Chapter 1
Adversarial Machine Learning

Carlos Javier Hernández-Castro*[1], Zhuoran Liu*[2], Alex Serban*[2], Ilias Tsingenopoulos*[3], and Wouter Joosen[3]

Abstract Recent innovations in machine learning enjoy a remarkable rate of adoption across a broad spectrum of applications, including cyber-security. While previous chapters study the application of machine learning solutions to cyber-security, in this chapter we present adversarial machine learning: a field of study concerned with the security of machine learning algorithms when faced with attackers. Likewise, adversarial machine learning enjoys remarkable interest from the community, with a large body of works that either propose attacks against machine learning algorithms, or defenses against adversarial attacks. In particular, adversarial attacks have been mounted in almost all applications of machine learning. Here, we aim to systematize adversarial machine learning, with a pragmatic focus on common computer security applications. Without assuming a strong background in machine learning, we also introduce the basic building blocks and fundamental properties of adversarial machine learning. This study is therefore accessible both to a security audience without in-depth knowledge of machine learning and to a machine learning audience.

1.1 Introduction

While previous chapters discuss the applications of machine learning (ML) to security, it is also compelling to reflect on the vulnerabilities of ML algorithms. In this chapter, we provide a broad introduction to adversarial ML – a field of study concerned with failure modes of ML algorithms when faced with adversarial attackers. In particular, we focus on adversarial ML for security-oriented practical cases, where the failure of ML algorithms to satisfy the intended functionality represents a security vulnerability. In such cases a well-thought and practical threat model is

1: Complutense University, Spain, · 2: Radboud University, The Netherlands · 3: imec-Distrinet, KU Leuven, Belgium

* Equal contributions, authors ordered alphabetically.

essential. This contrasts the recent increase in popularity of adversarial ML for deep learning (DL), where the focus is on the algorithmic robustness of computer vision and natural language processing. While adversarial ML may pose security risks for some applications of computer vision, e.g., autonomous driving, this does not entail that computer vision is a powerful attack vector for malicious attackers to exploit.

These contrasting views emerge due to two causes. First, the lack of thorough threat models leads to the false impression that some vulnerabilities of ML algorithms, such as the sensitivity to small perturbations in the input space, have immediate security consequences [41]. Second, the economics of creating defenses against adversarial attacks are often disregarded. The importance of protecting against adversarial threats is evident in ML-based security applications, for instance in malware detection. At the same time, solving vulnerabilities for some seemingly non-critical applications, like object detection in cloud image storage, can also prove crucial under a convincing threat model.

In this chapter, we take a practical approach and present the most important developments in applications of ML. Our presentation is focused on understanding the basic building blocks of adversarial ML and on showcasing different use cases where adversarial ML raises security concerns. The rest of this chapter is organized as follows. We begin with background information on ML and introduce adversarial ML from a historical perspective (Section 1.2). Next we discuss the elements for threat modeling (Section 1.3) and use them to present white-box attacks (Section 1.4) and black-box attacks (Section 1.5). Afterwards, we introduce defenses and a protocol for evaluating adversarial defenses (Section 1.6). We continue with a presentation of adversarial ML in different domains (Section 1.7) and concluding remarks (Section 1.8).

1.2 Background

Whenever not mentioned otherwise, we use the task of supervised classification to support our presentation. Mitchell describes the general task of learning from experience w.r.t to a task and a performance measure if the performance on the task increases with experience [86]. Given a set of training examples defined on an input space $\mathcal{X}$ and corresponding labels defined on an output space $\mathcal{Y}$, sampled from a fixed probability distribution over the space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, an algorithm seeks to uncover a mapping $f : \mathcal{X} \to \mathcal{Y}$ which maximizes the performance measure (i.e., minimizes the error rate). The error rate is measured by a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which is positive whenever there is an error and zero otherwise. During learning, an algorithm selects a function f^* from the space $\mathcal{F}$ (also called the hypothesis space) such that the expected loss $r(f) = \mathbb{E}_{(\boldsymbol{x},y\sim\mathcal{D})}[l(f(\boldsymbol{x},\boldsymbol{\theta}),y)]$ is minimal: $f^* = \arg\min_{f\in\mathcal{F}} r(f)$, where $\boldsymbol{\theta}$ are the parameters of f. However, at training time the full distribution $\mathcal{P}$ of the data is unknown. Instead, only a set of samples drawn from $\mathcal{P}$, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is available for training. Therefore, the empirical loss is used to approximate the expected loss:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \quad \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[l(f(\boldsymbol{x},\boldsymbol{\theta}),y)], \tag{1.1}$$

where $f(\boldsymbol{x},\boldsymbol{\theta})$ can be any mapping from $\mathcal{X}$ to $\mathcal{Y}$ such as a neural network or a linear function. The choice of $\mathcal{F}$ represents a bias induced by the algorithm developers and can lead to over-fitting when $\mathcal{F}$ is too expressive or under-fitting when $\mathcal{F}$ is not expressive enough.

Fundamental assumptions of ML.

Two fundamental assumptions underpin the ML framework introduced above. Firstly, the data used both at training and test time is sampled identically from the same probability distribution $\mathcal{P}$, and the samples are mutually independent (also called the i.i.d assumption). Secondly, since the input space $\mathcal{X}$ can be very large (in most applications of ML the input is high-dimensional), ML algorithms assume that most of this space consists of invalid inputs and that interesting inputs (together with all their variations) lie in a collection of lower dimensional manifolds.

Both assumptions have critical consequences for the security of ML algorithms because once any of the assumptions is broken, the algorithms become vulnerable. For example, if a malicious attacker uses at test time an input sampled from another distribution than $\mathcal{P}$, the algorithm is expected to fail because it was not designed to work with data outside $\mathcal{P}$. Similarly, if the test inputs lie off the training data manifold, the algorithms are unlikely to perform well.

1.2.1 Related Work

Research in adversarial ML began around 2004, when Dalvi et al. [28], followed by Lowd and Meek [82] targeted linear models for spam detection, and managed to construct viable adversarial attacks. Later on, Barreno et al. [9] introduced a taxonomy for attacks and defenses in adversarial settings, which was refined in [8]. This early taxonomy defined common ML threat models, and was comprehensive enough to encompass modern attacks. In particular, Barreno et al. made a clear distinction between attacks at *training time* – which assume attackers can manipulate the training data – and attacks at *test time* – which assume attackers can only query already trained models.

This classification shaped future work, where a large body of publications discussed adversarial attacks against ML models at both training [14, 97] and test time [43, 82] while defense mechanisms were devised against such attacks [18, 69]. In parallel to developing attacks and defenses, several works proposed methods to evaluate the security of ML models against adversarial attacks [8, 13].

With the burgeoning field of DL, a renewed wave of interest in adversarial ML is observed. The seminal work that sparked this interest was a publication for test time attacks in DL by Szegedy et al. [114]. The discovery that deep neural networks

(DNNs) are susceptible to small perturbations in the input space – also called adversarial examples – triggered an impressive body of publications that analyze the phenomenon from different angles, with a focus on computer vision algorithms. Several overviews and taxonomies on this topic can be found in the works of [2, 76, 103, 112]. While many of these publications claim security consequences of adversarial examples and hypothesize that deployment of ML algorithms may be delayed until these vulnerabilities are solved, other publications show these claims are spurious as they lack explicit security threats [41]. Biggio and Roli [15] draw a parallel between the evolution of adversarial ML and the rise of DL. They observe that publications which focus on DL seem unaware of early research in adversarial ML and miss important details for security, especially regarding threat modeling.

More recently, the interest on adversarial ML has overflowed to other types of attacks, such as train time attacks [56], model inversion [66] or model extraction [117]. These directions are meant to complement the research in adversarial examples and explore the full spectrum of adversarial ML.

Besides the works presented above, several publications present broad overviews of adversarial ML. The book by Joseph et al. [68] offers a comprehensive introduction to the field, tackling multiple topics of interest. Similarly, Vorobeychik and Kantarcioglu [119] published a book to introduce adversarial ML, while literature reviews [23, 103] with slightly different focuses have been published.

1.3 Threat modeling and taxonomy of adversarial machine learning

Before describing adversarial attacks and defenses, we introduce the threat modeling elements which can be used to characterize them. For any discussion about security it is mandatory to have a description of a threat model, also called an attack model. The attack model describes what an attacker aims to achieve (the attacker's goal) together with a description of the attacker's knowledge and capability. These form a description of the resources needed to mount the attack and achieve the goals.

