred by classifier $f : \mathcal{X} \mapsto \mathcal{Y}$ (parameterized by $\mathbf{w}$) on $\mathcal{D}$. Typically, this is computed by averaging a loss function $\ell(y, \mathbf{x}, \mathbf{w})$ computed on each data point, i.e., $L(\mathcal{D}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i, \mathbf{w})$. We assume that the classifier f is learned by minimizing an objective function $\mathcal{L}(\mathcal{D}, \mathbf{w})$ on the training data. Typically, this is an estimate of the generalization error, obtained by the sum of the empirical loss L on training data $\mathcal{D}$ and a regularization term.

2.1 Threat Model: Attacker's Goal

We define the attacker's goal based on the desired security violation. In particular, the attacker may aim to cause either an *integrity* violation, to evade detection without compromising normal system operation; or an *availability* violation, to compromise the normal system functionalities available to legitimate users.

2.2 Threat Model: Attacker's Knowledge

We characterize the attacker's knowledge κ as a tuple in an abstract knowledge space $\mathcal{K}$ consisting of four main dimensions, respectively representing knowledge of: (*k.i*) the training data $\mathcal{D}$; (*k.ii*) the feature set $\mathcal{X}$; (*k.iii*) the learning algorithm f, along with the objective function $\mathcal{L}$ minimized during training; and (*k.iv*) the parameters $\mathbf{w}$ learned after training the model. This categorization enables the definition of many different kinds of attacks, ranging from *white-box* attacks with full knowledge of the target classifier to *black-box* attacks in which the attacker has limited information about the target system.

White-Box Attacks. We assume here that the attacker has full knowledge of the target classifier, i.e., $\kappa = (\mathcal{D}, \mathcal{X}, f, \mathbf{w})$. This setting allows one to perform a worst-case evaluation of the security of machine-learning algorithms, providing empirical upper bounds on the performance degradation that may be incurred by the system under attack.

Black-Box Attacks. We assume here that the input feature representation $\mathcal{X}$ is known. For images, this means that we do consider pixels as the input features, consistently with other recent work on black-box attacks against machine learning [32, 33]. At the same time, the training data $\mathcal{D}$ and the type of classifier f are not known to the attacker. We consider the most realistic attack model in which the attacker does not have querying access to the classifier.

The attacker can collect a surrogate dataset $\hat{\mathcal{D}}$, ideally sampled from the same underlying data distribution as $\mathcal{D}$, and train a *surrogate model* $\hat{f}$ on such data to approximate the target function f. Then, the attacker can craft the attacks against $\hat{f}$, and then check whether they successfully *transfer* to the target classifier f. By denoting limited knowledge of a given component with the *hat* symbol, such black-box attacks can be denoted with $\hat{\kappa} = (\hat{\mathcal{D}}, \mathcal{X}, \hat{f}, \hat{\mathbf{w}})$.

2.3 Threat Model: Attacker's Capability

This attack characteristic defines how the attacker can influence the system, and how data can be manipulated based on application-specific constraints. If the attacker can manipulate both training and test data, the attack is said to be *causative*. It is instead referred to as *exploratory*, if the attacker can only manipulate test data. These scenarios are more commonly known as *poisoning* [4, 18, 24, 27, 48] and *evasion* [3, 8, 14, 42].

Another aspect related to the attacker's capability depends on the presence of application-specific constraints on data manipulation; e.g., to evade malware detection, malicious code has to be modified without compromising its intrusive functionality. This may be done against systems leveraging static code analysis, by injecting instructions that will never be executed [11, 15, 45]. These constraints can be generally accounted for in the definition of the optimal attack strategy by assuming that the initial attack sample $\mathbf{x}$ can only be modified according to a space of possible modifications $\Phi(\mathbf{x})$.

3 Optimization Framework for Gradient-based Attacks

We introduce here a general optimization framework that encompasses both evasion and poisoning attacks. Gradient-based attacks have been considered for evasion (e.g., [3, 8, 14, 23, 42]) and poisoning (e.g., [4, 18, 24, 27]). Our optimization framework not only unifies existing evasion and poisoning attacks, but it also enables the design of new attacks. After defining our general formulation, we instantiate it for evasion and poisoning attacks, and use it to derive a new poisoning availability attack for logistic regression.

3.1 Gradient-based Optimization Algorithm

Given the attacker's knowledge $\kappa \in \mathcal{K}$ and an attack sample $\mathbf{x}' \in \Phi(\mathbf{x})$ along with its label y, the attacker's goal can be defined in terms of an objective function $\mathcal{A}(\mathbf{x}', y, \kappa) \in \mathbb{R}$ (e.g., a loss function) which measures how effective the attack sample $\mathbf{x}'$ is. The optimal attack strategy can be thus given as:

$$\mathbf{x}^\star \in \arg\max_{\mathbf{x}' \in \Phi(\mathbf{x})} \mathcal{A}(\mathbf{x}', y, \kappa). \tag{1}$$

Note that, for the sake of clarity, we consider here the optimization of a single attack sample, but this formulation can be easily extended to account for multiple attack points. In

Algorithm 1 Gradient-based Evasion and Poisoning Attacks

Input: $\mathbf{x}, y$: the input sample and its label; $\mathcal{A}(\mathbf{x}, y, \kappa)$: the attacker's objective; $\kappa = (\mathcal{D}, \mathcal{X}, f, \mathbf{w})$: the attacker's knowledge parameter vector; $\Phi(\mathbf{x})$: the feasible set of manipulations that can be made on $\mathbf{x}$; $t > 0$: a small number.

Output: $\mathbf{x}'$: the adversarial example.

1: Initialize the attack sample: $\mathbf{x}' \leftarrow \mathbf{x}$
2: **repeat**
3:     Store attack from previous iteration: $\mathbf{x} \leftarrow \mathbf{x}'$
4:     Update step: $\mathbf{x}' \leftarrow \Pi_\Phi(\mathbf{x} + \eta \nabla_\mathbf{x} \mathcal{A}(\mathbf{x}, y, \kappa))$, where the step size η is chosen with line search (bisection method), and Π_Φ ensures projection on the feasible domain Φ.
5: **until** $|\mathcal{A}(\mathbf{x}', y, \kappa) - \mathcal{A}(\mathbf{x}, y, \kappa)| \leq t$
6: **return** $\mathbf{x}'$

particular, as in the case of poisoning attacks, the attacker can maximize the objective by iteratively optimizing one attack point at a time [5, 48].

Attack Algorithm. Algorithm 1 provides a general projected gradient-ascent algorithm that can be used to solve the aforementioned problem for both evasion and poisoning attacks. It iteratively updates the attack sample along the gradient of the objective function, ensuring the resulting point to be within the feasible domain through a projection operator Π_Φ. The gradient step size η is determined in each update step using a line-search algorithm based on the bisection method, which solves $\max_\eta \mathcal{A}(\mathbf{x}'(\eta), y, \kappa)$, with $\mathbf{x}'(\eta) = \Pi_\Phi(\mathbf{x} + \eta \nabla_\mathbf{x} \mathcal{A}(\mathbf{x}, y, \kappa))$. For the line search, in our experiments we consider a maximum of 20 iterations. This allows us to reduce the overall number of iterations required by Algorithm 1 to reach a local or global optimum. We also set the maximum number of iterations for Algorithm 1 to 1,000, but convergence (Algorithm 1, line 5) is typically reached only after a hundred iterations.

We finally remark that non-differentiable learning algorithms, like decision trees and random forests, can be attacked with more complex strategies [17, 19] or using gradient-based optimization against a differentiable surrogate learner [31, 37].

3.2 Evasion Attacks

In evasion attacks, the attacker manipulates test samples to have them misclassified, i.e., to evade detection by a learning algorithm. For white-box evasion, the optimization problem given in Eq. (1) can be rewritten as:

$$\max_{\mathbf{x}'} \quad \ell(y, \mathbf{x}', \mathbf{w}), \qquad (2)$$

$$\text{s.t.} \quad \|\mathbf{x}' - \mathbf{x}\|_p \leq \varepsilon, \qquad (3)$$

$$\mathbf{x}_{\text{lb}} \preceq \mathbf{x}' \preceq \mathbf{x}_{\text{ub}}, \qquad (4)$$

where $\|\mathbf{v}\|_p$ is the ℓ_p norm of $\mathbf{v}$, and we assume that the classifier parameters $\mathbf{w}$ are known. For the black-box case, it
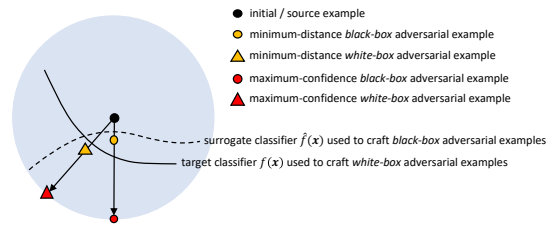

- ● initial / source example
- ○ minimum-distance *black-box* adversarial example
- △ minimum-distance *white-box* adversarial example
- ● maximum-confidence *black-box* adversarial example
- ▲ maximum-confidence *white-box* adversarial example
- --- surrogate classifier $\hat{f}(\mathbf{x})$ used to craft *black-box* adversarial examples
- — target classifier $f(\mathbf{x})$ used to craft *white-box* adversarial examples

Figure 2: Conceptual representation of maximum-confidence evasion attacks (within an ℓ_2 ball of radius ε) vs. minimum-distance adversarial examples. Maximum-confidence attacks tend to transfer better as they are misclassified with higher confidence (though requiring more modifications).

suffices to use the parameters $\hat{\mathbf{w}}$ of the surrogate classifier $\hat{f}$. In this work we consider $\ell(y, \mathbf{x}', \mathbf{w}) = -yf(\mathbf{x}')$, as in [3].

The intuition here is that the attacker maximizes the loss on the adversarial sample with the original class, to cause misclassification to the opposite class. The manipulation constraints $\Phi(\mathbf{x})$ are given in terms of: (*i*) a distance constraint $\|\mathbf{x}' - \mathbf{x}\|_p \leq \varepsilon$, which sets a bound on the maximum input perturbation between $\mathbf{x}$ (i.e., the input sample) and the corresponding modified adversarial example $\mathbf{x}'$; and (*ii*) a box constraint $\mathbf{x}_{\text{lb}} \preceq \mathbf{x}' \preceq \mathbf{x}_{\text{ub}}$ (where $\mathbf{u} \preceq \mathbf{v}$ means that each element of $\mathbf{u}$ has to be not greater than the corresponding element in $\mathbf{v}$), which bounds the values of the attack sample $\mathbf{x}'$.

For images, the former constraint is used to implement either *dense* or *sparse* evasion attacks [12, 25, 37]. Normally, the ℓ_2 and the ℓ_∞ distances between pixel values are used to cause an indistinguishable image blurring effect (by slightly manipulating all pixels). Conversely, the ℓ_1 distance corresponds to a sparse attack in which only few pixels are significantly manipulated, yielding a salt-and-pepper noise effect on the image [12, 37]. The box constraint can be used to bound each pixel value between 0 and 255, or to ensure manipulation of only a specific region of the image. For example, if some pixels should not be manipulated, one can set the corresponding values of $\mathbf{x}_{\text{lb}}$ and $\mathbf{x}_{\text{ub}}$ equal to those of $\mathbf{x}$.

Maximum-confidence vs. minimum-distance evasion. Our formulation of evasion attacks aims to produce adversarial examples that are misclassified with *maximum confidence* by the classifier, within the given space of feasible modifications. This is substantially different from crafting minimum-distance adversarial examples, as formulated in [42] and in follow-up work (e.g., [33]). This difference is conceptually depicted in Fig. 2. In particular, in terms of transferability, it is now widely acknowledged that higher-confidence attacks have better chances of successfully transfering to the target classifier (and even of bypassing countermeasures based on gradient masking) [2, 8, 13]. For this reason, in this work we consider evasion attacks that aim to craft adversarial examples misclassified with *maximum* confidence.

