SoK: Security and Privacy in Machine Learning

Nicolas Papernot*, Patrick McDaniel*, Arunesh Sinha†, and Michael P. Wellman†
* Pennsylvania State University
† University of Michigan
{ngp5056,mcdaniel}@cse.psu.edu, {arunesh,wellman}@umich.edu

Abstract—**Advances in machine learning (ML) in recent years have enabled a dizzying array of applications such as data analytics, autonomous systems, and security diagnostics. ML is now pervasive—new systems and models are being deployed in every domain imaginable, leading to widespread deployment of software based inference and decision making. There is growing recognition that ML exposes new vulnerabilities in software systems, yet the technical community's understanding of the nature and extent of these vulnerabilities remains limited. We systematize findings on ML security and privacy, focusing on attacks identified on these systems and defenses crafted to date. We articulate a comprehensive threat model for ML, and categorize attacks and defenses within an adversarial framework. Key insights resulting from works both in the ML and security communities are identified and the effectiveness of approaches are related to structural elements of ML algorithms and the data used to train them. In particular, it is apparent that constructing a theoretical understanding of the sensitivity of modern ML algorithms to the data they analyze, à la PAC theory, will foster a science of security and privacy in ML.**

1. Introduction

Advances in the science of machine learning (ML) coupled with growth in computational capacities transformed the technology landscape, as embodied by the automation of Machine Learning as a service on commercial cloud platforms. For example, ML-driven data analytics fundamentally altered the practice of health care and finance. Within the security domain, detection and monitoring systems now consume vast amounts of data and extract actionable information that in the past would have been impossible. In spite of these spectacular advances, the technical community's understanding of the vulnerabilities inherent to the design of systems built on ML and the means to defend against them are still in its infancy. There is a broad and pressing call to advance a science of security and privacy in ML [1].

Such calls have not gone unheeded. Many investigations have sought to expand our understanding of the threats, attacks and defenses of systems built on ML. This work, however, is fragmented across several research communities including ML, security, statistics, and theory of computation. As yet there is no unified lexicon or science spanning these disciplines. The fragmentation presents both a motivation and challenge for our effort to systematize knowledge about the myriad of security and privacy issues that involve ML.

In this paper, we develop a unified perspective on this field, based on a threat model that considers characteristics of the attack surface, adversarial goals, and possible defense and attack capabilities particular to systems built on machine learning. This security model serves as a roadmap for surveying knowledge about attacks and defenses of ML systems. We distill major themes and highlight results in the form of take-away messages about this new area of research.

In exploring security and privacy in this domain, it is instructive to view systems built on ML through the prism of the classical confidentiality, integrity, and availability (CIA) model [2]. In this work, *confidentiality* is defined with respect to the model or its data. Attacks on confidentiality attempt to expose the model structure or parameters (which may be highly valuable intellectual property) or the data used to train and test it (e.g., patient data). The latter class of attacks have a potential to impact the privacy of the data source, especially when model users are not trusted, which in cases such as patient clinical data used to train medical diagnostic models may be highly sensitive. We define attacks on the *integrity* as those that induce particular outputs or behaviors of the adversary's choosing. They are often conducted through manipulations of the data on which the ML system trains or predicts. Where those adversarial behaviors attempt to prevent legitimate users from accessing meaningful model outputs or the features of the system itself, such attacks fall within the realm of *availability*.

A second perspective in evaluating security and privacy focuses on attacks and defenses with respect to the *machine learning pipeline*. Here, we consider the lifecycle of a ML-based system from training to inference, and identify the adversarial goals and means at each phase. We observe that attacks on training generally attempt to influence the model by altering or injecting training samples—in essence guiding the learning process towards a vulnerable model. Attacks at inference time (runtime) are more diverse. Adversaries use exploratory attacks to induce targeted outputs, and oracle attacks to extract the model itself.

The science of defenses for machine learning is somewhat less well developed. We consider several defensive goals. The first is robustness to distribution drifts—maintaining performance as far as possible when the training and runtime input distributions differ. The second is to provide formal guarantees of privacy preservation—bounding the amount of information exposed by a learned model.