Similarly, when designing a defense, the defenders choose a model in which the defense works. This model involves the critical assumptions a defense makes, the processing step where the defense is applied, the resources needed to develop and deploy it and the resources needed to break or bypass it. The latter can be either an attempt to break the defense, or to disprove the claims necessary to develop it and although important, it is often overlooked [21, 116].

1.3.1 Attacks

We characterize adversarial attacks using three dimensions corresponding both to early taxonomies [8] and with recent updates [15,68,92]: (1) the attacker's influence,

(2) the attacker's goal or specificity and (3) the attacker's knowledge of the model under attack. A fourth dimension concerns the security or privacy violation that can map to the confidentiality, integrity, availability (CIA) triad [68]. However, in the chapter we obviate its use as it is not a prevailing approach in the related work.

Based on the assets influenced during the attack, attacks are classified in:

- *Train time attacks* – also called causative or training *data poisoning* attacks [15, 68]; these attacks manipulate the *training data* and influence the training process.
- *Test time attacks* – also called inference or exploratory attacks [68]; these attacks manipulate only the data used at *test time* and do not alter the training process.

Formally, assuming the available data set $\mathcal{D}$ for learning is divided in a training $\mathcal{D}_{train}$ and a testing $\mathcal{D}_{test}$ data set, the train time attacks assume knowledge of $\mathcal{D}_{train}$ and the capacity to alter or corrupt inputs or labels in order to bias the outcome of $f(\boldsymbol{\theta})$. In classification, we can picture a scenario in which a fraction of $\mathcal{D}_{train}$ is corrupted such that any examples from class c_1 will be classified as c_2. Similarly, test time attacks presume access only to $\mathcal{D}_{test}$. This type of attacks can only be mounted after training and assumes that test examples can be altered in order to achieve the attacker's goals.

Further on, based on the attacker's goals, attacks are classified in:

- *Targeted attacks* – focus on *one input*.
- *Untargeted attacks* – also called indiscriminate attacks, focus on a *set of inputs*.

In the supervised classification scenario introduced in Section 1.2, targeted attacks can also be interpreted as inducing a misclassification of an example towards a *specific class*. Conversely, untargeted attacks can be interpreted as inducing a *random* misclassification. Formally, given a sample $\boldsymbol{x}$, targeted attacks aim to find a perturbation η, $\boldsymbol{x'} = \boldsymbol{x} + \eta$ such that $f(\boldsymbol{x'}) = \hat{y}$ where $\hat{y} \neq y$ is a label selected by the attacker and y is the true label. Similarly, in untargeted attacks $\hat{y}$ is any label not equal with y: $f(\boldsymbol{x'}) = \hat{y}, \forall\, \hat{y} \neq y$.

Based on the the attacker's knowledge, attacks are classified in:

- *White-box attacks* – attackers have complete knowledge of the model under attack, the training and the test data and can completely replicate the model.
- *Black-box attacks* – attackers do not have any knowledge of the model under attack, but can query it as an oracle.

Formally, white-box attacks presume the attacker has full access to $\mathcal{D}$ and $f(\boldsymbol{\theta})$, while black box attacks only have access to $\boldsymbol{x'}$ and y, with $y = f(\boldsymbol{x'}, \boldsymbol{\theta})$. An illustration of white-box and black-box attacks is provided in Figure 1.1.

Biggio and Roli [15] also consider a gray box scenario in which the attacker has partial knowledge about the model under attack. However, this attack type is an instance of the black-box attack where the attacker has some restrictions [21].

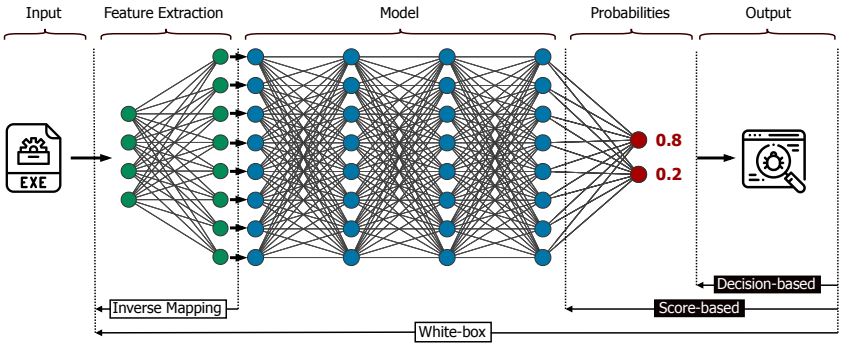

Fig. 1.1 The components of an abstract malware detection model, that form a pipeline from sample input to output. Below the model are the various levels of accessibility an adversary can have.

1.3.2 Defenses

Defenders can *react* and improve the system in response to new attacks or act in advance and *proactively* design the ML algorithms with security in mind. This corresponds to two types of defenses, namely *reactive* and *proactive* [14, 76].

Moreover, defenses can be classified based on the step of the processing pipeline in which they are deployed, in [103]:

- *Guards* – guards act as a preprocessing step before malicious inputs reach a ML models. These methods try to detect adversarial inputs or apply input transformations to reduce their impact. Formally, guards can be defined as a function $g(\phi)$ that processes the input before the ML model: $f(g(x,\phi),\boldsymbol{\theta})$. In some cases, $g(\phi)$ can choose to discard the input altogether.
- *Defenses by design* – also called model hardening. These defenses consist of new models, new regularization techniques or new training procedures that alleviate the impact of adversarial inputs.
- *Certified defenses* – defenses which use formal methods to certify that models cannot be attacked within some bounds. Certified defenses can only provide certificates for the training data set (and not for the testing data set).

1.4 White-box attacks

Although the scenario in which attackers have complete knowledge of the model under attack is not probable in security sensitive contexts, understanding white-box attacks can help in many ways. For example, white-box attacks enhance the algorithms' designers power to evaluate and understand the models developed [19]. Moreover, white-box attacks can be used to create powerful defenses [84]. In this

section we present the basics of white-box attacks, using the assets influenced during the attack, introduced in Section 1.3.

1.4.1 Train time white-box attacks

Train time attacks inject adversarial samples in the training data in order to increase the number of misclassified examples. Based on the attack specificity introduced above, train time attacks can be untargeted (also called error-generic [15]) in which an attacker cause a denial of service by raising the number of misclassified examples in all classes, and targeted (also called error-specific) in which an attacker causes specific misclassifications, e.g., for one class. In general, train time attacks can be defined as:

$$\begin{aligned} &\arg\max \mathcal{D}_c \quad f'(\cdot, \boldsymbol{\theta}^*) \\ &s.t. \qquad\qquad\quad \boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \; l(\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{c}}, \boldsymbol{\theta}), \end{aligned} \tag{1.2}$$

where the outer maximization problem selects the corrupted data points $\mathcal{D}_c$ that will lead to the best outcome, based on an attacker's defined $f'(\cdot)$. The inner minimization problem retrains the ML model with these data points [67]. In all cases, attackers are interested in minimizing the number of samples $\mathcal{D}_c$ that have to be corrupted in order to achieve their goals.

Particularly interesting for train time attacks is the scenarios of online learning, where algorithms learn on the fly, as new data becomes available. A landmark example of train time attacks is the Tay Twitter bot launched by Microsoft, which quickly learned to post offensive Tweets, looking at other comments.

1.4.2 Test time white-box attacks

At test time, an attacker can change a sample in order to induce a desired behavior for the model under attack. In classification, the problem can be framed as finding a sample $\boldsymbol{x}'$ derived from a natural input $\boldsymbol{x}$ such that $f(\boldsymbol{x}', \boldsymbol{\theta}) = \hat{y}$, where $\hat{y}$ can be any label different than the original label y in the untargeted scenario or a specific label in the targeted scenario.