Initialization. There is another factor known to improve trans-

ferability of evasion attacks, as well as their effectiveness in the white-box setting. It consists of running the attack starting from different initialization points to mitigate the problem of getting stuck in poor local optima [3, 13, 50]. In addition to starting the gradient ascent from the initial point $\mathbf{x}$, for non-linear classifiers we also consider starting the gradient ascent from the projection of a randomly-chosen point of the opposite class onto the feasible domain. This double-initialization strategy helps finding better local optima, through the identification of more promising paths towards evasion [13, 47, 50].

3.3 Poisoning Availability Attacks

Poisoning attacks consist of manipulating training data (mainly by injecting adversarial points into the training set) to either favor intrusions without affecting normal system operation, or to purposely compromise normal system operation to cause a denial of service. The former are referred to as poisoning integrity attacks, while the latter are known as poisoning availability attacks [5, 48]. Recent work has mostly addressed transferability of poisoning integrity attacks [41], including backdoor attacks [9, 16]. In this work we focus on poisoning availability attacks, as their transferability properties have not yet been widely investigated. Crafting transferable poisoning availability attacks is much more challenging than crafting transferable poisoning integrity attacks, as the latter have a much more modest goal (modifying prediction on a small set of targeted points).

As for the evasion case, we formulate poisoning in a white-box setting, given that the extension to black-box attacks is immediate through the use of surrogate learners. Poisoning is formulated as a bilevel optimization problem in which the outer optimization maximizes the attacker's objective $\mathcal{A}$ (typically, a loss function L computed on untainted data), while the inner optimization amounts to learning the classifier on the poisoned training data [4, 24, 48]. This can be made explicit by rewriting Eq. (1) as:

$$\max_{\mathbf{x}'} \quad L(\mathcal{D}_{\text{val}}, \mathbf{w}^\star) = \sum_{j=1}^{m} \ell(y_j, \mathbf{x}_j, \mathbf{w}^\star) \qquad (5)$$

$$\text{s.t.} \quad \mathbf{w}^\star \in \arg\min_{\mathbf{w}} L(\mathcal{D}_{\text{tr}} \cup (\mathbf{x}', y), \mathbf{w}) \qquad (6)$$

where $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{val}}$ are the training and validation datasets available to the attacker. The former, along with the poisoning point $\mathbf{x}'$, is used to train the learner on poisoned data, while the latter is used to evaluate its performance on untainted data, through the loss function $L(\mathcal{D}_{\text{val}}, \mathbf{w}^\star)$. Notably, the objective function implicitly depends on $\mathbf{x}'$ through the parameters $\mathbf{w}^\star$ of the poisoned classifier.

The attacker's capability is limited by assuming that the attacker can inject only a small fraction α of poisoning points into the training set. Thus, the attacker solves an optimization problem involving a set of poisoned data points (αn) added to the training data.

Poisoning points can be optimized via gradient-ascent procedures, as shown in Algorithm 1. The main challenge is to compute the gradient of the attacker's objective (i.e., the validation loss) with respect to each poisoning point. In fact, this gradient has to capture the implicit dependency of the optimal parameter vector $\mathbf{w}^\star$ (learned after training) on the poisoning point being optimized, as the classification function changes while this point is updated. Provided that the attacker function is differentiable w.r.t. $\mathbf{w}$ and $\mathbf{x}$, the required gradient can be computed using the chain rule [4, 5, 24, 27, 48]:

$$\nabla_{\mathbf{x}} \mathcal{A} = \nabla_{\mathbf{x}} L + \frac{\partial \mathbf{w}}{\partial \mathbf{x}}^\top \nabla_{\mathbf{w}} L, \qquad (7)$$

where the term $\frac{\partial \mathbf{w}}{\partial \mathbf{x}}$ captures the implicit dependency of the parameters $\mathbf{w}$ on the poisoning point $\mathbf{x}$. Under some regularity conditions, this derivative can be computed by replacing the inner optimization problem with its stationarity (Karush-Kuhn-Tucker, KKT) conditions, i.e., with its implicit equation $\nabla_{\mathbf{w}} L(\mathcal{D}_{\text{tr}} \cup (\mathbf{x}', y), \mathbf{w}) = \mathbf{0}$ [24, 27].[1] By differentiating this expression w.r.t. the poisoning point $\mathbf{x}$, one yields:

$$\nabla_{\mathbf{x}} \nabla_{\mathbf{w}} L + \frac{\partial \mathbf{w}}{\partial \mathbf{x}}^\top \nabla_{\mathbf{w}}^2 L = \mathbf{0}. \qquad (8)$$

Solving for $\frac{\partial \mathbf{w}}{\partial \mathbf{x}}$, we obtain $\frac{\partial \mathbf{w}}{\partial \mathbf{x}}^\top = -(\nabla_{\mathbf{x}} \nabla_{\mathbf{w}} L)(\nabla_{\mathbf{w}}^2 L)^{-1}$, which can be substituted in Eq. (7) to obtain the required gradient:

$$\nabla_{\mathbf{x}} \mathcal{A} = \nabla_{\mathbf{x}} L - (\nabla_{\mathbf{x}_c} \nabla_{\mathbf{w}} L)(\nabla_{\mathbf{w}}^2 L)^{-1} \nabla_{\mathbf{w}} L. \qquad (9)$$

Gradients for SVM. Poisoning attacks against SVMs were first proposed in [4]. Here, we report a simplified expression for SVM poisoning, with $\mathcal{L}$ corresponding to the dual SVM learning problem, and L to the hinge loss (in the outer optimization):

$$\nabla_{\mathbf{x}_c} \mathcal{A} = -\alpha_c \frac{\partial \mathbf{k}_{kc}}{\partial \mathbf{x}_c} \mathbf{y}_k + \alpha_c \begin{bmatrix} \frac{\partial \mathbf{k}_{sc}}{\partial \mathbf{x}_c} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{K}_{ss} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{K}_{sk} \\ \mathbf{1}^\top \end{bmatrix} \mathbf{y}_k. \quad (10)$$

We use c, s and k here to respectively index the attack point, the support vectors, and the validation points for which $\ell(y, \mathbf{x}, \mathbf{w}) > 0$ (corresponding to a non-null derivative of the hinge loss). The coefficient α_c is the dual variable assigned to the poisoning point by the learning algorithm, and $\mathbf{k}$ and $\mathbf{K}$ contain kernel values between the corresponding indexed sets of points.

Gradients for Logistic Regression. Logistic regression is a linear classifier that estimates the probability of the positive class using the sigmoid function. A poisoning attack against logistic regression has been derived in [24], but maximizing a different outer objective and not directly the validation loss.

---

[1]More rigorously, we should write the KKT conditions in this case as $\nabla_{\mathbf{w}} L(\mathcal{D}_{\text{tr}} \cup (\mathbf{x}', y), \mathbf{w}) \in \mathbf{0}$, as the solution may not be unique.

One of our contributions is to compute gradients for logistic regression under our optimization framework. Using logistic loss as the attacker's loss, the poisoning gradient for logistic regression can be computed as:

$$\nabla_{\mathbf{x}_c}\mathcal{A} = -\begin{bmatrix} \nabla_{\mathbf{x}_c}\nabla_{\theta}\mathcal{L} \\ C\,z_c\,\theta \end{bmatrix}^{\top} \begin{bmatrix} \nabla_{\theta}^2\mathcal{L} & \mathbf{X}\,\mathbf{z}\,C \\ C\,\mathbf{z}\,\mathbf{X} & C\sum_i^n z_i \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}(\mathbf{y}\circ\sigma-\mathbf{y}) \\ \mathbf{y}^{\top}(\sigma-1) \end{bmatrix} C,$$

where θ are the classifier weights (bias excluded), $\circ$ is the element-wise product, $\mathbf{z}$ is equal to $\sigma(1-\sigma)$, σ is the sigmoid of the signed discriminant function (each element of that vector is therefore: $\sigma_i = \frac{1}{1+\exp(-y_i f_i)}$ with $f_i = \mathbf{x}_i\theta + b$), and:

$$\nabla_{\theta}^2\mathcal{L} = C\sum_i^n \mathbf{x_i}z_i\mathbf{x}_i^{\top} + \mathbb{I}, \qquad (11)$$

$$\nabla_{\mathbf{x}_c}\nabla_{\theta}\mathcal{L} = C(\mathbb{I}\circ(y_c\sigma_c - y_c) + z_c\theta\mathbf{x}_c^{\top}) \qquad (12)$$

In the above equations, $\mathbb{I}$ is the identity matrix.

4 Transferability Definition and Metrics

We discuss here an intriguing connection among transferability of both evasion and poisoning attacks, input gradients and model complexity, and highlight the factors impacting transferability between a surrogate and a target model. Model complexity is a measure of the capacity of a learning algorithm to fit the training data. It is typically penalized to avoid overfitting by reducing either the number of classifier parameters to be learnt or their size (e.g., via regularization) [6]. Given that complexity is essentially controlled by the hyperparameters of a given learning algorithm (e.g., the number of neurons in the hidden layers of a neural network, or the regularization hyperparameter C of an SVM), *only models that are trained using the same learning algorithm should be compared in terms of complexity*. As we will see, this is an important point to correctly interpret the results of our analysis. For notational convenience, we denote in the following the attack points as $\mathbf{x}^{\star} = \mathbf{x} + \hat{\delta}$, where $\mathbf{x}$ is the initial point and $\hat{\delta}$ the adversarial perturbation optimized by the attack algorithm against the *surrogate* classifier, for both evasion and poisoning attacks. We start by formally defining transferability for evasion attacks, and then discuss how this definition and the corresponding metrics can be generalized to poisoning.

Transferability of Evasion Attacks. Given an evasion attack point $\mathbf{x}^{\star}$, crafted against a surrogate learner (parameterized by $\hat{\mathbf{w}}$), we define its *transferability* as the loss attained by the target classifier f (parameterized by $\mathbf{w}$) on that point, i.e., $T = \ell(y, \mathbf{x} + \hat{\delta}, \mathbf{w})$. This can be simplified through a linear approximation of the loss function as:

$$T = \ell(y, \mathbf{x} + \hat{\delta}, \mathbf{w}) \cong \ell(y, \mathbf{x}, \mathbf{w}) + \hat{\delta}^{\top}\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \mathbf{w}). \qquad (13)$$

This approximation may not only hold for sufficiently-small input perturbations. It may also hold for larger perturbations

if the classification function is linear or has a small curvature (e.g., if it is strongly regularized). It is not difficult to see that, for any given point $\mathbf{x}, y$, the evasion problem in Eqs. (2)-(3) (without considering the feature bounds in Eq. 4) can be rewritten as:

$$\hat{\delta} \in \arg\max_{\|\delta\|_p \leq \varepsilon} \ell(y, \mathbf{x} + \delta, \hat{\mathbf{w}}). \qquad (14)$$

Under the same linear approximation, this corresponds to the maximization of an inner product over an ε-sized ball:

$$\max_{\|\delta\|_p \leq \varepsilon} \delta^{\top}\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \hat{\mathbf{w}}) = \varepsilon\|\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \hat{\mathbf{w}})\|_q, \qquad (15)$$

where ℓ_q is the dual norm of ℓ_p.

The above problem is maximized as follows:

1. For $p = 2$, the maximum is $\hat{\delta} = \varepsilon\frac{\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \hat{\mathbf{w}})}{\|\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \hat{\mathbf{w}})\|_2}$;

2. For $p = \infty$, the maximum is $\hat{\delta} \in \varepsilon\cdot\text{sign}\{\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \hat{\mathbf{w}})\}$;

3. For $p = 1$, the maximum is achieved by setting the values of $\hat{\delta}$ that correspond to the maximum absolute values of $\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \hat{\mathbf{w}})$ to their sign, i.e., ± 1, and 0 otherwise.