DOI 10.1109/EuroSP.2018.00035

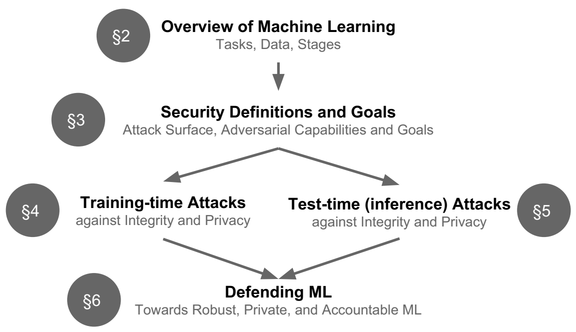
§2
Overview of Machine Learning
Tasks, Data, Stages
§3
Security Definitions and Goals
Attack Surface, Adversarial Capabilities and Goals
§4
Training-time Attacks
against Integrity and Privacy
Test-time (inference) Attacks
against Integrity and Privacy
§5
§6
Defending ML
Towards Robust, Private, and Accountable ML

Figure 1. **Outline for this SoK.**

Third, defenses aim to ensure fairness (preventing biased outputs) and accountability (explanations of why particular outputs were generated, also known as transparency).

In exploring these facets of machine learning attacks and defense, we make the following contributions:

- We introduce a unifying threat model to allow structured reasoning about the security and privacy of systems that incorporate ML (Section 3). This model departs from previous efforts by considering the entire data pipeline, of which ML is a component, instead of ML algorithms in isolation.
- We taxonomize attacks and defenses identified by the various technical communities. Section 4 details the challenges of learning in adversarial settings. Section 5 considers deployed systems. Beyond previous surveys of these areas, we cover attacks mounted with adversarial examples and recent progress towards practical differentially private learning.
- We systematize desirable properties to improve the security and privacy of machine learning (Section 6).

ML methods take several forms, such as classification, regression, and policy learning. For brevity and ease of exposition, we focus much of the current paper on classification. We further state that the related study of the implications of AI on safety in societies is outside the scope of this paper, and refer interested readers to a review by Amodei et al. [3].

We present a systematization of knowledge about security and privacy in ML, focusing on attacks and defenses. Based on our analysis of the ML threat model, we have selected seminal and representative works from the literature. While we attempt to be comprehensive, it is a practical impossibility to cite all works. For instance, we do not cover trusted computing platforms for ML [4]. We begin in the next section by introducing the basic structure and lexicon of ML systems. The paper's organization is shown in Figure 1.

2. About Machine Learning

We start with a brief overview of how systems apply ML. In particular, we compare different kinds of learning tasks, and some specifics of their practical implementation.

2.1. Overview of Machine Learning Tasks

Machine learning automates the analysis of (typically) large data sets, producing models or decision procedures reflecting general relationships found in that data [5]. ML techniques are commonly divided into three classes, characterized by the nature of the data available for analysis.

Supervised learning: Methods that are given training examples in the form of inputs labeled with corresponding outputs are *supervised learning* techniques. Generally, the objective is to induce a model mapping inputs (including unseen inputs) to outputs. If the output domain is categorical, the task is called *classification*, and if cardinal the task is *regression*. Classic examples of supervised learning tasks include: object recognition in images [6], machine translation [7], and spam filtering [8].

Unsupervised learning: When the method is given unlabeled inputs, its task is *unsupervised*. This includes problems such as *clustering* points according to a similarity metric [9], *dimensionality reduction* to project data in lower dimensional subspaces [10], and *model pre-training* [11]. For instance, clustering may be applied to anomaly detection [12].

Reinforcement learning: Data in the form of sequences of actions, observations, and rewards (e.g., runs of video game play) fall in the scope of *reinforcement learning* (RL) [13], [14]. The goal of RL is to produce a policy for acting in the environment, and so it is the subfield of ML concerned with planning and control. RL agents learn by experience exploring their environments. It was reinforcement learning in combination with supervised and unsupervised methods that recently enabled a computer to defeat a human champion at the game of Go [15].