Finding $\boldsymbol{x}'$ can be framed as an optimization problem where we seek to discover the minimum change $\eta = d(\boldsymbol{x}' - \boldsymbol{x})$ s.t. $\hat{y} \neq y$:

$$\begin{aligned} &\min . \quad d(\boldsymbol{x}' - \boldsymbol{x}), \\ &\text{s.t.} \qquad f(\boldsymbol{x}', \boldsymbol{\theta}) = \hat{y}, \end{aligned} \tag{1.3}$$

where $d(\cdot)$ is a distance function defined on the (metric) input space $\mathcal{X}$. The choice of $d(\cdot)$ is context dependent and influences $\boldsymbol{x}'$. For example, in object recogni-

tion or image classification $d(\cdot)$ is commonly a p-norm, with $p \in \{1, 2, \infty\}$. Research on human visual perception-aware adversarial images also consider geometric transformation [31, 123], textural-aware p-norm [27, 83, 131] color transformation [10, 57, 73, 105, 132], and existing perceptual metrics [96, 121, 133]. Choosing different values for p influences the size of the perturbation η. In other contexts, such as malware detection, $d(\cdot)$ can be the number of bits that can be flipped s.t. the binary still compiles. Similarly, for spam detection $d(\cdot)$ can be the number of characters that have to be changed in a document in order to bypass a detector.

With full knowledge of the parameters $\boldsymbol{\theta}$, Eq. 1.3 can be solved with algorithms like L-BFGS [114] or genetic algorithms [4]. However, in contexts where it is needed to solve Eq. 1.3 fast, attackers cannot rely on classical solvers. For example, if we wish to strengthen the model under attack using adversarial samples in training, a fast method to generate adversarial examples is needed. Conversely, if one sample suffices for the attacker – think of malware detection, where one sample that passes a detector is sufficient for a successful attack – attackers can rely on classical solvers.

In order to overcome the computational disadvantage of classical solvers, researchers have developed fast methods to generate adversarial examples. The most popular attack, called FGSM [46] takes one step towards maximizing the loss function (assuming the loss is differentiable): $\eta = \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} l(f(\boldsymbol{x}, \boldsymbol{\theta}), y))$, which corresponds to a step towards the sign of the gradient. In order to strengthen this attack, it can be applied iteratively, and the outcome can be projected on the space defined by $d(\cdot)$ [84]. This iterative attack called projected gradient descent (PGD) is one of the most popular, that also stood the test of time [84].

Other types of attacks mounted at test time use modified samples in order to check if an input was used during training, i.e., model inversion [66] or for trying to reverse engineer the model under attack, i.e., model extraction [117]. In all cases the problem can be framed as an optimization problem or as solving a system of equations [117].

1.5 Black-box attacks

In comparison to white-box attacks, black-box attacks are more likely in security-critical applications. For example, a cloud ML service, an online malware analysis tool, a network intrusion detection system, or a CAPTCHA test are scenarios in which an attacker can only observe the decisions of a model.

Black-box attacks against ML models are delineated by two main assumptions: (1) queries to the model under attack can be submitted consistently and (2) the model responds with its decisions, without any further access to the specifics behind it. Services like Google Cloud Vision AI[2] or VirusTotal[3], which offer image classification and malware detection APIs respectively, have been shown to be susceptible to

[2] https://cloud.google.com/vision

[3] https://www.virustotal.com/gui/home/upload

black-box attacks [6, 64]. Even in such restricted environments, the mere decisions that a model makes are sufficient to mount adversarial attacks.

Based on the underlying method, the query requirements, and the underlying assumptions, we can classify black-box attacks in:

- *Score-based Attacks* – use gradient-free optimization methods to approximate the gradient of a black-box loss function and generate adversarial examples.
- *Transfer-based Attacks* – use a substitute model, trained by selectively querying the black-box model. Afterwards the substitute model is used to generate adversarial examples in a white-box setting.
- *Decision-based (also called Boundary) Attacks* – the attacks use only the final, hard-label decision of the model.

1.5.1 Score-based attacks

A black-box ML model means that attackers do not have access to the analytical expression that defines it. Therefore, score-based attacks use the class probabilities that a model outputs, as a way to estimate the gradient of its loss function. One of the first papers to propose gradient estimation as a way to attack ML models was by Chen et al. [25], where they approximate the gradient using the finite differences method $df/dx \approx (f_2 - f_1)/\Delta x$. However, for a complex model, e.g., Inception-v3 for object classification, approximating the gradient of only one sample $\boldsymbol{x}$ with the finite differences method requires more than 500,000 queries.

While interesting in nature, this approach has limited practical implications because it faces scalability issues. Subsequent publications [64, 65] expand on it, using bayesian optimization or natural evolution strategies to reduce the number of queries.

The class probabilities generated by the black-box model can also be used differently. Alzantot et al. [5] use this score as the degree of fitness of the adversarial perturbations. The candidate solutions are constructed around an initial input example $\boldsymbol{x}$ by applying independent and uniformly distributed random noise to every dimension of the input vector. They evaluate the fitness of a generation's members to select parents, then crossover and mutation are performed to form the next generation until a successful adversarial example is discovered.

Score-based attacks can have a high success rate because the gradient estimation is calculated and refined directly on the underlying black-box function (the model under attack), and they can function irrespective to the model family being attacked. At the same time they have a high query budget and they construct single-use, conditional on a specific sample perturbations.

1.5.2 Transfer-based attacks

Transfer-based attacks stem from the observation that adversarial examples crafted on one model can also transfer to other models, regardless on the algorithms used [114]. These attacks restrict or completely eschew the number of queries submitted to the model under attack. In order to mount transfer-based attacks, an attacker requires information about the data used for training, as their effectiveness relies considerably on the degree of access to the training set. Liu et al. [78] performed a comprehensive evaluation of transfer-based attacks, between and within model families. They showed that transferability is more prominent when three conditions hold: a) examples are transferred between models of the same family, e.g., between convolutional neural networks, b) when the training data are described by the same distribution, c) when the hyperparameters used to train the model are similar or comparable in magnitude. Conceptually it is evident why: the intention is for the substitute model to learn a decision boundary as close to the black-box model as possible.

Another approach in training substitute models for black-box attacks involves generative adversarial networks (GANs). To further increase the utility of queries, Xiao et al. [122] utilize generative models in order to generate perturbations conditional on input. Their pivotal idea is that instead of learning the ground-truth labels the goal of the GAN discriminator is to fit the black-box detector, by replacing the ground-truth labels with the black-box model output. Hu et al. [59] formulate a similar attack methodology against a malware classifier. Moreover, if the adversary's capabilities allow it, and in order for adversarial examples to be more robust and transferable, they can be created against an ensemble of diverse substitute models.

1.5.3 Decision-based attacks

Decision-based attacks rely solely on the final decision of the model, i.e., the class output of a classifier. These attacks can be considered a constrained version of score-based attacks, where the only informative signal is the final output.

The first to propose effective decision-based attacks were Brendel et al. [17], which introduced a novel approach to generate adversarial examples. Instead of starting from the original example, the authors initiate the attack from an example belonging to the target class (i.e. one with a potentially large perturbation) and minimize this perturbation by moving the example closer to ones in the original class. The strength of this approach is that an attacker has access to a continuous and more informative signal while minimizing this distance.

In order to discover better adversarial examples, i.e., with minimal perturbation, decision-based attacks have to look for perturbation directions that take a step away from the decision boundary. The overall query efficiency depends significantly on the proposed distribution of directions to explore. In Brunner et al. [20] the number of queries needed is reduced through biased sampling of this distribution, while Chen et al. [24] have further improved the efficiency by replacing this exploratory step with

an improved estimation of the gradient direction. Vulnerability to decision-based attacks is essential to evaluate for two key reasons: (1) conceptually they correspond to many realistic use-cases, where the model exposes an interface and is attacked in real time (rather than pre-calculating a perturbation on a substitute model), and (2) they can be more efficient compared to attacks with access to more information, and this efficiency is steadily improving. To better visualize the various threat models, an abstract malware detection model is depicted in Figure 1.1.

1.6 Defenses

Although a plethora of defenses against adversarial attacks have been proposed over time, the landscape of *effective* adversarial defenses remains scarce. Particularly because the correct evaluation of adversarial defenses proves difficult [21, 116]. For example, many defenses against test time attacks induce a phenomenon called "gradient masking" [7], in which the gradient used by attackers to generate adversarial examples is obscured. Such defenses provide a misleading sense of security, while in fact the models remain vulnerable to other types of attacks that do not use gradients.