Substituting the optimal value of $\hat{\delta}$ into Eq. (13), we can compute the loss increment $\Delta\ell = \hat{\delta}^{\top}\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \mathbf{w})$ under a transfer attack in closed form; e.g., for $p = 2$, it is given as:

$$\Delta\ell = \varepsilon\frac{\nabla_{\mathbf{x}}\hat{\ell}^{\top}}{\|\nabla_{\mathbf{x}}\hat{\ell}\|_2}\nabla_{\mathbf{x}}\ell \leq \varepsilon\|\nabla_{\mathbf{x}}\ell\|_2, \qquad (16)$$

where, for compactness, we use $\hat{\ell} = \ell(y, \mathbf{x}, \hat{\mathbf{w}})$ and $\ell = \ell(y, \mathbf{x}, \mathbf{w})$. In this equation, the left-hand side is the increase in the loss function in the black-box case, while the right-hand side corresponds to the white-box case. The upper bound is obtained when the surrogate classifier $\hat{\mathbf{w}}$ is equal to the target $\mathbf{w}$ (white-box attacks). Similar results hold for $p = 1$ and $p = \infty$ (using the dual norm in the right-hand side).

Intriguing Connections and Transferability Metrics. The above findings reveal some interesting connections among transferability of attacks, model complexity (controlled by the classifier hyperparameters) and input gradients, as detailed below, and allow us to define simple and computationally-efficient transferability metrics.

(1) Size of Input Gradients. The first interesting observation is that transferability depends on the size of the gradient of the loss ℓ computed using the *target* classifier, regardless of the surrogate: the larger this gradient is, the larger the attack impact may be. This is inferred from the upper bound in Eq. (16). We define the corresponding metric $S(\mathbf{x}, y)$ as:

$$S(\mathbf{x}, y) = \|\nabla_{\mathbf{x}}\ell(y, \mathbf{x}, \mathbf{w})\|_q, \qquad (17)$$

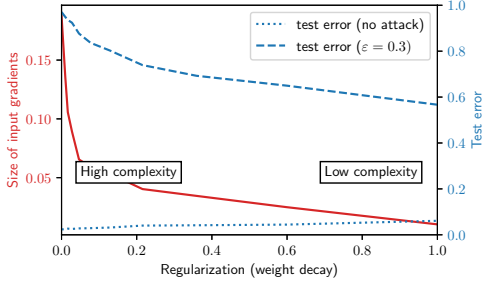where q is the dual of the perturbation norm.

Figure 3: Size of input gradients (averaged on the test set) and test error (in the absence and presence of evasion attacks) against regularization (controlled via weight decay) for a neural network trained on MNIST89 (see Sect. 5.1.1). Note how the size of input gradients and the test error under attack decrease as regularization (complexity) increases (decreases).

The size of the input gradient also depends on the complexity of the given model, controlled, e.g., by its regularization hyperparameter. Less complex, strongly-regularized classifiers tend to have smaller input gradients, i.e., they learn smoother functions that are more robust to attacks, and vice-versa. Notably, this holds for both evasion and poisoning attacks (e.g., the poisoning gradient in Eq. 10 is proportional to α_c, which is larger when the model is weakly regularized). In Fig. 3 we report an example showing how increasing regularization (i.e., decreasing complexity) for a neural network trained on MNIST89 (see Sect. 5.1.1), by controlling its *weight decay*, reduces the average size of its input gradients, improving adversarial robustness to evasion. It is however worth remarking that, since complexity is a model-dependent characteristic, the size of input gradients cannot be directly compared across different learning algorithms; e.g., if a linear SVM exhibits larger input gradients than a neural network, we cannot conclude that the former will be more vulnerable.

Another interesting observation is that, if a classifier has large input gradients (e.g., due to high-dimensionality of the input space and low level of regularization), for an attack to succeed it may suffice to apply only tiny, *imperceptible* perturbations. As we will see in the experimental section, this explains why adversarial examples against deep neural networks can often only be slightly perturbed to mislead detection, while when attacking less complex classifiers in low dimensions, modifications become more evident.

(2) Gradient Alignment. The second relevant impact factor on transferability is based on the alignment of the input gradients of the loss function computed using the target and the surrogate learners. If we compare the increase in the loss function in the black-box case (the left-hand side of Eq. 16) against that corresponding to white-box attacks (the right-hand side of Eq. 16), we find that the relative increase in loss, at least for ℓ_2 perturbations, is given by the following value:

$$R(\mathbf{x}, y) = \frac{\nabla_{\mathbf{x}} \hat{\ell}^{\top} \nabla_{\mathbf{x}} \ell}{\|\nabla_{\mathbf{x}} \hat{\ell}\|_2 \|\nabla_{\mathbf{x}} \ell\|_2} . \qquad (18)$$
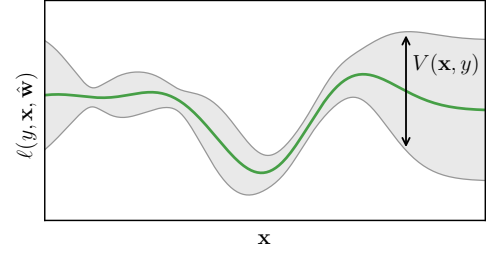

Figure 4: Conceptual representation of the variability of the loss landscape. The green line represents the expected loss with respect to different training sets used to learn the surrogate model, while the gray area represents the variance of the loss landscape. If the variance is too large, local optima may change, and the attack may not successfully transfer.

Interestingly, this is exactly the cosine of the angle between the gradient of the loss of the surrogate and that of the target classifier. This is a novel finding which explains why the cosine angle metric between the target and surrogate gradients can well characterize the transferability of attacks, confirming empirical results from previous work [21]. For other kinds of perturbation, this definition slightly changes, but gradient alignment can be similarly evaluated. Differently from the gradient size S, gradient alignment is a pairwise metric, allowing comparisons across different surrogate models; e.g., if a surrogate SVM is better aligned with the target model than another surrogate, we can expect that attacks targeting the surrogate SVM will transfer better.

(3) Variability of the Loss Landscape. We define here another useful metric to characterize attack transferability. The idea is to measure the variability of the loss function $\hat{\ell}$ when the training set used to learn the surrogate model changes, even though it is sampled from the same underlying distribution. The reason is that the loss $\hat{\ell}$ is exactly the objective function $\mathcal{A}$ optimized by the attacker to craft evasion attacks (Eq. 1). Accordingly, if this loss landscape changes dramatically even when simply resampling the surrogate training set (which may happen, e.g., for surrogate models exhibiting a large error variance, like neural networks and decision trees), it is very likely that the local optima of the corresponding optimization problem will change, and this may in turn imply that the attacks will not transfer correctly to the target learner.

We define the variability of the loss landscape simply as the *variance* of the loss, estimated at a given attack point $\mathbf{x}, y$:

$$V(\mathbf{x}, y) = \mathbb{E}_{\mathcal{D}}\{\ell(y, \mathbf{x}, \hat{\mathbf{w}})^2\} - \mathbb{E}_{\mathcal{D}}\{\ell(y, \mathbf{x}, \hat{\mathbf{w}})\}^2, \qquad (19)$$

where $\mathbb{E}_{\mathcal{D}}$ is the expectation taken with respect to different (surrogate) training sets. This is very similar to what is typically done to estimate the variance of classifiers' predictions. This notion is clarified also in Fig. 4. As for the size of input gradients S, also the loss variance V should only be compared across models trained with the same learning algorithm.

The transferability metrics S, R and V defined so far depend on the initial attack point $\mathbf{x}$ and its label y. In our experiments, we will compute their mean values by averaging on different initial attack points.

Transferability of Poisoning Attacks. For poisoning attacks, we can essentially follow the same derivation discussed before. Instead of defining transferability in terms of the loss attained on the modified test point, we define it in terms of the validation loss attained by the target classifier under the influence of the poisoning points. This loss function can be linearized as done in the previous case, yielding: $T \cong L(\mathcal{D}, \mathbf{w}) + \hat{\delta}^{\top} \nabla_{\mathbf{x}} L(\mathcal{D}, \mathbf{w})$, where $\mathcal{D}$ are the untainted validation points, and $\hat{\delta}$ is the perturbation applied to the initial poisoning point $\mathbf{x}$ against the surrogate classifier. Recall that L depends on the poisoning point through the classifier parameters $\mathbf{w}$, and that the gradient $\nabla_{\mathbf{x}} L(\mathcal{D}, \mathbf{w})$ here is equivalent to the generic one reported in Eq. (9). It is then clear that the perturbation $\hat{\delta}$ maximizes the (linearized) loss when it is best aligned with its derivative $\nabla_{\mathbf{x}} L(\mathcal{D}, \mathbf{w})$, according to the constraint used, as in the previous case. The three transferability metrics defined before can also be used for poisoning attacks provided that we simply replace the evasion loss $\ell(y, \mathbf{x}, \mathbf{w})$ with the validation loss $L(\mathcal{D}, \mathbf{w})$.

5 Experimental Analysis

In this section, we evaluate the transferability of both evasion and poisoning attacks across a range of ML models. We highlight some interesting findings about transferability, based on the three metrics developed in Sect. 4. In particular, we analyze attack transferability in terms of its connection to the size of the input gradients of the loss function, the gradient alignment between surrogate and target classifiers, and the variability of the loss function optimized to craft the attack points. We provide recommendations on how to choose the most effective surrogate models to craft transferable attacks in the black-box setting.

5.1 Transferability of Evasion Attacks

We start by reporting our experiments on evasion attacks. We consider here two distinct case studies, involving handwritten digit recognition and Android malware detection.

5.1.1 Handwritten Digit Recognition

The MNIST89 data includes the MNIST handwritten digits from classes 8 and 9. Each digit image consists of 784 pixels ranging from 0 to 255, normalized in $[0,1]$ by dividing such values by 255. We run 10 independent repetitions to average the results on different training-test splits. In each repetition, we run white-box and black-box attacks, using $5,900$ samples to train the target classifier, $5,900$ distinct samples to train the surrogate classifier (without even relabeling the surrogate data
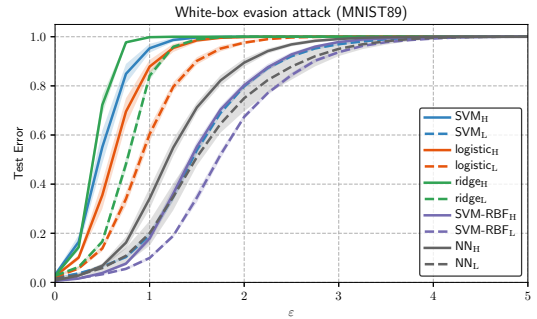

Figure 5: White-box evasion attacks on MNIST89. Test error against increasing maximum perturbation ε.

with labels predicted by the target classifier; i.e., we do not perform any query on the target), and $1,000$ test samples. We modified test digits in both classes using Algorithm 1 under the ℓ_2 distance constraint $\|\mathbf{x} - \mathbf{x}'\|_2 \leq \varepsilon$, with $\varepsilon \in [0,5]$.