Readers interested in surveys of ML are well served by many books covering this rich topic [5], [16], [17]. Work on ML security and privacy to date has been primarily conducted in supervised settings, as reflected by our presentation in Sections 4 and 5. Since security issues are just as relevant for unsupervised and reinforcement learning, we present results in more general settings when meaningful.

2.2. ML Stages: Training and Inference

It is helpful to separate the *training* stage where a model is learned from input data, from the *inference* stage where the trained model is applied to a task.

Training: Most ML models can be described as functions $h_\theta(x)$ taking an input x and parametrized by a vector $\theta \in \Theta$.[1] The output $h_\theta(x)$ is the model's prediction for x of some property of interest. The input x is typically represented as a vector of values called *features*. The space of functions $\mathcal{H} = \{x \mapsto h_\theta(x) \mid \theta \in \Theta\}$ is the set of candidate hypotheses. A *learning algorithm* utilizes the training data to determine θ. When learning is supervised, the parameters are adjusted to align model predictions $h_\theta(x)$ with the expected output y as indicated by the dataset. This is achieved by minimizing a loss function that captures the dissimilarity of $h_\theta(x)$ and corresponding y. The model performance is

1. Some models (e.g., nearest-neighbor [18]) are non-parametric.

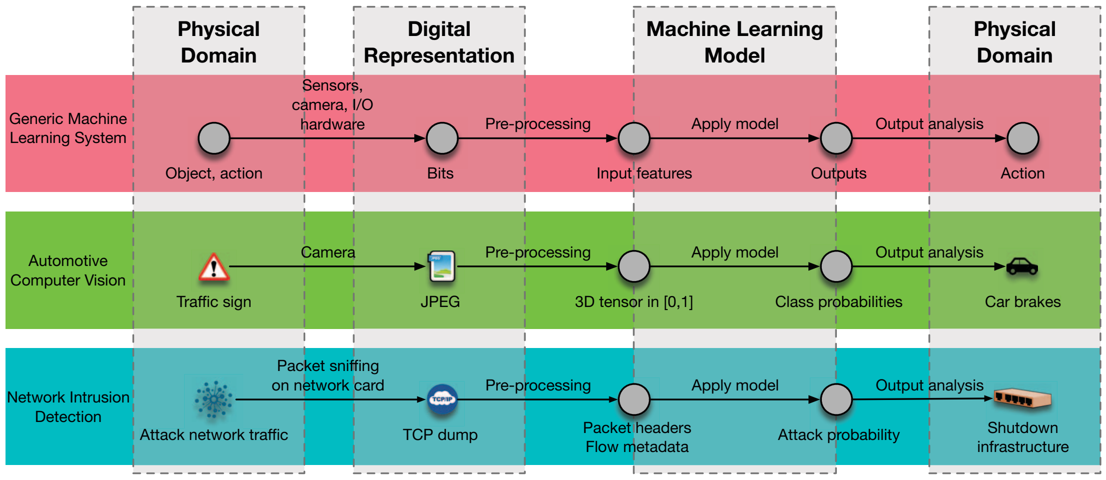

Figure 2. **System's attack surface.** The generic model (top row) is illustrated with two example scenarios (middle and bottom rows): a computer vision model used by an automotive system to recognize traffic signs on the road and a network intrusion detection system.

then validated on a test dataset, which must be disjoint from the training dataset in order to measure the model's *generalization* (performance on unseen data). For a supervised problem, we can evaluate model *accuracy* with respect to *test data*: a set of labeled data held distinct from training data. For example, in malware classification (see above), accuracy would measure the proportion of predictions $h_\theta(x)$ that matched the label y (malware or benign) associated with the executable x in the test dataset. In reinforcement learning, h_θ encodes a policy, and the aim of training is to prescribe actions that yield the highest expected reward for input history x. When learning is done in an online fashion (supervised, unsupervised, or reinforcement), parameters θ are updated as new training points become available.