Similarly, many defenses which propose to use distinct training objectives in order to achieve robustness (defenses by design, Section 1.3) induce a comparable phenomenon [116]. However, instead of hiding the gradient information these defenses are evaluated against classical attacks (e.g., PGD) that are not catered to accommodate the new loss functions. This evaluation can lead to illusive results, showing the models are robust when in fact they are not. Tramer et al. [116] showed recently that many defenses published at top tier conferences in ML suffer from this phenomenon.

Moreover, while appealing because they can give formal guarantees, certified defenses [63] can provide the guarantees only for the training data set. In case the training data is not representative for the data generation distribution, certified defenses will render the models vulnerable to data sampled from the same distribution, but distinct from the training data.

In light of this, we consider it futile to introduce state-of-the art defenses which are not proven to stand the test of time or cannot provide guarantees for test data. Instead, we only briefly discuss some defenses that proved effective over time and comment on a protocol of evaluating defenses in Section 1.6.1.

The most common defense against attacks at training time is to perform outlier detection or input sanitization before training [113]. In this case, the quality of the outlier detection mechanism employed will dictate the strength of the defense.

Conversely, the most common defense against attacks at test time, which is also considered one of the most successful, is adversarial training. This defense adds adversarial examples in the training data set and changes the training objective to: $\tilde{l}(\cdot) = \alpha l(f(\boldsymbol{x}, \boldsymbol{\theta}), y) + (1 - \alpha) l(f(\boldsymbol{x'}, \boldsymbol{\theta}), y)$, where α controls the influence of adversarial examples during training. When $\alpha = 0$, training is performed only with adversarial examples. If the method to generate adversarial examples manages to approximate well the space we want to provide robustness to (defined by $d(\cdot)$), ad-

versarial training can, in principle, provide robustness for this space. For example, pairing the PGD attack to approximate the space around an input with adversarial training, is known to be an effective defense. However, since PGD follows an iterative procedure, the training time is substantially increased. Moreover, although effective, adversarial training has not succeeded to fully protect a model in practice. New training procedures partially alleviate some of these issue by accumulating perturbations during training and reducing the number of iterations [104], but most adversarially trained models remain vulnerable to adversarial attacks.

The overall dynamics of defenses change in realistic black-box scenarios, where we consider the black-box model as a deployed interface that accepts queries and responds with decisions. This opens the opportunity for a complementary defense, beyond building a priori more robust models, through observing and adapting the system behavior as the attack unfolds. For example, Chen et al. [26] have proposed a stateful detection methodology for an image classification model. Their approach keeps track of the history of submitted queries in order to detect if a query is part of an attack or benign. Each query attributed to a uniquely identified actor is mapped to a lower dimensional space using an encoder. The encoder is pre-trained to map visually similar queries close to each other, even if adversaries have used transformations like rotation and translation to obfuscate their activity. The lower dimensional mapping contributes also to the efficiency of searching the history of queries per actor and to the scalability of the approach.

1.6.1 On the evaluation of adversarial defenses

As mentioned above, the evaluation of adversarial defenses is not trivial because the effects of a defense – e.g., if the defense hides some information or actually protects a model – are hard to determine. Therefore, there is no standard way to test adversarial defenses. The general recommendation is to test a defense rigorously, against a wide variety of attacks that can be adapted and catered to specific characteristics of the defense, i.e., to custom loss functions or regularization terms. Until now, the most detailed protocol for evaluating adversarial defenses is the work of Carlini et al. [21]. Moreover, the lessons learned from [7] and [116], which succeeded in breaking a variety of defenses can contribute to strengthen the evaluation protocols.

Similarly, it is recommended to design and think of tests for data out of the training distribution [40] (see Section 1.2) and adapt traditional and practical security techniques such as red teaming [19] in order to constantly evaluate ML models and develop a sense of skepticism about their robustness.

From a theoretical perspective, it is rather unlikely that completely robust ML models can be built in the near future, without trading other resources. For example, in some learning settings, developing robust models requires a trade-off between accuracy and robustness [100], a phenomenon which was also observed empirically in other settings [118]. A way to overcome such trade-offs is to collect more training data. However, the upper bounds on the data set sizes needed to develop robust models

are not realistic in practice [100]. We recommend following the standard procedures defined in [21,116] when designing or evaluating ML models in adversarial settings.

1.7 Domains of adversarial machine learning

Adversarial examples can be built for any applications of ML & DL. However, each application domain has a particular type of input being processed. Initial techniques for creating adversarial examples assumed a continuous domain – even though the input was quantized. Conversely, many practical applications have a discrete input space that has to comply with data formats (e.g., a PDF file, PE executable, etc.). More restrictions on the input data, such that some important characteristic of the input-related semantics are preserved impose new constraints on the methods used to mount adversarial attacks.

For example, in some domains (others than computer vision), the feature extraction is a separate step preceding the input to the model, and it is usually neither invertible nor differentiable (see Figure 1.1). As a consequence, inverse-mapping the necessary perturbations from the feature space, that is the representation of an example that a ML model accepts, to the problem space, that is the actual example as it occurs, is a non-trivial process. This process is aggravated further if we consider that despite any adversarial perturbation, the mapping has to guarantee that the semantics of the original unperturbed sample are preserved.

In this section we discuss practical domain where adversarial ML poses security risks, and scenarios in which the two steps mentioned above are separated.

1.7.1 Malware Detection

The rise of polymorphic malware and its spread meant that manual detection rules (or *signatures*[4]) could not keep up with the number of new malware [72]. Therefore, the detection of malware moved from hand-crafted heuristics to hand-crafted features and ML [102]. This was the case for both static analysis, in which the binary code is examined, and dynamic analysis, in which the code behavior is examined - typically by looking at which OS or API functions it calls, its network behavior, and additional information from the anti-virus clouds.

Malware detection using ML or DL has some particular requirements:

- Interpretability – when a false positive or a false negative occurs, it is desirable to be able to understand whether there is a problem in the model, or the data, making it easier to correct its behavior.

[4] These signatures typically consisted on code fragments, file properties, hashes of the file or fragments, and combinations of these.

- Low FPR – false positives should be extremely low in order to make the application usable. For this, some companies use a model ensemble that allows them to fix false positives without a whole model retraining [72].
- Dataset size and diversity: when using DL, large data-sets are required. These should include representative malware and benign files, produced with different compilers, libraries, etc.
- Adaptability – ML algorithms typically assume a fixed distribution, or one that changes slightly over time. In the case of malware, algorithms are up against active adversaries that evolve and change their malware to avoid detection. At the same time, companies produce new types and versions of executables that are benign.

Prior to DL, the feature set was always handcrafted, which requires domain expertise. In a more recent example of hand-crafting features, [62] created 114 higher concepts from API call sequences and combined them with their inputs (as tri-grams) to build a malware detector and a malware family classifier. By combining both outputs, they improved the performance of both classifiers. Also for the identification of the malware family other authors proposed a DL solution: using dynamic analysis, they used the sequence of API calls (from a group of 60 selected) of a malware and analyzed them using convolutional and recurrent neural networks [71]. These solutions still relied on hand-crafted features though.

One of the first proposals that did not used hand-crafted features is MalConv, a convolutional neural network proposed by [94] that analyzes executable files in PE format up to several million bytes. It does not rely on any kind of feature engineering. Even so, and just using static analysis, is able to reach state-of-the-art performance or surpass it (depending on the metric).

Kolosnjaji et al. [70] proposes an attack against MalConv based on appending bytes, calculated using gradient descend (similar to the FGSM attack described in Section 1.4), with a success rate of 60%. In order to protect against these attacks, Al-Dujail et al. [3] propose an adversarial training framework samples generated with this method are included in the training data set.

Hu and Tan [58] also focus on the PE format, but on classifiers using RNNs applied to sequences of API calls. The authors train an RNN that approximates the detector, and then train another RNN to generate successful adversarial examples, with non-detection rates over 96%.

A different way to create adversarial malware is by training a Reinforcement learning (RL) agent [6]. This is a fully black-box attack, where only the final decision of the classifier is observed. The authors train the agent by doing functionality-preserving operations directly on the executable, a non-trivial approach to execute and often overlooked in adversarial machine learning research. The agent learns which sequences of actions are likely to result in evading the classifier, by selecting the optimal one from a closed set at each step. Actions that the authors considered was adding a function to the import address table that is never used, manipulating existing section names, creating new (unused) sections, creating a new entry point, manipulating debug info, and so forth.