For each of the following learning algorithms, we train a high-complexity (H) and a low-complexity (L) model, by changing its hyperparameters: (*i*) SVMs with linear kernel (SVM_H with $C = 100$ and SVM_L with $C = 0.01$); (*ii*) SVMs with RBF kernel ($SVM\text{-}RBF_H$ with $C = 100$ and $SVM\text{-}RBF_L$ with $C = 1$, both with $\gamma = 0.01$); (*iii*) logistic classifiers ($logistic_H$ with $C = 10$ and $logistic_L$ with $C = 1$); (*iv*) ridge classifiers ($ridge_H$ with $\alpha = 1$ and $ridge_L$ with $\alpha = 10$);[2] (*v*) fully-connected neural networks with two hidden layers including 50 neurons each, and ReLU activations (NN_H with no regularization, i.e., weight decay set to 0, and NN_L with weight decay set to 0.01), trained via cross-entropy loss minimization; and (*vi*) random forests consisting of 30 trees (RF_H with no limit on the depth of the trees and RF_L with a maximum depth of 8). These configurations are chosen to evaluate the robustness of classifiers that exhibit similar test accuracies but different levels of complexity.

How does model complexity impact evasion attack success in the white-box setting? The results for white-box evasion attacks are reported for all classifiers that fall under our framework and can be tested for evasion with gradient-based attacks (SVM, Logistic, Ridge, and NN). This excludes random forests, as they are not differentiable. We report the complete *security evaluation curves* [5] in Fig. 5, showing the mean test error (over 10 runs) against an increasing maximum admissible distortion ε. In Fig. 6a we report the mean test error at $\varepsilon = 1$ for each target model against the size of its input gradients (S, averaged on the test samples and on the 10 runs).

The results show that, for each learning algorithm, the low-complexity model has smaller input gradients, and it is less vulnerable to evasion than its high-complexity counterpart, confirming our theoretical analysis. This is also confirmed by the *p*-values reported in Table 1 (first column), obtained by

---

[2]Recall that the level of regularization increases as α increases, and as C decreases.
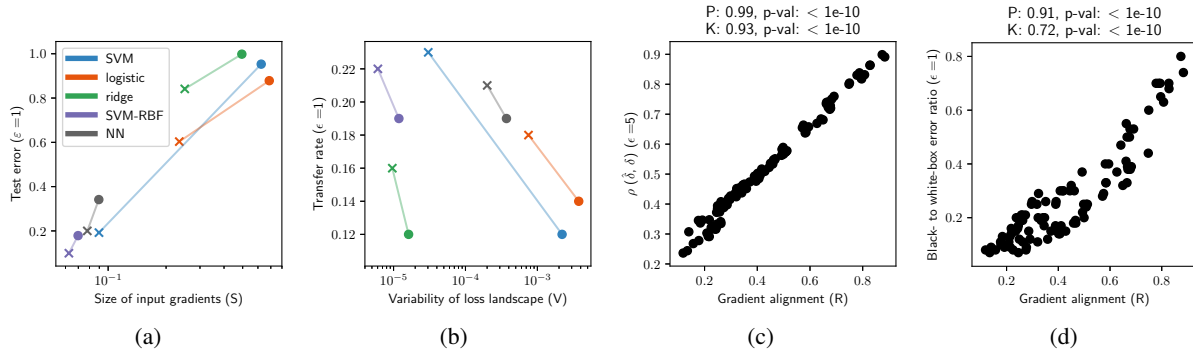
Figure 6: Evaluation of our metrics for evasion attacks on MNIST89. (a) Test error under attack vs average size of input gradients (S) for low- (denoted with '×') and high-complexity (denoted with '○') classifiers. (b) Average transfer rate vs variability of loss landscape (V). (c) Pearson correlation coefficient $\rho(\hat{\delta}, \delta)$ between black-box ($\hat{\delta}$) and white-box (δ) perturbations (values in Fig. 8, right) vs gradient alignment (R, values in Fig. 8, left) for each target-surrogate pair. Pearson (P) and Kendall (K) correlations between ρ and R are also reported along with the p-values obtained from a permutation test to assess statistical significance.

	Evasion				Poisoning			
	MNIST89		DREBIN		MNIST89		LFW	
	$\varepsilon = 1$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 30$	5%	20%	5%	20%
SVM	<1e-2	<1e-2	<1e-2	<1e-2	<1e-2	<1e-2	<1e-2	0.75
logistic	<1e-2	<1e-2	<1e-2	0.02	<1e-2	<1e-2	0.10	0.21
ridge	<1e-2	<1e-2	<1e-2	<1e-2	0.02	<1e-2	0.02	0.75
SVM-RBF	<1e-2	<1e-2	<1e-2	<1e-2	<1e-2	<1e-2	<1e-2	0.11
NN	<1e-2	<1e-2	<1e-2	0.02				

Table 1: Statistical significance of our results. For each attack, dataset and learning algorithm, we report the p-values of two two-sided binomial tests, to respectively reject the null hypothesis that: (*i*) for white-box attacks, the test errors of the high- and low-complexity target follow the same distribution; and (*ii*) for black-box attacks, the transfer rates of the high- and low-complexity surrogate follow the same distribution. Each test is based on 10 samples, obtained by comparing the error of the high- and low-complexity models for each learning algorithm in each repetition. In the first (second) case, success corresponds to a larger test (transfer) error for the high-complexity target (low-complexity surrogate).

running a binomial test for each learning algorithm to compare the white-box test error of the corresponding high- and low-complexity models. All the p-values are smaller than 0.05, which confirms 95% statistical significance. Recall that these results hold only when comparing models trained using the same learning algorithm. This means that we can compare, e.g., the S value of SVM_H against SVM_L, but not that of SVM_H against logistic$_H$. In fact, even though logistic$_H$ exhibits the largest S value, it is not the most vulnerable classifier. Another interesting finding is that nonlinear classifiers tend to be less vulnerable than linear ones.

How do evasion attacks transfer between models in black-box settings? In Fig. 7 we report the results for black-box evasion attacks, in which the attacks against surrogate models (in rows) are transferred to the target models (in columns).

The top row shows results for surrogates trained using only 20% of the surrogate training data, while in the bottom row surrogates are trained using all surrogate data, i.e., a training set of the same size as that of the target. The three columns report results for $\varepsilon \in \{1, 2, 5\}$.

It can be noted that lower-complexity models (with stronger regularization) provide better surrogate models, on average. In particular, this can be seen best in the middle column for medium level of perturbation, in which the lower-complexity models (SVM_L, logistic$_L$, ridge$_L$, and SVM-RBF_L) provide on average higher error when transferred to other models. The reason is that they learn smoother and stabler functions, that are capable of better approximating the target function. Surprisingly, this holds also when using only 20% of training data, as the black-box attacks relying on such low-complexity models still transfer with similar test errors. This means that most classifiers can be attacked in this black-box setting with almost no knowledge of the model, no query access, but provided that one can get a small amount of data similar to that used to train the target model.

These findings are also confirmed by looking at the variability of the loss landscape, computed as discussed in Sect. 4 (by considering 10 different training sets), and reported against the average transfer rate of each surrogate model in Fig. 6b. It is clear from that plot that higher-variance classifiers are less effective as surrogates than their less-complex counterparts, as the former tend to provide worse, unstable approximations of the target classifier. To confirm the statistical significance of this result, for each learning algorithm we also compare the mean transfer errors of high- and low-complexity surrogates with a binomial test whose p-values (always lower than 0.05) are reported in Table 1 (second column).

Another interesting, related observation is that the adversarial examples computed against lower-complexity surrogates have to be perturbed more to evade (see Fig. 9), whereas the perturbation of the ones computed against complex models

Figure 7 — Black-box (transfer) evasion attacks on MNIST89.

(a) ε = 1

Top (surrogate trained with 20% of data):

	SVM_H	SVM_L	logistic_H	logistic_L	ridge_H	ridge_L	SVM-RBF_H	SVM-RBF_L	NN_H	NN_L	RF_H	RF_L	transfer rate
target error	.03	.02	.02	.02	.03	.03	.01	.01	.01	.02	.02	.02	
white box	.96	.19	.89	.60	1.00	.83	.17	.10	.34	.19			
SVM_H	.15	.08	.13	.11	.11	.10	.04	.04	.07	.07	.49	.51	.16
SVM_L	.22	.13	.20	.18	.15	.15	.06	.07	.12	.12	.53	.54	.21
logistic_H	.19	.09	.17	.14	.12	.12	.04	.05	.09	.09	.50	.52	.18
logistic_L	.21	.11	.19	.16	.14	.13	.05	.05	.10	.10	.52	.53	.19
ridge_H	.07	.04	.06	.05	.08	.06	.02	.02	.03	.03	.42	.43	.11
ridge_L	.13	.07	.12	.10	.13	.12	.03	.03	.06	.06	.48	.50	.15
SVM-RBF_H	.19	.11	.17	.15	.13	.13	.06	.06	.11	.11	.52	.53	.19
SVM-RBF_L	.22	.13	.20	.17	.15	.14	.07	.07	.12	.12	.53	.54	.20
NN_H	.20	.10	.18	.15	.13	.12	.05	.05	.10	.10	.51	.52	.18
NN_L	.21	.11	.19	.16	.13	.13	.05	.05	.10	.10	.52	.53	.19

Bottom (surrogate trained with 100% of data):

	SVM_H	SVM_L	logistic_H	logistic_L	ridge_H	ridge_L	SVM-RBF_H	SVM-RBF_L	NN_H	NN_L	RF_H	RF_L	transfer rate
white box	.96	.19	.89	.60	1.00	.83	.17	.10	.31	.21			
SVM_H	.09	.05	.08	.07	.07	.06	.02	.02	.03	.05	.43	.45	.12
SVM_L	.28	.14	.26	.22	.19	.17	.07	.07	.13	.14	.53	.54	.23
logistic_H	.12	.06	.11	.09	.10	.09	.03	.03	.04	.06	.47	.49	.14
logistic_L	.19	.09	.18	.15	.15	.13	.04	.04	.08	.08	.50	.52	.18
ridge_H	.08	.04	.07	.05	.11	.07	.02	.02	.03	.04	.43	.45	.12
ridge_L	.15	.07	.13	.10	.21	.15	.03	.03	.05	.06	.47	.49	.16
SVM-RBF_H	.19	.10	.17	.15	.13	.12	.06	.06	.10	.11	.53	.53	.19
SVM-RBF_L	.25	.13	.23	.20	.17	.16	.08	.08	.14	.14	.53	.54	.22
NN_H	.20	.10	.18	.15	.14	.12	.05	.05	.11	.10	.52	.53	.19
NN_L	.24	.12	.22	.20	.16	.15	.07	.07	.13	.13	.53	.53	.21

(b) ε = 2

Top:

	SVM_H	SVM_L	logistic_H	logistic_L	ridge_H	ridge_L	SVM-RBF_H	SVM-RBF_L	NN_H	NN_L	RF_H	RF_L	transfer rate
white box	1.00	.79	1.00	.98	1.00	1.00	.81	.66	.90	.76			
SVM_H	.52	.35	.51	.50	.51	.46	.22	.17	.37	.29	.52	.53	.41
SVM_L	.72	.66	.73	.74	.79	.78	.49	.47	.63	.55	.56	.57	.64
logistic_H	.62	.46	.62	.62	.66	.61	.32	.26	.48	.37	.53	.54	.51
logistic_L	.68	.55	.69	.69	.74	.71	.39	.33	.55	.45	.54	.55	.57
ridge_H	.18	.07	.16	.12	.27	.18	.04	.04	.07	.07	.48	.48	.18
ridge_L	.43	.25	.43	.41	.69	.63	.17	.13	.28	.21	.51	.52	.39
SVM-RBF_H	.63	.56	.65	.65	.71	.68	.46	.40	.57	.48	.55	.56	.57
SVM-RBF_L	.70	.64	.71	.72	.77	.76	.51	.49	.64	.56	.56	.58	.64
NN_H	.66	.54	.67	.67	.68	.65	.38	.33	.54	.44	.54	.55	.55
NN_L	.68	.56	.69	.69	.72	.70	.40	.35	.56	.46	.54	.55	.58

Bottom:

	SVM_H	SVM_L	logistic_H	logistic_L	ridge_H	ridge_L	SVM-RBF_H	SVM-RBF_L	NN_H	NN_L	RF_H	RF_L	transfer rate
white box	1.00	.79	1.00	.98	1.00	1.00	.81	.66	.90	.73			
SVM_H	.27	.11	.25	.23	.24	.19	.06	.06	.12	.12	.49	.49	.22
SVM_L	.80	.70	.82	.81	.88	.87	.53	.50	.68	.59	.56	.57	.69
logistic_H	.39	.18	.40	.37	.46	.37	.10	.09	.19	.18	.50	.51	.31
logistic_L	.63	.41	.66	.64	.76	.70	.25	.20	.43	.35	.53	.53	.51
ridge_H	.23	.08	.21	.15	.46	.27	.04	.04	.08	.08	.47	.49	.21
ridge_L	.52	.23	.51	.47	.89	.81	.15	.11	.26	.23	.51	.52	.43
SVM-RBF_H	.63	.51	.65	.65	.71	.68	.48	.38	.56	.48	.55	.55	.57
SVM-RBF_L	.76	.66	.77	.77	.85	.83	.58	.53	.69	.60	.57	.57	.68
NN_H	.65	.49	.67	.66	.72	.68	.40	.33	.60	.46	.54	.54	.56
NN_L	.75	.63	.77	.77	.82	.80	.52	.47	.67	.58	.56	.56	.66

(c) ε = 5

Top:

	SVM_H	SVM_L	logistic_H	logistic_L	ridge_H	ridge_L	SVM-RBF_H	SVM-RBF_L	NN_H	NN_L	RF_H	RF_L	transfer rate
white box	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00			
SVM_H	.98	.97	.99	.99	1.00	1.00	.92	.93	.97	.94	.59	.58	.91
SVM_L	1.00	1.00	1.00	1.00	1.00	1.00	.99	1.00	1.00	1.00	.75	.76	.96
logistic_H	.99	.99	1.00	1.00	1.00	1.00	.97	.97	.99	.97	.61	.62	.93
logistic_L	1.00	1.00	1.00	1.00	1.00	1.00	.98	.99	.99	.99	.64	.65	.94
ridge_H	.62	.54	.64	.67	.89	.88	.47	.44	.54	.46	.51	.51	.60
ridge_L	.93	.93	.96	.98	1.00	1.00	.83	.83	.93	.87	.55	.54	.86
SVM-RBF_H	1.00	.99	1.00	1.00	1.00	1.00	.99	.99	1.00	.99	.69	.69	.95
SVM-RBF_L	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.77	.77	.96
NN_H	1.00	.99	1.00	1.00	1.00	1.00	.98	.99	.99	.99	.63	.64	.93
NN_L	1.00	1.00	1.00	1.00	1.00	1.00	.99	.99	1.00	.99	.64	.64	.94

Bottom:

	SVM_H	SVM_L	logistic_H	logistic_L	ridge_H	ridge_L	SVM-RBF_H	SVM-RBF_L	NN_H	NN_L	RF_H	RF_L	transfer rate
white box	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00			
SVM_H	.83	.72	.89	.90	.94	.96	.58	.59	.71	.60	.53	.52	.73
SVM_L	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.75	.73	.96
logistic_H	.94	.86	.97	.97	.99	.99	.71	.73	.87	.75	.55	.55	.82
logistic_L	.99	.98	1.00	1.00	1.00	1.00	.95	.95	.99	.95	.62	.60	.92
ridge_H	.71	.56	.74	.75	.98	.97	.46	.42	.57	.50	.51	.51	.64
ridge_L	.97	.92	.98	.98	1.00	1.00	.81	.81	.92	.82	.56	.54	.86
SVM-RBF_H	.99	.99	1.00	1.00	1.00	1.00	.99	.99	.99	.98	.72	.69	.94
SVM-RBF_L	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.78	.75	.96
NN_H	.99	.98	1.00	1.00	1.00	1.00	.98	.98	1.00	.98	.67	.64	.94
NN_L	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.75	.72	.95

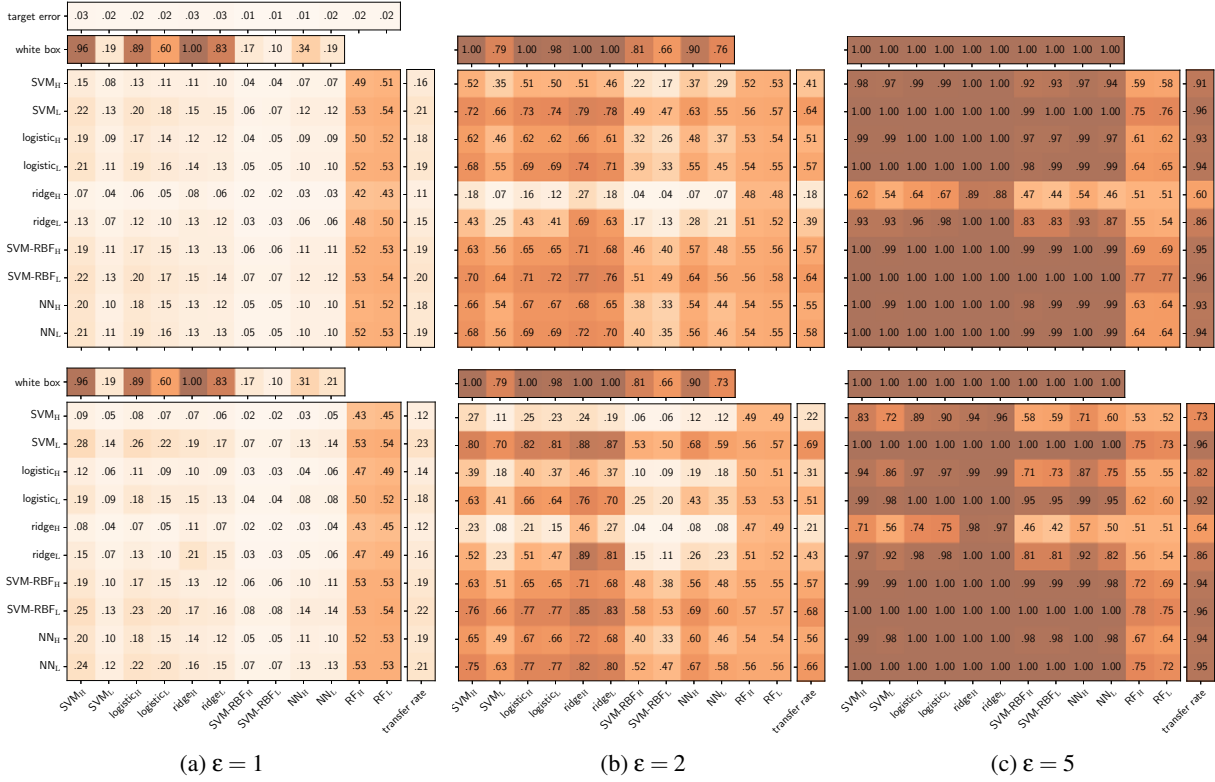(a) ε = 1   (b) ε = 2   (c) ε = 5

Figure 7: Black-box (transfer) evasion attacks on MNIST89. Each cell contains the test error of the target classifier (in columns) computed on the attack samples crafted against the surrogate (in rows). Matrices in the top (bottom) row correspond to attacks crafted against surrogate models trained with 20% (100%) of the surrogate training data, for ε ∈ {1, 2, 5}. The test error of each target classifier in the absence of attack (target error) and under (white-box) attack are also reported for comparison, along with the mean transfer rate of each surrogate across targets. Darker colors mean higher test error, i.e., better transferability.

can be smaller. This is again due to the instability induced by high-complexity models into the loss function optimized to craft evasion attacks, whose sudden changes cause the presence of closer local optima to the initial attack point.

On the vulnerability of random forests. A noteworthy finding is that random forests can be effectively attacked at small perturbation levels using most other models (see last two columns in Fig. 7). We looked at the learned trees and discovered that trees often are susceptible to small changes. In one example, a node of the tree checked if a particular feature value was above 0.002, and classified samples as digit 8 if that condition holds (and digit 9 otherwise). The attack modified that feature from 0 to 0.028, causing it to be immediately misclassified. This vulnerability is intrinsic in the selection process of the threshold values used by these decision trees to split each node. The threshold values are selected among the existing values in the dataset (to correctly handle categorical attributes). Therefore, for pixels which are highly discriminant (e.g., mostly black for one class and white for the other), the threshold will be either very close to one extreme or the other, making it easy to subvert the prediction by a small change. Since ℓ_2-norm attacks change almost all feature values, with high probability the attack modifies at least one feature on every path of the tree, causing misclassification.

Is gradient alignment an effective transferability metric? In Fig. 8, we report on the left the gradient alignment computed between surrogate and target models, and on the right the Pearson correlation coefficient $\rho(\hat{\delta}, \delta)$ between the perturbation optimized against the surrogate (i.e., the black-box perturbation $\hat{\delta}$) and that optimized against the target (i.e., the white-box perturbation δ). We observe immediately that gradient alignment provides an accurate measure of transferability: the higher the cosine similarity, the higher the correlation (meaning that the adversarial examples crafted against the two models are similar). We correlate these two measures in Fig. 6c, and show that such correlation is statistically significant for both Pearson and Kendall coefficients. In Fig. 6d we also correlate gradient alignment with the ratio between the test error of the target model in the black- and white-box setting (extrapolated from the matrix corresponding to ε = 1 in the bottom row of Fig. 7), as suggested by our theoretical derivation. The corresponding permutation tests confirm statistical significance. We finally remark that gradient alignment is extremely fast to evaluate, as it does not require simulating any attack, but it is only a relative measure of the attack trans-
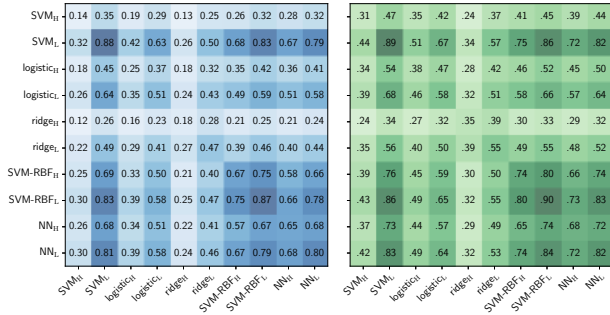
Figure 8: Gradient alignment and perturbation correlation for evasion attacks on MNIST89. *Left:* Gradient alignment R (Eq. 18) between surrogate (rows) and target (columns) classifiers, averaged on the unmodified test samples. *Right:* Pearson correlation coefficient $\rho(\delta, \hat{\delta})$ between white-box and black-box perturbations for $\varepsilon = 5$.

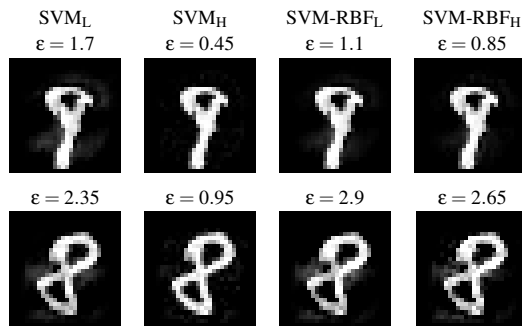ferability, as the latter also depends on the complexity of the target model; i.e., on the size of its input gradients.

Figure 9: Digit images crafted to evade linear and RBF SVMs. The values of ε reported here correspond to the minimum perturbation required to evade detection. Larger perturbations are required to mislead low-complexity classifiers (L), while smaller ones suffice to evade high-complexity classifiers (H).