Inference: Once training completes, the model is deployed to *infer* predictions on inputs unseen during training. That is, the value of parameters θ are fixed, and the model computes $h_\theta(x)$ for new inputs x. In our malware classification example, the model predicts the label for each executable x encountered. The model prediction may take different forms but the most common for classification is a vector assigning a probability for each class of the problem, which characterizes how likely the input is to belong to that class. For our unsupervised network intrusion detection system, the model would instead return the pattern representation $h_\theta(x)$ that corresponds to a new input network traffic x.

3. Threat Model

The security of any system is measured with respect to the adversarial goals and capabilities that it is designed to defend against–the system's *threat model*. In this section, we taxonomize the definition and scope of threat models in ML systems and map the space of security models.

We begin by identifying the attack surface of ML systems to inform where and how an adversary will attempt to subvert the system. Our development of the threat model in following sections builds on previous treatments [19], [20], [21]. We further draw on insights from recent developments like adversarial examples and membership inference attacks.

3.1. The ML Attack Surface

The attack surface of a system built with data and machine learning depends on its purpose. Nevertheless, one can view such systems within a generalized data processing pipeline (see Figure 2, top).

At inference, (a) input features are collected from sensors or data repositories, (b) processed in the digital domain, (c) used by the model to produce an output, and (d) the output is communicated to an external system or user and acted upon. Figure 2 illusrates this pipeline generically (top), and for autonomous vehicle (middle), and network intrusion detection systems (bottom). The ML systems collect sensor inputs (video image, network events) from which features (pixels, flows) are extracted and input to the models. The meaning of the model output (stop sign, network attack) is then interpreted and action taken (stopping the car, filtering future traffic from an IP). Here, the attack surface for the system can be defined with respect to the data processing pipeline. Adversaries can attempt to manipulate the collection of data, corrupt the model, or tamper with the outputs.

Recall that the model is trained using either an offline or online process. The training data used to learn the model includes vectors of features used as inputs during inference, as well as expected outputs for supervised learning or a reward function for reinforcement learning. As discussed below, the means of collection and validation processes offer another attack surface—adversaries who can manipulate the data collection process can do so to induce targeted model behaviors. Similar attacks in an online setting can be quite damaging, where the adversary can slowly alter the model with crafted inputs submitted at runtime. Such attacks on anomaly detectors have been observed in domains such as spam and network intrusion [22].

Attacks against ML can have potentially devastating consequences as they become more practical [23]. Defending against them is necessary to maintain essential properties for high-stakes ML systems: safety (e.g., for autonomous vehicles), security (e.g., intrusion detectors), and correctness (e.g., regulated financial markets).

3.2. Trust Model

The trust model of a any ML-based system is determined in large part by the context of its deployment as it relates to the trust placed in the relevant actors. To abstract a bit, we can think of the several classes of actors relevant to a deployed ML-based system. First, there are *data-owner*s, who are the owners or trustees of the data/environment that the system is deployed within, e.g., an IT organisation deploying a face recognition authentication service. Second, there are *system providers* which construct the system and algorithms, e.g., the authentication service software vendors. Third, there may be *consumers* of the service the system provides, e.g., the enterprise users. Lastly, there are *outsiders* who may have explicit or incidental access to the systems, or may simply be able to influence the system inputs, e.g., other users or adversaries within the enterprise. Note that there may be multiple users, providers, data-owners, or outsiders involved in a given deployment.

A trust model for the given system assigns a level of trust to each actor within that deployment. Any actor can be trusted, untrusted, or partially trusted (trusted to perform or not perform certain actions). The sum of those trust assumptions forms the trust model–and therein identifies the potential ways that bad actors may attack the system. We do not seek in this paper to identify a specific trust model or even set of "good" trust models (a likely impossible endeavour), but to highlight the dangers presented by bad actors. Here, we explore the space of trust models simply by viewing them through the prism of the adversarial capability space (see next Section). In this way, we provide insight into any trust model appropriate for a given deployment.