Note that these works are academic and typically do not fully resemble the more complex DL architectures used in production environments. For example, [72] proposes a model based on similarity hashing, and then fine-grained analysis of each hash bucket depending on its classification difficulty using either an static label or a ML model. They also use an exemplar network [29] to classify targeted attacks, of which there are one or a few examples.

Grosse et al. [47] present one of the earlier works on the creation of adversarial examples for malware detection on Android applications. Using the DREBIN data set, the authors create an state-of-the-art classifier, and later use it to craft adversarial examples. As the input is discrete (a machine code program, or binary vector), so are changes to the input. In order not to alter the malware functionality, changes are restricted to certain instruction additions and changes in the *AndroidManifest.xml* file. Even with these restrictions, the authors achieve a miss-classification rate of over 60% with as little as an average of 14 binary changes to the input. Liu et al. [77] solve the same task using genetic algorithms.

These are not the only binary formats attacked. Other binary formats attacked with adversarial examples are PDF files [12], in order to inject undetected malware through adversarial examples. They also did it by appending new features to the PDF. Their work was followed by Xu et al. [126], who use genetic algorithms to fool two PDF malware detectors (PDFrate and Hidost).

1.7.2 Authentication

Biometric authentication refers to the process of ensuring that individuals are who they claim to be, by matching their physical or behavioral biometrics to an existing template. As a paradigm it is becoming widespread in various domains like law enforcement, border control, civil identification, but also physical and logical access.

On account of their performance, DNNs have been adopted in face recognition systems. Sharif et al. [107] were the first to attack such a system with a physically realizable adversarial attack, by constraining the adversarial perturbation only within a fixed frame around the eyes. Thus the adversary has the ability to 3D print the perturbation as wearable glasses and subsequently wear them in order to bypass detectors. In an attempt to make the attack general and to take into account potential environment discrepancies like pose or lighting, they follow a universal adversarial approach by computing perturbations on a range of the adversary's images. The attack is highly successful (over 80%) against common face recognition systems.

In [11], Biggio et al. comprehensively explore the attack surface of biometric authentication systems and showcase the vulnerabilities a self-updating system introduces. The physically realizable attacks can potentially exploit this vulnerability, called template poisoning. For a user to get authenticated, the extracted representation of their biometrics is compared to a template database. This comparison results in a confidence score that, if larger than a threshold, authenticates the user and the sample is enrolled in the template database. [81] show that an adversaries can

take advantage of this self-updating system and use adversarial glasses not only to authenticate, but through successive attempts can also move the template closer to them so that in the future they can authenticate without any accessories.

This property holds in every other context where systems learn and update themselves in a continuous and online manner. Attacks in these contexts can carry a dual, more insidious nature: able to evade and poison a system at the same time.

1.7.2.1 CAPTCHAs

Over twenty years ago CAPTCHAs, or Completely Automated Public Turing test to tell Computers and Humans Apart, were conceived as tests to detect humans from robots online [87]. The principle by which CAPTCHAs function is by presenting problems or puzzles that are difficult for AI but easy for humans to solve.

Unfortunately, CAPTCHAs have been unsuccessful in delivering robust security, either due to shortcomings in design or due to resourceful attacks [35–39, 48–54, 88, 127, 135, 136]. The most pertinent design criteria that such a test should embody are:

- Easy to construct and to grade for a machine.
- Easy for a human to solve.
- Difficult for an algorithm to guess the correct solution, even at very low success rates ($< 1\%$).
- Public, so that their security should not be based on the obscurity of the algorithms used.
- Time-proof, or resilient to the advances in ML & DL.

Text-based CAPTCHAs in particular had enjoyed widespread adoption and a long evolution, but were ultimately defeated by a combination of apt feature extraction and ML [39]. Some implementations, like reCAPTCHA, started by using their own set of hard image-recognition problems as tests, only to later realize that their own DNNs were able to solve them [45, 108]. reCAPTCHA also offered speech recognition tasks, that were either too simple for algorithms [98, 99, 115], or too difficult for humans [16].

The interest in using image recognition as a CAPTCHA has steadily increased while algorithms have maintained the ability to solve them, both prior to the rise of DL [30, 44], and after [110, 134]. There have been several proposals to make CAPTCHAs more resilient with little success so far. Using trap images has been shown to be prone to statistical analysis [55], while more recent proposals that create tests by using adversarial examples [91] and the transferability property show more promise. Lately, the area of bot identification has moved towards the security by obscurity paradigm in what some companies call *behavioral CAPTCHAs*, thus obviating the public part of the acronym. These detection mechanisms are based on user and client (browser) fingerprinting, as well as *possibly* more evolved algorithms that observe user behavior in order to correctly classify unknown clients.

While behavioral CAPTCHAs have not worked particularly well so far [109, 111, 120], they are still largely ongoing by virtue of being adaptable to attackers. Image-based CAPTCHAs are still used when the behavioral algorithm is unable to predict a new client with certainty.

1.7.3 Computer vision

Adversarial examples for computer vision algorithms have dominated the scene, with multiple applications ranging from object detection, object classification, image segmentation or depth estimation [125]. Nonetheless, the range of application where computer vision algorithms are used in security (and safety) critical systems is smaller [41].

A line of research concerned with autonomous driving makes extensive use of DL-based computer vision algorithms, which are highly susceptible to adversarial attacks. For instance, adversarial examples can be encountered in traffic sign recognition [33]. However, as safety and security are paramount concerns in autonomous vehicles, it is unlikely that the information from computer vision algorithms will not be double checked with other information sources, such as high definition maps. As far as we are concerned, traffic signs are designed for humans, which cannot maintain accurate maps of all the traffic signs. Nevertheless, computers do not suffer from this weakness, and mobile applications for driving assistance already integrate traffic signs. Therefore, although it is theoretically possible to endanger autonomous vehicles with adversarial attacks, in practice this attack vector may not lead to effectively realized attacks.

A related line of work is concerned with object detection and their removal from videos [32]. If computer vision algorithms are used for object tracking, in surveillance or other applications, security risks can emerge. However, in critical applications other sensors (e.g., infra-red) are also used to detect objects, as they are considered more reliable. Albeit appealing in theory, these applications are as well less susceptible to effective adversarial attacks.

Nonetheless, research in adversarial computer vision has contributed a plethora of compelling results and it is currently leading the field of adversarial ML, as well as being the benchmark environment and context where almost the entirety of fundamentally new attacks and defenses are evaluated on.

1.7.4 Speech recognition

Automatic Speech recognition (ASR) enables machines to transform human speech information into text, in order to improve human-machine interactions. There have been a large number of voice controllable systems (VCS) that are based on ASR techniques, for instance, voice-based search, personal assistance of smartphones,

home interaction systems, and in-vehicle infotainment systems [129]. Recently, DL-based ASR has achieved unprecedented developments, but adversarial ML also make ASR one of the most vulnerable applications.

Authentication (see also Section 1.7.2) is one security critical application of ASR, where people's voice is used as the biometric to identify individuals [1]. Note that compared with other biometrics, speech information is easily accessible. From the point of system availability [8], adversarial examples can be used to increase the false negative rate of the ASR system in order to make it unreliable. From the point of system integrity [8], adversasrial examples can be generated to resemble any voice.

VCS is another important application that is susceptible to adversarial examples, and it is widely deployed in commercial services, e.g., Siri, Google Assistant, Alexa, etc. By injecting voice commands that are inaudibility to human listeners, adversaries can manipulate the VCS-supported devices to achieve some malicious goals. For example, DolphinAttack [130] can manipulate the navigation system in an Audi automobile by modulating voice commands on ultrasonic carriers, which may influence the driver and cause severe results. And SurfingAttack [128] leverages ultrasonic guided wave to mount non-line-in-sight attack. Adversarial examples in DL-based ASR was comprehensively explored by Carlini et al. [22], where they analyzed hidden voice commands attacks that are unintelligible to human listeners and proposed detection and mitigation strategies against such attacks.

One characteristic of audio adversarial examples is "over-the-air", which indicates whether the adversarial voice is still effective after playing the adversarial audio on a speaker and record them with a microphone. Another characteristic of adversarial examples in ASR is inaudibility, which indicates whether human listeners can precept the adversarial voice. Inaudibility is equivalent to *imperceptibility* of adversarial images [114]. Instead of l_∞ norm in images, psychoacoustics were leveraged to achieve the goal of inaudibility [93, 101].