5.1.2 Android Malware Detection

The Drebin data [1] consists of around 120,000 legitimate and around 5000 malicious Android applications, labeled using the VirusTotal service. A sample is labeled as malicious (or positive, $y = +1$) if it is classified as such from at least five out of ten anti-virus scanners, while it is flagged as legitimate (or negative, $y = -1$) otherwise. The structure and the source code of each application is encoded as a *sparse* feature vector consisting of around a million binary features denoting the presence or absence of permissions, suspicious URLs and other relevant information that can be extracted by statically analyzing Android applications. Since we are working with sparse binary features, we use the ℓ_1 norm for the attack.

We use 30,000 samples to learn surrogate and target classifiers, and the remaining 66,944 samples for testing. The
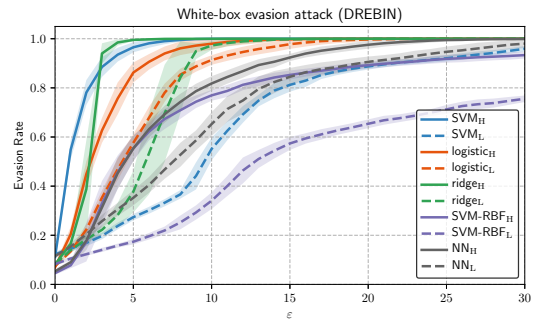
Figure 10: White-box evasion attacks on DREBIN. Evasion rate against increasing maximum perturbation ε.

classifiers and their hyperparameters are the same used for MNIST89, apart from (*i*) the number of hidden neurons for NN_H and NN_L, set to 200, (*ii*) the weight decay of NN_L, set to 0.005; and (*iii*) the maximum depth of RF_L, set to 59.

We perform feature selection to retain those 5,000 features which maximize information gain, i.e., $|p(x_k = 1|y = +1) - p(x_k = 1|y = -1)|$, where x_k is the kth feature. While this feature selection process does not significantly affect the detection rate (which is only reduced by 2%, on average, at 0.5% false alarm rate), it drastically reduces the computational complexity of classification.

In each experiment, we run white-box and black-box evasion attacks on 1,000 distinct malware samples (randomly selected from the test data) against an increasing number of modified features in each malware $\varepsilon \in \{0, 1, 2, \ldots, 30\}$. This is achieved by imposing the ℓ_1 constraint $\|\mathbf{x}' - \mathbf{x}\|_1 \leq \varepsilon$. As in previous work, we further restrict the attacker to only *inject* features into each malware sample, to avoid compromising its intrusive functionality [3, 11].

To evaluate the impact of the aforementioned evasion attack, we measure the evasion rate (i.e., the fraction of malware samples misclassified as legitimate) at 0.5% false alarm rate (i.e., when only 0.5% of the legitimate samples are misclassified as malware). As in the previous experiment, we report the complete *security evaluation curve* for the white-box attack case, whereas we report only the value of test error for the black-box case. The results, reported in Figs. 10, 11, 12, and 13, along with the statistical tests in Table 1 (third and fourth columns) confirm the main findings of the previous experiments. One significant difference is that random forests are much more robust in this case. The reason is that the ℓ_1-norm attack (differently from the ℓ_2) only changes a small number of features, and thus the probability that it will change features in all the ensemble trees is very low.

5.2 Transferability of Poisoning Attacks

For poisoning attacks, we report experiments on handwritten digits and face recognition.
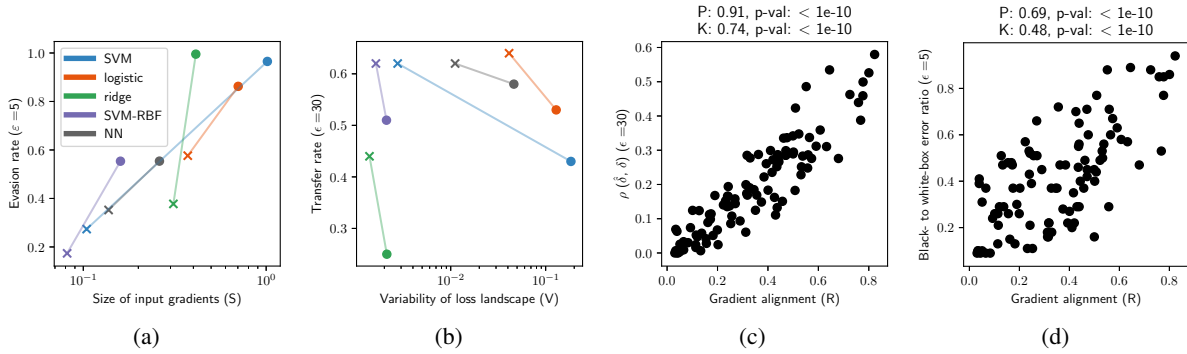
Figure 11: Evaluation of our metrics for evasion attacks on DREBIN. See the caption of Fig. 6 for further details.

5.2.1 Handwritten Digit Recognition

We apply our optimization framework to poison SVM, logistic, and ridge classifiers in the white-box setting. Designing efficient poisoning availability attacks against neural networks is still an open problem due to the complexity of the bilevel optimization and the non-convexity of the inner learning problem. Previous work has mainly considered integrity poisoning attacks against neural networks [5, 20, 41], and it is believed that neural networks are much more resilient to poisoning availability attacks due to their memorization capability. Poisoning random forests is not feasible with gradient-based attacks, and we are not aware of any existing attacks for this ensemble method. We thus consider as surrogate learners: (*i*) linear SVMs with $C = 0.01$ (SVM$_L$) and $C = 100$ (SVM$_H$); (*ii*) logistic classifiers with $C = 0.01$ (logistic$_L$) and $C = 10$ (logistic$_H$); (*iii*) ridge classifiers with $\alpha = 100$ (ridge$_L$) and $\alpha = 10$ (ridge$_H$); and (*iv*) SVMs with RBF kernel with $\gamma = 0.01$ and $C = 1$ (SVM-RBF$_L$) and $C = 100$ (SVM-RBF$_H$). We additionally consider as target classifiers: (*i*) random forests with 100 base trees, each with a maximum depth of 6 for RF$_L$, and with no limit on the maximum depth for RF$_H$; (*ii*) feed-forward neural networks with two hidden layers of 200 neurons each and ReLU activations, trained via cross-entropy loss minimization with different regularization (NN$_L$ with weight decay 0.01 and NN$_H$ with no decay); and (*iii*) the Convolutional Neural Network (CNN) used in [7].

We consider 500 training samples, 1,000 validation samples to compute the attack, and a separate set of 1,000 test samples to evaluate the error. The test error is computed against an increasing number of poisoning points into the training set, from 0% to 20% (corresponding to 125 poisoning points). The reported results are averaged on 10 independent, randomly-drawn data splits.

How does model complexity impact poisoning attack success in the white-box setting? The results for white-box poisoning are reported in Fig. 14. Similarly to the evasion case, high-complexity models (with larger input gradients, as shown in Fig. 15a) are more vulnerable to poisoning attacks than their low-complexity counterparts (i.e., given that the same

learning algorithm is used). This is also confirmed by the statistical tests in the fifth column of Table 1. Therefore, model complexity plays a large role in a model's robustness also against poisoning attacks, confirming our analysis.

How do poisoning attacks transfer between models in black-box settings? The results for black-box poisoning are reported in Fig. 16. For poisoning attacks, the best surrogates are those matching the complexity of the target, as they tend to be better aligned and to share similar local optima, except for low-complexity logistic and ridge surrogates, which seem to transfer better to linear classifiers. This is also witnessed by gradient alignment in Fig. 17, which is again not only correlated to the similarity between black- and white-box perturbations (Fig. 15c), but also to the ratio between the black- and white-box test errors (Fig. 15d). Interestingly, these error ratios are larger than one in some cases, meaning that attacking a surrogate model can be more effective than running a white-box attack against the target. A similar phenomenon has been observed for evasion attacks [33], and it is due to the fact that optimizing attacks against a *smoother* surrogate may find better local optima of the target function (e.g., by overcoming gradient obfuscation [2]). According to our findings, for poisoning attacks, reducing the variability of the loss landscape (V) of the surrogate model is less important than finding a good alignment between the surrogate and the target. In fact, from Fig. 15b it is evident that increasing V is even beneficial for SVM-based surrogates (and all these results are statistically significant according to the *p*-values in the sixth column of Table 1). A visual inspection of the poisoning digits in Fig. 18 reveals that the poisoning points crafted against high-complexity classifiers are only minimally perturbed, while the ones computed against low-complexity classifiers exhibit larger, visible perturbations. This is again due to the presence of closer local optima in the former case. Finally, a surprising result is that RFs are quite robust to poisoning, as well as NNs when attacked with low-complexity linear surrogates. The reason may be that these target classifiers have a large capacity, and can thus fit *outlying* samples (like the digits crafted against low-complexity classifiers in Fig. 18) without affecting the classification of the other training samples.