3.3. Adversarial Capabilities

A threat model is also defined by the actions and information the adversary has at their disposal. The definition of security is made with respect to stronger or weaker adversaries who have more or less access to the system and its data. The term *capabilities* refers to the whats and hows of the available attacks, and defines the possible attack vectors on a threat surface. For instance, in network intrusion detection, an internal adversary may have access to the model used to distinguish attacks from normal behavior, whereas a weaker eavesdropping adversary may have access only to TCP dumps of the network traffic. Here the attack surface is constant, but the attacker with more information is a strictly stronger adversary. We explore the range of attacker capabilities in ML systems as they relate to inference and training phases (see Figure 3).

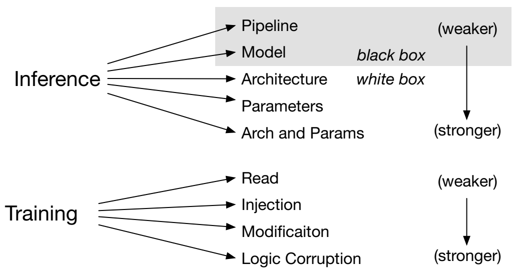

Figure 3. **Adversarial Capabilities.** Adversaries attack ML systems at inference time by exploiting model internal information (white box) or probing the system to infer system vulnerabilities (black box). Adversaries use read or write access to the training data to mimic or corrupt the model.

Inference Phase: Attacks at inference time—*exploratory attacks* [19]—do not tamper with the targeted model but instead either cause it to produce adversary selected outputs (an example of an Integrity attack in the taxonomy of adversarial goals below) or collect evidence about the model characteristics (a Confidentiality attack). As discussed in Section 5, the effectiveness of such attacks is largely determined by the information that is available to the adversary about the model and its use in the target environment.

Inference phase attacks can be classified into either white box or black box attacks. In white box attacks, the adversary has some information about the model or its original training data, possibly due to untrusted actors in the data processing pipeline. White box attacks may be further distinguished by the information used: about the model architecture (algorithm and structure of the hypothesis h), model parameters θ (weights), training data, or combinations of these. The adversary exploits available information to identify where a model is vulnerable. For example, an adversary who has access to the model h and its parameters θ may identify parts of the feature space for which the model has high error, and exploit that by altering an input into that space, as in adversarial example crafting [24].

Conversely black box attacks assume no knowledge about the model. The adversary in these attacks use information about the setting or past inputs to infer model vulnerability. For example, in an oracle attack, the adversary explores a model by providing a series of carefully crafted inputs and observing outputs [25]. Oracle attacks work because a good deal of information about a model can be extracted from input/output pairs, and relatively little information is required because of the transferability property exhibited by many model architectures (See Section 5.2).

Training Phase: Attacks on training attempt to learn, influence, or corrupt the model itself. The simplest and arguably weakest attack on training is simply accessing a summary, a portion, or all of the training data. This could be through explicit attacks or via an untrusted data collection component. Depending on the quality and volume of data, the adversary can create a substitute model (also referred to as a surrogate or auxiliary model) to mount attacks on the victim system. For example, the adversary can use a substitute model to test potential inputs before submitting them to the victim [26]. Note that these attacks are offline attempts at model reconnaissance, and thus may be used to undermine privacy (see below).

There are two broad attack strategies for altering the model. The first alters the training data either by inserting adversarial inputs into the existing training data (injection), possibly as a malicious user, or altering the training data directly (modification) by direct attacks or via an untrusted data collection component. In the case of reinforcement learning, the adversary may modify the environment in which the agent is evolving. Lastly, the adversaries can tamper with the learning algorithm, sometimes easily by colluding with an untrusted ML training component. We refer to these attacks as logic corruption. Obviously, adversaries that alter the learning logic (and thus controls the model itself) are very powerful and difficult to defend against.