1.7.5 Reinforcement learning

Besides the supervised classification example introduced with Section 1.2, adversarial examples also manifest in RL. The objective in RL is for an agent to learn an optimal behavior or decision-making in an environment, by optimizing a reward function. RL solutions have been adopted in a wide range of domains, including many in cybersecurity such as cyber-physical systems [34], network attacks [85], smart grid security [89] and mobile edge caching [124].

Moreover, RL relates to adversarial machine learning in two ways:

1. RL can be used to create adversarial attacks to other ML models, that in turn, can be RL agents.
2. RL can be attacked using adversarial ML.

A RL approach for adversarial attacks has already been discussed in the malware subsection 1.7.1.

Parameterizing RL agents with DNNs introduces the same vulnerabilities that DNNs have, as it is possible to craft adversarial examples that fool the estimation of the state or the output of policy network. Huang et al. [60] have demonstrated that black-box attacks against RL agents are possible. For example, adversarial inputs on RL agents transfer to other agents as long as the goal remains the same. The authors tested three well-known RL algorithms – namely DQN, TRPO and A3C – using the FGSM attack (Section 1.4). In case they do not know which algorithm they are attacking, they calculate a proxy using the aforementioned RL algorithms. They discover that some adversarial input strategies are more transferable than others: for instance, when the adversary is constrained to a single pixel perturbation in the input image, the attack is particularly transferable.

Note that the input does not need to be adversarial constantly in order to fool an agent. Lin et al. [75] show that it is possible to time the adversarial input in order to diminish the reward of the agent. In particular, they attack both DQN and A3C in a white-box setting using two different approaches: the strategically-timed attack, where they choose the best moment to push the agent to select the least likely action (minimum reward), and the enchanting attack, where with a sequence of adversarial inputs they lead the agent to a pre-defined target state.

Being able to modify an agent's observation during training time is a strong assumption. To this end, Gleave et al. [42] introduce another adversarial agent in the environment instead of directly manipulating the input. The second agent learns how to fool the first one by discovering an adversarial policy, resulting from a zero-sum game. They evaluate the approach in a number of two-player games in robotics environments that include simulated physics and visual interaction (MuJoCo). The adversarial agent is able to learn policies that force the other agent to fail at its task.

In order to have a realistic evaluation of how robust are RL policies in adversarial settings, the policy of every agent must be determined through adversarial training, in the presence of adversarial agents. The authors demonstrated than an attacker can learn to produce close to natural observations, that are adversarial.

1.7.6 Other domains

We aimed to cover adversarial examples in the most prominent security applications. As ML methodologies have proliferated widely, it is not possible to be exhaustive, while also focusing on the fundamentals of adversarial ML. One such domain that we do not tackle is intrusion detection systems that are always a target of evasion attacks. Even though the construction of adversarial examples in the feature space proved to be feasible, and ML solutions are gradually being adopted in network intrusion detection systems, the effectiveness and practicality of adversarial examples in the context of live network environments remains an open question. For more information we refer to the Open-World Network Intrusion Detection chapter.

As a rule of thumb and irrespective of the domain, if a system is based on or uses ML, it is vulnerable to adversaries and adversarial examples. Therefore, it is

imperative, as a user, engineer or system designer, to be aware and well-informed about the vulnerabilities that ML models can exhibit. Interestingly, in contrast to all malicious adversarial attacks we have discussed, note that adversarial examples are not naturally-born nefarious, and it can be beneficial, like being used for privacy protections against unauthorized ML [61, 74, 79, 80, 90, 95, 106]. We refer readers interested in privacy protection to the Privacy Enhancement chapter.

1.8 Conclusions

We provided a brief introduction to adversarial machine learning (ML) – a field of study concerned with failure modes of ML algorithms when faced with adversarial attackers. We focused on presenting the range of attacks grouped by threat models and explaining the basic formulation for the attacks. We hope this information can help practitioners to understand how to construct and protect against adversarial inputs for their models. Furthermore, we introduced a considerable range use cases in security critical applications, where adversarial ML is mature and practiced.

We acknowledge that given the space constraints, this presentation can not be exhaustive. Whenever possible, we give pointers to literature offering more comprehensive overviews. We note that adversarial ML is an active research field, and that vulnerabilities to adversaries are inherent to the way current ML models are built. Up to this moment, no ML model is robust against adversaries and no defense is known to protect against adversarial inputs. While the arms race between attacks and defenses accelerates, proofs of robust behavior are ever more challenging and ultimately ephemeral. While this opens many research avenues, it also raises concerns about the deployment of ML algorithms in security critical contexts without prior and extensive consideration.

Acknowledgements

This research is partially funded by the Research Fund KU Leuven, and by the Flemish Research Programme Cybersecurity.

References

1. Automatic speaker verification spoofing and countermeasures challenge, `http://www.asvspoof.org/`
2. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access **6**, 14410–14430 (2018)
3. Al-Dujaili, A., Huang, A., Hemberg, E., O'Reilly, U.M.: Adversarial deep learning for robust detection of binary encoded malware. In: S&P Workshops. pp. 76–82. IEEE (2018)

4. Alzantot, M., Balaji, B., Srivastava, M.: Did you hear that? adversarial examples against automatic speech recognition. NIPS Workshop on Machine Deception (2018)
5. Alzantot, M., Sharma, Y., Chakraborty, S., Zhang, H., Hsieh, C.J., Srivastava, M.B.: Genattack: Practical black-box attacks with gradient-free optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1111–1119. ACM (2019)
6. Anderson, H.S., Kharkar, A., Filar, B., Evans, D., Roth, P.: Learning to evade static pe machine learning malware models via reinforcement learning. arXiv:1801.08917 (2018)
7. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. ICLR (2018)
8. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.D.: The security of machine learning. Machine Learning **81**(2), 121–148 (2010)
9. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: CCS. pp. 16–25. ACM (2006)
10. Bhattad, A., Chong, M.J., Liang, K., Li, B., Forsyth, D.A.: Unrestricted adversarial examples via semantic manipulation. ICLR (2020)
11. Biggio, B., g. fumera, Russu, P., Didaci, L., Roli, F.: Adversarial biometric recognition : A review on biometric system security from the adversarial machine-learning perspective. IEEE Signal Processing Magazine **32**(5), 31–41 (2015)
12. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 387–402. Springer (2013)
13. Biggio, B., Fumera, G., Roli, F.: Security evaluation of pattern classifiers under attack. IEEE Transactions on Knowledge and Data Engineering **26**(4), 984–996 (2013)
14. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: ICML. pp. 1467–1474 (2012)
15. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018)
16. Bigham, J.P., Cavender, A.C.: Evaluating existing audio CAPTCHAs and an interface optimized for non-visual users. In: CHI. pp. 1829–1838. ACM (2009)
17. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In: ICLR (2018)
18. Brückner, M., Kanzow, C., Scheffer, T.: Static prediction games for adversarial learning problems. Journal of Machine Learning Research **13**(Sep), 2617–2654 (2012)
19. Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., et al.: Toward trustworthy AI development: Mechanisms for supporting verifiable claims. arXiv:2004.07213 (2020)
20. Brunner, T., Diehl, F., Le, M.T., Knoll, A.: Guessing smart: Biased sampling for efficient black-box adversarial attacks. In: ICCV. pp. 4958–4966 (2019)
21. Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., Kurakin, A.: On evaluating adversarial robustness. arXiv:1902.06705 (2019)
22. Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., Zhou, W.: Hidden voice commands. In: USENIX Security. pp. 513–530 (2016)
23. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: A survey. arXiv:1810.00069 (2018)
24. Chen, J., Jordan, M.I., Wainwright, M.J.: Hopskipjumpattack: A query-efficient decision-based attack. In: S&P. pp. 668–685. IEEE
25. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. p. 15–26. AISec '17, ACM (2017)
26. Chen, S., Carlini, N., Wagner, D.: Stateful detection of black-box adversarial attacks. arXiv:1907.05587 (2019)
27. Croce, F., Hein, M.: Sparse and imperceivable adversarial attacks. In: ICCV. pp. 4724–4732 (2019)

28. Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al.: Adversarial classification. In: KDD. pp. 99–108. ACM (2004)
29. Dosovitskiy, A., Fischer, P., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with exemplar convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(9), 1734–1747 (2015)
30. Elson, J., Douceur, J.R., Howell, J., Saul, J.: Asirra: a captcha that exploits interest-aligned manual image categorization. In: CCS. pp. 366–374. ACM (2007)
31. Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A.: A rotation and a translation suffice: Fooling CNNs with simple transformations. In: NIPS 2017 Workshop on Machine Learning and Computer Security (2017)
32. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Tramèr, F., Prakash, A., Kohno, T., Song, D.: Physical adversarial examples for object detectors. arXiv:1807.07769 (2018)
33. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: CVPR. pp. 1625–1634 (2018)
34. Ferdowsi, A., Challita, U., Saad, W., Mandayam, N.B.: Robust Deep Reinforcement Learning for Security and Safety in Autonomous Vehicle Systems. In: IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC. pp. 307–312 (2018)
35. Fritsch, C., Netter, M., Reisser, A., Pernul, G.: Attacking image recognition captchas. In: International Conference on Trust, Privacy and Security in Digital Business. pp. 13–25. Springer (2010)
36. Gao, H., Lei, L., Zhou, X., Li, J., Liu, X.: The Robustness of Face-Based CAPTCHAs. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. pp. 2248–2255 (2015)
37. Gao, H., Wang, W., Fan, Y.: Divide and Conquer: An Efficient Attack on Yahoo! CAPTCHA. In: IEEE International Conference on Trust, Security and Privacy in Computing and Communications. pp. 9–16 (2012)
38. Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., Yan, J.: The robustness of hollow CAPTCHAs. In: CCS. pp. 1075–1086. ACM (2013)
39. Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., Zhang, P., Zhou, X., Wang, X., Li, J.: A Simple Generic Attack on Text Captchas . NDSS pp. 21–24 (2016)
40. Geirhos, R., Jacobsen, J.H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., Wichmann, F.A.: Shortcut learning in deep neural networks. Nature Machine Intelligence **2**(11), 665–673 (2020)
41. Gilmer, J., Adams, R.P., Goodfellow, I., Andersen, D., Dahl, G.E.: Motivating the rules of the game for adversarial example research. arXiv:1807.06732 (2018)
42. Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., Russell, S.: Adversarial policies: Attacking deep reinforcement learning. In: ICLR (2019)
43. Globerson, A., Roweis, S.: Nightmare at test time: robust learning by feature deletion. ICML (2006)
44. Golle, P.: Machine learning attacks against the asirra captcha. In: SOUPS. ACM (2009)
45. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.D.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. In: ICLR (2014)
46. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
47. Grosse, K., Papernot, N., Manoharan, P., Backes, M., McDaniel, P.: Adversarial examples for malware detection. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security – ESORICS 2017. pp. 62–79. Springer (2017)
48. Hernández-Castro, C.J., R-Moreno, M.D., Barrero, D.F.: Using JPEG to Measure Image Continuity and Break Capy and Other Puzzle CAPTCHAs. IEEE Internet Computing **vol. 19**(6), 46–53 (2015)
49. Hernandez-Castro, C.J., Ribagorda, A., Hernandez-Castro, J.C.: On the strength of egglue and other logic CAPTCHAs. In: SECRYPT. pp. 157–167 (2011)

50. Hernandez-Castro, C.J., Ribagorda, A.: Pitfalls in captcha design and implementation: the math captcha, a case study. Computers & Security **vol. 29**(1), 141–157 (2010)
51. Hernandez-Castro, C.J., Barrero, D.F., R-Moreno, M.D.: A machine learning attack against the civil rights captcha. In: International Symposium on Intelligent Distributed Computing (IDC) (2014)
52. Hernandez-Castro, C.J., Hernandez-Castro, J.C., Stainton-Ellis, J.D., Ribagorda, A.: Shortcomings in captcha design and implementation: Captcha2, a commercial proposal. In: International Network Conference (INC) (2010)
53. Hernández-Castro, C.J., R-moreno, M.D., Barrero, D.F.: Side-channel attack against the Capy HIP. In: International Conference on Emerging Security Technologies (EST). pp. 99–104. IEEE (2014)
54. Hernandez-Castro, C.J., Ribagorda, A., Saez, Y.: Side-channel attack on labeling captchas. In: SECRYPT (2010)
55. Hernández-Castro, C., Li, S., R-Moreno, M.: All about uncertainties and traps: Statistical oracle-based attacks on a new captcha protection against oracle attacks. Computers & Security **92**, 101758 (2020)
56. Hong, S., Chandrasekaran, V., Kaya, Y., Dumitraş, T., Papernot, N.: On the effectiveness of mitigating data poisoning attacks with gradient shaping. arXiv:2002.11497 (2020)
57. Hosseini, H., Poovendran, R.: Semantic adversarial examples. In: CVPR Workshops. pp. 1614–1619 (2018)
58. Hu, W., Tan, Y.: Black-box attacks against rnn based malware detection algorithms. In: AAAI Workshops (2017)
59. Hu, W., Tan, Y.: Generating adversarial malware examples for black-box attacks based on GANs. arXiv:1702.05983 (2017)
60. Huang, S., Papernot, N., Goodfellow, I., Duan, Y., Abbeel, P.: Adversarial attacks on neural network policies. In: ICLR (2017)
61. Huang, W.R., Geiping, J., Fowl, L., Taylor, G., Goldstein, T.: Metapoison: Practical general-purpose clean-label data poisoning. In: NeurIPS (2020)
62. Huang, W., Stokes, J.W.: Mtnet: a multi-task neural network for dynamic malware classification. In: International conference on detection of intrusions and malware, and vulnerability assessment. pp. 399–418. Springer (2016)
63. Huang, X., Kroening, D., Kwiatkowska, M., Ruan, W., Sun, Y., Thamo, E., Wu, M., Yi, X.: Safety and trustworthiness of deep neural networks: A survey. arXiv:1812.08342 (2018)
64. Ilyas, A., Engstrom, L., Athalye, A., Lin, J.: Black-box adversarial attacks with limited queries and information. In: ICML. pp. 2137–2146 (2018)
65. Ilyas, A., Engstrom, L., Madry, A.: Prior convictions: Black-box adversarial attacks with bandits and priors. In: ICLR (2019)
66. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks. In: USENIX Security (2019)
67. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B.: Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In: S&P. pp. 19–35. IEEE (2018)
68. Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.: Adversarial Machine Learning. Cambridge University Press (2018)
69. Kołcz, A., Teo, C.H.: Feature weighting for improved classifier robustness. In: CEAS (2009)
70. Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., Roli, F.: Adversarial malware binaries: Evading deep learning for malware detection in executables. In: EUSIPCO. pp. 533–537. IEEE (2018)
71. Kolosnjaji, B., Zarras, A., Webster, G., Eckert, C.: Deep learning for classification of malware system call sequences. In: Australasian Joint Conference on Artificial Intelligence (2016)
72. Labs, K.: Machine learning methods for malware detection. `https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf` (2020)
73. Laidlaw, C., Feizi, S.: Functional adversarial attacks. NeurIPS (2019)