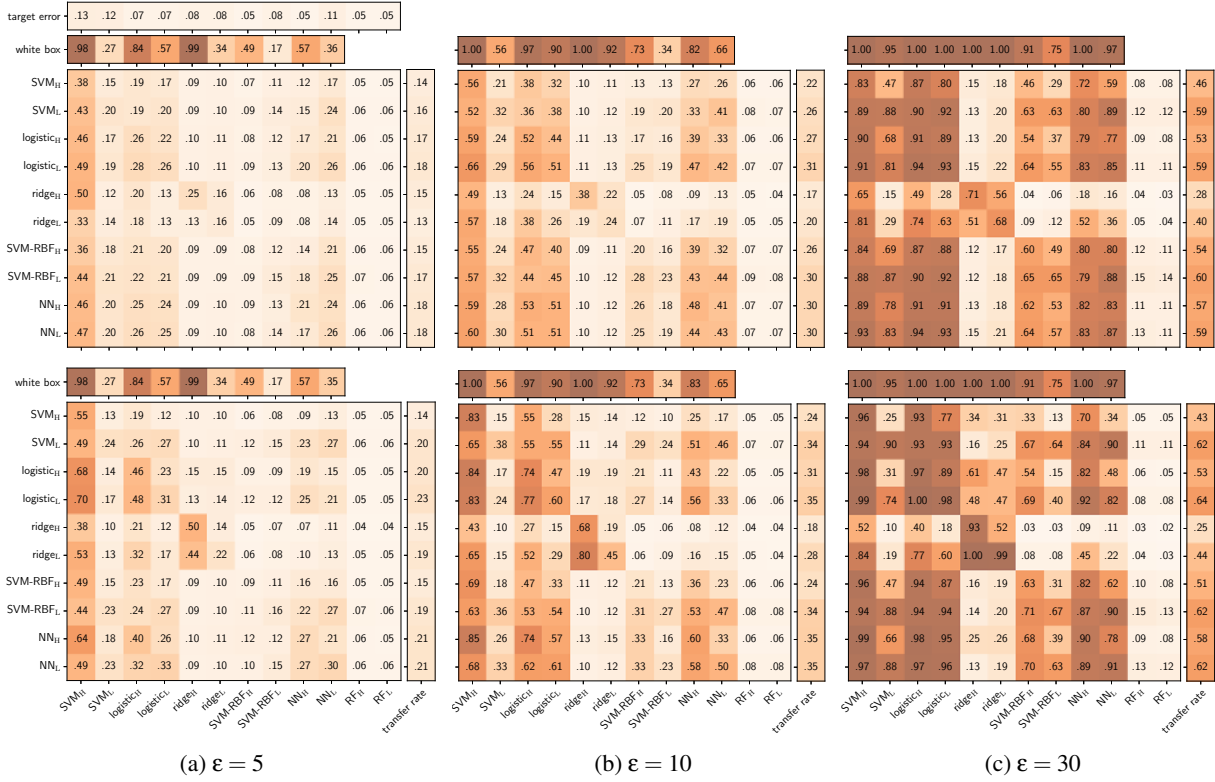(a) ε = 5   (b) ε = 10   (c) ε = 30

Figure 12: Black-box (transfer) evasion attacks on DREBIN. See the caption of Fig. 7 for further details.
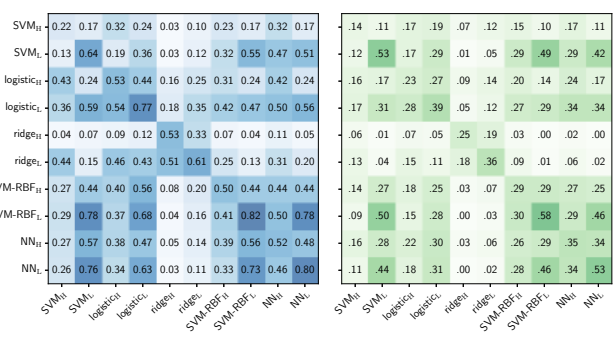
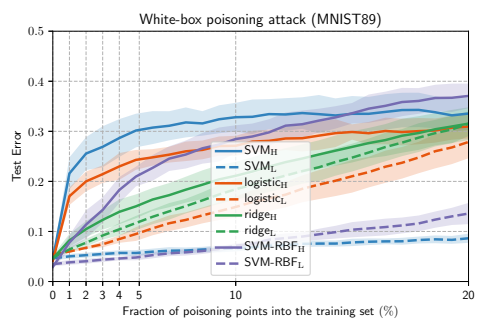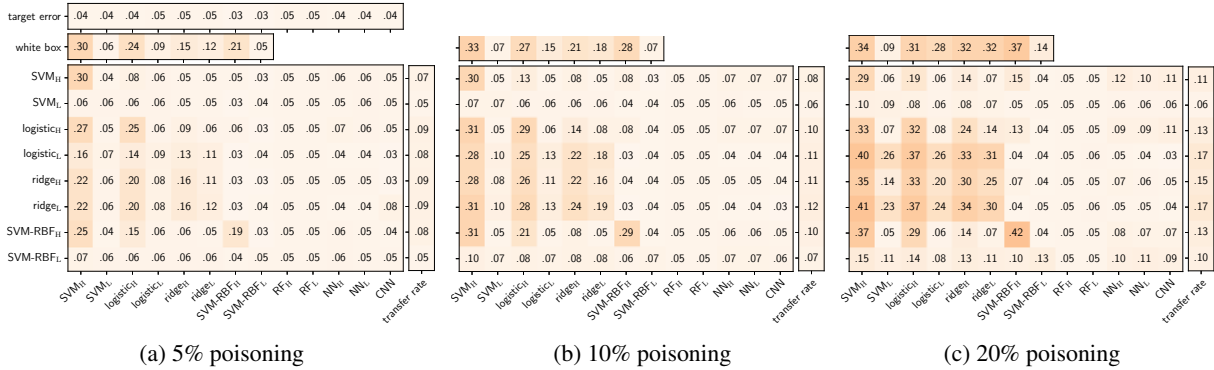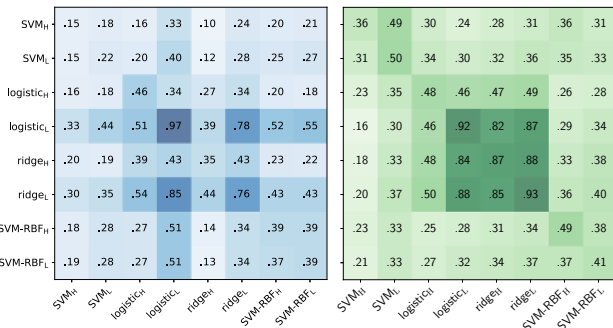Figure 13: Gradient alignment and perturbation correlation (at ε = 30) for evasion attacks on DREBIN. See the caption of Fig. 8 for further details.

Figure 14: White-box poisoning attacks on MNIST89. Test error against an increasing fraction of poisoning points.

5.2.2 Face Recognition

The Labeled Face on the Wild (LFW) dataset consists of faces of famous peoples collected on Internet. We considered the six identities with the largest number of images in the dataset. We considered the person with most images as positive class, and all the others as negative. Our dataset consists of 530 positive and 758 negative images. The classifiers and their hyperparameters are the same used for MNIST89, except that we set: (i) $C = 0.1$ for $logistic_L$, (ii) $\alpha = 1$ for $ridge_H$, (iii) $\gamma = 0.001, C = 10$ for SVM-RBF$_L$, (iv) $\gamma = 0.001, C = 1000$

for SVM-RBF$_H$, and (v) weight decay to 0.001 for NN$_L$. We run 10 repetitions with 300 samples in each training, validation and test set. The results are shown in Figs 19, 20, 21 and 22, confirming the main findings discussed for poisoning attacks on MNIST89. Statistical tests for significance are reported in Table 1 (seventh and eighth columns). In this case, there is not a significant distinction between the mean transfer rates of high- and low-complexity surrogates, probably due to the reduced size of the training sets used. Finally, in Fig. 23 we report examples of perturbed faces against surrogates with different complexities, confirming again that larger perturbations are required to attack lower-complexity models.
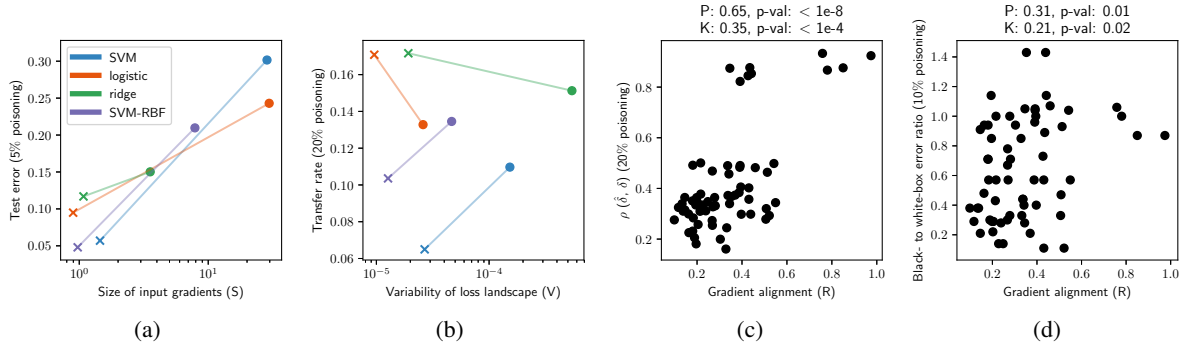
Figure 15: Evaluation of our metrics for poisoning attacks on MNIST89. See the caption of Fig. 6 for further details.

(a) 5% poisoning   (b) 10% poisoning   (c) 20% poisoning

Figure 16: Black-box (transfer) poisoning attacks on MNIST89. See the caption of Fig. 7 for further details.

Figure 17: Gradient alignment and perturbation correlation (at 20% poisoning) for poisoning attacks on MNIST89. See the caption of Fig. 8 for further details.

Figure 18: Poisoning digits crafted against linear and RBF SVMs. Larger perturbations are required to have significant impact on low-complexity classifiers (*L*), while minimal changes are very effective on high-complexity SVMs (*H*).

5.3 Summary of Transferability Evaluation

We summarize the results of transferability for evasion and poisoning attacks below.

(1) Size of input gradients. Low-complexity target classifiers are less vulnerable to evasion and poisoning attacks than high-complexity target classifiers trained with the same learning algorithm, due to the reduced size of their input gradients. In general, nonlinear models are more robust than linear models to both types of attacks.

(2) Gradient alignment. Gradient alignment is correlated

with transferability. Even though it cannot be directly measured in black-box scenarios, some useful guidelines can be derived from our analysis. For evasion attacks, low-complexity surrogate classifiers provide stabler gradients which are better aligned, on average, with those of the target models; thus, it is generally preferable to use strongly-regularized surrogates. For poisoning attacks, instead, gradient alignment tends to improve when the surrogate matches the complexity (regularization) of the target (which may be estimated using techniques from [46]).

(3) Variability of the loss landscape. Low-complexity surro-
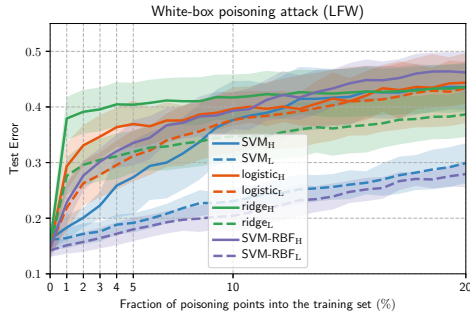
Figure 19: White-box poisoning attacks on LFW. Test error against an increasing fraction of poisoning points.

gate classifiers provide loss landscapes with lower variability than high-complexity surrogate classifiers trained with the same learning algorithm, especially for evasion attacks. This results in better transferability.

To summarize, for evasion attacks, decreasing complexity of the surrogate model by properly adjusting the hyperparameters of its learning algorithm provides adversarial examples that transfer better to a range of models. For poisoning attacks, the best surrogates are generally models with similar levels of regularization as the target model. The reason is that the poisoning objective function is relatively stable (i.e., it has low variance) for most classifiers, and gradient alignment between surrogate and target becomes a more important factor.

Understanding attack transferability has two main implications. First, even when attackers do not know the target classifier, our findings suggest that low-complexity surrogates have a better chance of transferring to other models. Our recommendation to performing black-box evasion attacks is to choose surrogates with low complexity (e.g., by using strong regularization and reducing model variance). To perform poisoning attacks, our recommendation is to acquire additional information about the level of regularization of the target and train a surrogate model with a similar level of regularization. Second, our analysis also provides recommendations to defenders on how to design more robust models against evasion and poisoning attacks. In particular, lower-complexity models tend to have more resilience compared to more complex models. Of course, we need to take into account the bias-variance trade-off and choose models that still perform relatively well on the original prediction tasks.

6 Related Work

Transferability for evasion attacks. Transferability of evasion attacks has been studied in previous work, e.g., [3, 13, 14, 21, 26, 32, 33, 42, 43, 47]. Biggio et al. [3] have been the first to consider evasion attacks against surrogate models in a limited-knowledge scenario. Goodfellow et al. [14], Tramer et al. [43], and Moosavi et al. [26] have made the observation that different models might learn intersecting decision bound-

aries in both benign and adversarial dimensions and in that case adversarial examples transfer better. Tramer et al. have also performed a detailed study of transferability of model-agnostic perturbations that depend only on the training data, noting that adversarial examples crafted against linear models can transfer to higher-order models. We answer some of the open questions they posed about factors contributing to attack transferability. Liu et al. [21] have empirically observed the gradient alignment between models with transferable adversarial examples. Papernot et al. [32, 33] have observed that adversarial examples transfer across a range of models, including logistic regression, SVMs and neural networks, without providing a clear explanation of the phenomenon. Prior work has also investigated the role of input gradients and Jacobians. Some authors have proposed to decrease the magnitude of input gradients during training to defend against evasion attacks [22, 35] or improve classification accuracy [40, 44]. In [35, 39], the magnitude of input gradients has been identified as a cause for vulnerability to evasion attacks. A number of papers have shown that transferability of adversarial examples is increased by averaging the gradients computed for ensembles of models [13, 21, 43, 47]. We highlight that we obtain similar effect by attacking a strongly-regularized surrogate model with a smoother and stabler decision boundary (resulting in a lower-variance model). The advantage of our approach is to reduce the computational complexity compared to attacking classifier ensembles. Through our formalization, we shed light on the most important factors for transferability. In particular, we identify a set of conditions that explain transferability including the gradient alignment between the surrogate and targeted models, and the size of the input gradients of the target model, connected to model complexity. We demonstrate that adversarial examples crafted against lower-variance models (e.g., those that are strongly regularized) tend to transfer better across a range of models.

Transferability for poisoning attacks. There is very little work on the transferability of poisoning availability attacks, except for a preliminary investigation in [27]. That work indicates that poisoning examples are transferable among very simple network architectures (logistic regression, MLP, and Adaline). Transferability of targeted poisoning attacks has been addressed recently in [41]. We are the first to study in depth transferability of poisoning availability attacks.

7 Conclusions

We have conducted an analysis of the transferability of evasion and poisoning attacks under a unified optimization framework. Our theoretical transferability formalization sheds light on various factors impacting the transfer success rates. In particular, we have defined three metrics that impact the transferability of an attack, including the complexity of the target model, the gradient alignment between the surrogate and
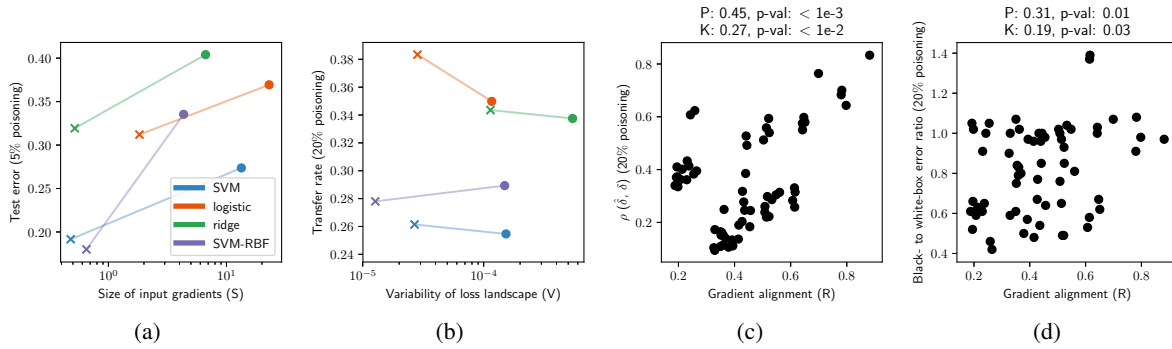
(a) (b) (c) (d)

Figure 20: Evaluation of our metrics for poisoning attacks on LFW. See the caption of Fig. 6 for further details.

(a) 5% poisoning
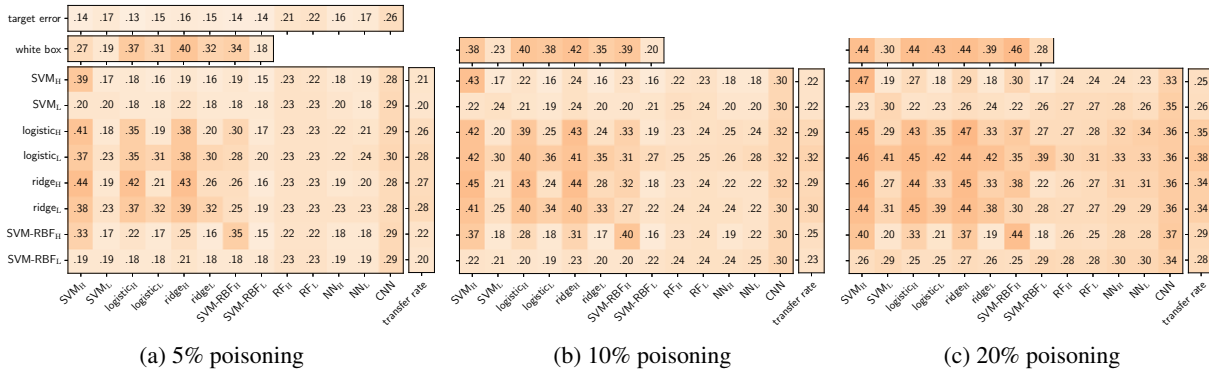
(b) 10% poisoning

(c) 20% poisoning

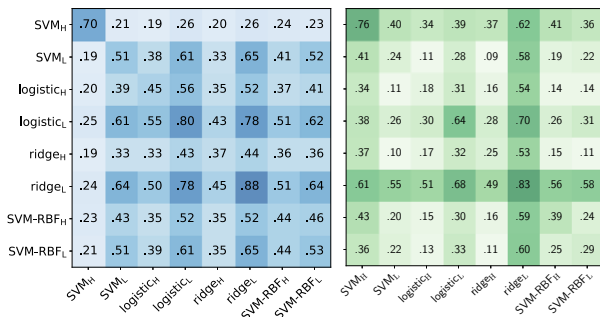Figure 21: Black-box (transfer) poisoning attacks on LFW. See the caption of Fig. 7 for further details.

Figure 22: Gradient alignment and perturbation correlation (at 20% poisoning) for poisoning attacks on LFW. See the caption of Fig. 8 for further details.

Figure 23: Adversarial examples crafted against linear and RBF SVMs. Larger perturbations are required to have significant impact on low-complexity classifiers (L), while minimal changes are very effective on high-complexity SVMs (H).

target models, and the variance of the attacker optimization objective. The lesson to system designers is to evaluate their classifiers against these criteria and select lower-complexity, stronger regularized models that tend to provide higher robustness to both evasion and poisoning. Interesting avenues for future work include extending our analysis to multi-class classification settings, and considering a range of gray-box models in which attackers might have additional knowledge of the machine learning system (as in [41]). Application-dependent scenarios such as cyber security might provide additional constraints on threat models and attack scenarios and could impact transferability in interesting ways.

Acknowledgements

References

[1] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck. Drebin: Efficient and explainable detection of android malware in your pocket. In *21st NDSS*. The Internet Society, 2014.

[2] A. Athalye, N. Carlini, and D. A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, vol. 80 of *JMLR W&CP*, pp. 274–283. JMLR.org, 2018.

[3] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In H. Blockeel et al., editors, ECML PKDD, Part III, vol. 8190 of *LNCS*, pp. 387–402. Springer Berlin Heidelberg, 2013.

[4] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In J. Langford and J. Pineau, editors, *29th Int'l Conf. on Machine Learning*, pp. 1807–1814. Omnipress, 2012.

[5] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Stats)*. Springer, 2007.

[7] N. Carlini and D. A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In B. M. Thuraisingham et al., editors, *10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pp. 3–14, New York, NY, USA, 2017. ACM.

[8] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symp. on Sec. and Privacy*, pp. 39–57. IEEE Computer Society, 2017.

[9] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *ArXiv e-prints*, abs/1712.05526, 2017.

[10] H. Dang, Y. Huang, and E.-C. Chang. Evading classifiers by morphing in the dark. In *24th ACM SIGSAC Conf. on Computer and Comm. Sec.*, CCS, 2017.

[11] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Trans. Dependable and Secure Computing*, In press.

[12] A. Demontis, P. Russu, B. Biggio, G. Fumera, and F. Roli. On security and sparsity of linear classifiers for adversarial settings. In A. Robles-Kelly et al., editors, *Joint IAPR Int'l Workshop on Structural, Syntactic, and Statistical Patt. Rec.*, vol. 10029 of *LNCS*, pp. 322–332, Cham, 2016. Springer International Publishing.

[13] Y. Dong, F. Liao, T. Pang, X. Hu, and J. Zhu. Boosting adversarial examples with momentum. In *CVPR*, 2018.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[15] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. D. McDaniel. Adversarial examples for malware detection. In *ESORICS (2)*, vol. 10493 of *LNCS*, pp. 62–79. Springer, 2017.

[16] T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *NIPS Workshop on Machine Learning and Computer Security*, vol. abs/1708.06733, 2017.

[17] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In J. Dy and A. Krause, editors, *35th ICML*, vol. 80, pp. 2137–2146. PMLR, 2018.

[18] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE Symp. S&P*, pp. 931–947. IEEE CS, 2018.

[19] A. Kantchelian, J. D. Tygar, and A. D. Joseph. Evasion and hardening of tree ensemble classifiers. In *33rd ICML*, vol. 48 of *JMLR W&CP*, pp. 2387–2396. JMLR.org, 2016.

[20] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proc. of the 34th Int'l Conf. on Machine Learning*, ICML, 2017.

[21] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[22] C. Lyu, K. Huang, and H.-N. Liang. A unified gradient regularization family for adversarial examples. In *IEEE Int'l Conf. on Data Mining (ICDM)*, vol. 00, pp. 301–309, Los Alamitos, CA, USA, 2015. IEEE CS.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[24] S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *29th AAAI Conf. Artificial Intelligence (AAAI '15)*, 2015.

[25] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, and F. Roli. Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid. In *ICCVW ViPAR*, pp. 751–759. IEEE, 2017.

[26] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *CVPR*, 2017.

[27] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In B. M. Thuraisingham et al., editors, *10th ACM Workshop on AI and Sec.*, AISec '17, pp. 27–38, New York, NY, USA, 2017. ACM.

[28] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *LEET '08*, pp. 1–9, Berkeley, CA, USA, 2008. USENIX Association.

[29] A. Newell, R. Potharaju, L. Xiang, and C. Nita-Rotaru. On the practicality of integrity attacks on document-level sentiment analysis. In *AISec*, 2014.

[30] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *RAID*, pp. 81–105. Springer, 2006.

[31] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv:1605.07277, 2016.

[32] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *ASIA CCS '17*, pp. 506–519, New York, NY, USA, 2017. ACM.

[33] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv e-prints*, abs/1605.07277, 2016.

[34] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif. Misleading worm signature generators using deliberate noise injection. In *IEEE Symp. Sec. & Privacy*, 2006.

[35] A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *AAAI*. AAAI Press, 2018.

[36] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *9th ACM SIGCOMM Internet Measurement Conf.*, IMC '09, pp. 1–14, NY, USA, 2009. ACM.

[37] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli. Secure kernel machines against evasion attacks. In *9th ACM Workshop on AI and Sec.*, AISec '16, pp. 59–69, New York, NY, USA, 2016. ACM.

[38] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM SIGSAC Conf. on Comp. and Comm. Sec.*, pp. 1528–1540. ACM, 2016.

[39] C. J. Simon-Gabriel, Y. Ollivier, B. Schölkopf, L. Bottou, and D. Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. *ArXiv*, 2018.

[40] J. Sokolić, R. Giryes, G. Sapiro, and M. R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Trans. on Signal Proc.*, 65(16):4265–4280, 2017.

[41] O. Suciu, R. Marginean, Y. Kaya, H. D. III, and T. Dumitras. When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks. In *27th USENIX Sec.*, pp. 1299–1316, 2018. USENIX Assoc.

[42] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[43] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. The space of transferable adversarial examples. *ArXiv e-prints*, 2017.

[44] D. Varga, A. Csiszárik, and Z. Zombori. Gradient Regularization Improves Accuracy of Discriminative Models. *ArXiv e-prints ArXiv:1712.09936*, 2017.

[45] N. Šrndic and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *IEEE Symp. Sec. and Privacy*, SP '14, pp. 197–211, 2014. IEEE CS.

[46] B. Wang and N. Z. Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 36–52. IEEE, 2018.

[47] L. Wu, Z. Zhu, C. Tai, and W. E. Enhancing the transferability of adversarial examples with noise reduced gradient. *ArXiv e-prints*, 2018.

[48] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In F. Bach and D. Blei, editors, *JMLR W&CP - 32nd ICML*, vol. 37, pp. 1689–1698, 2015.

[49] W. Xu, Y. Qi, and D. Evans. Automatically evading classifiers a case study on PDF malware classifiers. In *NDSS*. Internet Society, 2016.

[50] F. Zhang, P. Chan, B. Biggio, D. Yeung, and F. Roli. Adversarial feature selection against evasion attacks. *IEEE Trans. on Cybernetics*, 46(3):766–777, 2016.