74. Larson, M., Liu, Z., Brugman, S., Zhao, Z.: Pixel privacy: Increasing image appeal while blocking automatic inference of sensitive scene information. In: Working Notes Proceedings of the MediaEval Workshop (2018)
75. Lin, Y.C., Hong, Z.W., Liao, Y.H., Shih, M.L., Liu, M.Y., Sun, M.: In: IJCAI. p. 3756–3762. AAAI Press (2017)
76. Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S., Leung, V.C.: A survey on security threats and defensive techniques of machine learning: A data driven view. IEEE Access **6**, 12103–12117 (2018)
77. Liu, X., Du, X., Zhang, X., Zhu, Q., Wang, H., Guizani, M.: Adversarial samples on android malware detection systems for iot systems. Sensors **19**(4), 974 (2019)
78. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: ICLR (2017)
79. Liu, Z., Zhao, Z., Larson, M.: Pixel privacy 2019: Protecting sensitive scene information in images. In: Working Notes Proceedings of the MediaEval Workshop (2019)
80. Liu, Z., Zhao, Z., Larson, M.: Who's afraid of adversarial queries? the impact of image modifications on content-based image retrieval. In: ICMR (2019)
81. Lovisotto, G., Eberz, S., Martinovic, I.: Biometric backdoors: A poisoning attack against unsupervised template updating. In: Euro S&P (2019)
82. Lowd, D., Meek, C.: Adversarial learning. In: KDD. pp. 641–647. ACM (2005)
83. Luo, B., Liu, Y., Wei, L., Xu, Q.: Towards imperceptible and robust adversarial example attacks against neural networks. In: AAAI. vol. 32 (2018)
84. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018)
85. Malialis, K., Kudenko, D.: Distributed response to network intrusions using multiagent reinforcement learning. Engineering Applications of Artificial Intelligence **41**, 270–284 (2015)
86. Mitchell, T.M., et al.: Machine learning. 1997. Burr Ridge, IL: McGraw Hill **45**(37), 870–877 (1997)
87. Naor, M.: Verification of a human in the loop or Identification via the Turing Test. `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps` (1996)
88. Nguyen, V.D., Chow, Y.W., Susilo, W.: Attacking Animated CAPTCHAs via Character Extraction, pp. 98–113. Springer (2012)
89. Ni, Z., Paul, S.: A multistage game in smart grid security: A reinforcement learning solution. IEEE Transactions on neural networks and learning systems **30**(9), 2684–2695 (2019)
90. Oh, S.J., Fritz, M., Schiele, B.: Adversarial image perturbation for privacy protection a game theory perspective. In: ICCV. pp. 1491–1500 (2017)
91. Osadchy, M., Hernandez-Castro, J., Hernandez, J., Gibson, S., Dunkelman, O., Pérez-Cabo, D.: No Bot Expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples, With Applications to CAPTCHA Generation. IEEE Transactions on Information Forensics and Security **vol. 12**(11), 2640 – 2653 (2016)
92. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.: Towards the science of security and privacy in machine learning. arXiv:1611.03814 (2016)
93. Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I., Raffel, C.: Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In: ICML. pp. 5231–5240 (2019)
94. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K.: Malware detection by eating a whole exe. In: AAAI (2018)
95. Rajabi, A., Bobba, R.B., Rosulek, M., Wright, C.V., Feng, W.c.: On the (im) practicality of adversarial perturbation for image privacy. Proceedings on Privacy Enhancing Technologies pp. 85–106 (2021)
96. Rozsa, A., Rudd, E.M., Boult, T.E.: Adversarial diversity and hard positive generation. In: CVPR Workshops. pp. 25–32 (2016)
97. Rubinstein, B.I., Nelson, B., Huang, L., Joseph, A.D., Lau, S.h., Rao, S., Taft, N., Tygar, J.: Antidote: understanding and defending against poisoning of anomaly detectors. In: ACM SIGCOMM Conference on Internet Measurement. pp. 1–14. ACM (2009)
98. Sano, S., Otsuka, T., Okuno, H.G.: Solving Google's Continuous Audio CAPTCHA with HMM-Based Automatic Speech Recognition, pp. 36–52. Springer (2013)

99. Santamarta, R.: Breaking gmail's audio captcha. http://blog.wintercore.com/?p=11 (2008), http://blog.wintercore.com/?p=11, accessed on 2010-13-02
100. Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., Madry, A.: Adversarially robust generalization requires more data. In: NeurIPS. pp. 5014–5026 (2018)
101. Schönherr, L., Kohls, K., Zeiler, S., Holz, T., Kolossa, D.: Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In: NDSS (2019)
102. Schultz, M.G., Eskin, E., Zadok, F., Stolfo, S.J.: Data mining methods for detection of new malicious executables. In: S&P. pp. 38–49. IEEE (2001)
103. Serban, A., Poll, E., Visser, J.: Adversarial examples on object recognition: A comprehensive survey. ACM Computing Surveys (CSUR)
104. Shafahi, A., Najibi, M., Ghiasi, M.A., Xu, Z., Dickerson, J., Studer, C., Davis, L.S., Taylor, G., Goldstein, T.: Adversarial training for free! In: NeurIPS. pp. 3353–3364 (2019)
105. Shamsabadi, A.S., Sanchez-Matilla, R., Cavallaro, A.: Colorfool: Semantic adversarial colorization. In: CVPR. pp. 1151–1160 (2020)
106. Shan, S., Wenger, E., Zhang, J., Li, H., Zheng, H., Zhao, B.Y.: Fawkes: Protecting privacy against unauthorized deep learning models. In: USENIX Security. pp. 1589–1604 (2020)
107. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: CCS. p. 1528–1540. ACM (2016)
108. Shet, V.: Street View and reCAPTCHA technology just got smarter. https://security.googleblog.com/2014/04/street-view-and-recaptcha-technology.html (2014), accessed on 2017-08-14
109. Sidorov, Z.: Rebreakcaptcha: Breaking google's recaptcha v2 using google. https://east-ee.com/2017/02/28/rebreakcaptcha-breaking-googles-recaptcha-v2-using-google/ (2017)
110. Sivakorn, S., Polakis, I., Keromytis, A.D.: I am robot: (deep) learning to break semantic image captchas. In: Euro S&P. pp. 388–403. IEEE (2016)
111. Sivakorn, S., Polakis, J., Keromytis, A.D.: I'm not a human : Breaking the google recaptcha (2016)
112. Smith, L.N.: A useful taxonomy for adversarial robustness of neural networks. arXiv:1910.10679 (2019)
113. Steinhardt, J., Koh, P.W.W., Liang, P.S.: Certified defenses for data poisoning attacks. In: NeurIPS. pp. 3517–3529 (2017)
114. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: ICLR (2013)
115. Tam, J., Simsa, J., Hyde, S., von Ahn, L.: Breaking audio captchas. pp. 1625–1632. Curran Associates, Inc. (2008)
116. Tramèr, F., Carlini, N., Brendel, W., Madry, A.: On adaptive attacks to adversarial example defenses. In: NeurIPS (2020)
117. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: USENIX Security. pp. 601–618 (2016)
118. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: ICLR (2019)
119. Vorobeychik, Y., Kantarcioglu, M.: Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **12**(3), 1–169 (2018)
120. Wang, D., Moh, M., Moh, T.S.: Using deep learning to solve google recaptcha v2's image challenges. pp. 1–5 (2020)
121. Wong, E., Schmidt, F., Kolter, Z.: Wasserstein adversarial examples via projected sinkhorn iterations. In: ICML. pp. 6808–6817 (2019)
122. Xiao, C., Li, B., yan Zhu, J., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. In: IJCAI. pp. 3905–3911 (2018)
123. Xiao, C., Zhu, J.Y., Li, B., He, W., Liu, M., Song, D.: Spatially transformed adversarial examples. In: ICLR (2018)
124. Xiao, L., Wan, X., Dai, C., Du, X., Chen, X., Guizani, M.: Security in Mobile Edge Caching with Reinforcement Learning. IEEE Wireless Communications **25**(3), 116–122 (2018)

125. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection. In: ICCV. pp. 1369–1378 (2017)
126. Xu, W., Qi, Y., Evans, D.: Automatically evading classifiers: A case study on pdf malware classifiers. In: NDSS (2016)
127. Yan, J., Ahmad, A.S.E.: A low-cost attack on a microsoft captcha. In: CCS. pp. 543–554. ACM (2008)
128. Yan, Q., Liu, K., Zhou, Q., Guo, H., Zhang, N.: Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided wave. In: NDSS (2020)
129. Yu, D., Deng, L.: Automatic speech recognition. Springer (2016)
130. Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., Xu, W.: Dolphinattack: Inaudible voice commands. In: CCS. pp. 103–117. ACM (2017)
131. Zhang, H., Avrithis, Y., Furon, T., Amsaleg, L.: Smooth adversarial examples. EURASIP Journal on Information Security **2020**(1), 1–12 (2020)
132. Zhao, Z., Liu, Z., Larson, M.: Adversarial color enhancement: Generating unrestricted adversarial images by optimizing a color filter. In: BMVC (2020)
133. Zhao, Z., Liu, Z., Larson, M.: Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In: CVPR. pp. 1039–1048 (2020)
134. Zhou, Y., Yang, Z., Wang, C., Boutell, M.: Breaking google recaptcha v2. J. Comput. Sci. Coll. **34**(1), 126–136 (2018)
135. Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K.: Attacks and design of image recognition captchas. In: CCS. pp. 187–200. ACM (2010)
136. Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K.: Attacks and design of image recognition captchas. In: CCS. pp. 187–200. ACM, ACM (2010)