3.4. Adversarial Goals

The last piece of a threat model specifies the goals of the adversary. We model desired ends as impacting confidentiality, integrity, and availability (the CIA triad mentioned in Section 1), and adding a fourth property, privacy. An interesting duality emerges when taking a view in this way: attacks on system integrity and availability are closely related in goal and method, as are confidentiality and privacy. Integrity and privacy can both be understood at the level of the ML model, as well as for the entire system deploying it. Availability is however ill defined for a model in isolation but makes sense for the system and environment it operates in. Desirable security properties may also be defined and enforced at the level of this environment. The security of the ML model is a necessary but not sufficient condition for establishing the security of the environment. For instance, integrity and availability of a driverless car's vision model is needed but not sufficient to guarantee availability of the road to other vehicles. This aspect falls outside the scope of our survey, and calls for future treatment similar to the one of Amodei et al. for safety-related issues [3]. We discuss below the range of adversarial goals that relate to each risk.

Confidentiality and Privacy: Attacks on confidentiality and privacy are with respect to the model and data. If the adversary is an untrusted user of the model, it may attempt to extract information about the model. These attacks generally fall under the realm of confidentiality. When the ML model itself represents intellectual property and its consumers are not trusted by the model owner, it requires that the model and its parameters be confidential, e.g., financial market systems [27]. On the contrary, if the model owner is not trusted by model users, these users might want to protect the confidentiality of their data from the model owner or the privacy of their data from attacks mounted by other model users. This is a common setting in medical applications [28]. Regardless of the goal, the attacks and defenses for confidentiality and privacy relate to exposing or preventing the exposure of the model and training data. Distinguishing the two notions is a result of the trust model.

Machine learning models have enough capacity to capture and memorize elements of their training data [29]. As such, it is hard to provide guarantees that participation in a dataset does not harm the privacy of an individual. Potential risks are adversaries performing membership test (to know whether an individual is in a dataset or not) [30], recovering of partially known inputs (use the model to complete an input vector with the most likely missing bits), and extraction of the training data using the model's predictions [29].

Integrity and Availability: Attacks on integrity and availability are with respect to model outputs. Here the goal is to induce model behavior as chosen by the adversary. Attacks attempting to control model outputs are at the heart of integrity attacks—the integrity of the inference process is undermined. For example, attacks that attempt to induce false positives in a face recognition system affect the authentication process's integrity [23]. Closely related, attacks on availability attempt to reduce the quality (e.g., confidence or consistency), performance (e.g., speed), or access (e.g., denial of service). Here again, while the goals of these two classes of attacks may be different, the means by which the adversary achieves them is often similar.

Integrity is essential in ML, and is the center of attention for most performance metrics used: e.g., accuracy [31]. However, researchers have shown that the integrity of ML systems may be compromised by adversaries capable of manipulating model inputs [24] or its training data [32]. First, the ML model's confidence may be targeted by an adversary: reducing this value may change the behavior of the overall system. For instance, an intrusion detection system may only raise an alarm when its confidence is over a specified threshold. Input misprocessing aims at misleading the model into producing wrong outputs for some inputs, either modified at the entrance of the pipeline, or at the input of the model directly. Depending on the task type, the wrong outputs differ. For a ML classifier, it may assign the wrong class to a legitimate image, or classify noise with confidence. For an unsupervised feature extractor, it may produce a meaningless representation of the input. For a reinforcement learning agent, it may act unintelligently given the environment state. However, when the adversary is capable of subverting the input-output mapping completely, it can control the model and the system's behavior. For instance, it may force an automotive's computer vision system to misprocess a traffic sign, resulting in the car accelerating.

Availability is somewhat different than integrity, as it is about the prevention of access to an asset: an output or an action induced by a model output. Hence, the goal of these attacks is to make the model inconsistent or unreliable in the target environment. For example, the goal of the adversary attacking an autonomous vehicle may be to get it to behave erratically or non-deterministically in a given environment. Yet most of the attacks in this space require corrupting the model through training input poisoning and other confidence reduction attacks using many of the same methods used for integrity attacks.

If the system depends on the output of the ML model to take decisions that impact its availability, it may be subject to attacks falling under the broad category of denial of service. Continuing with the previous example, an attack that produces vision inputs that force a autonomous vehicle to stop immediately may cause a denial of service by preventing its owner from using it. More broadly, ML models may also not perform correctly when some of their input features are corrupted or missing [33]. Thus, by denying access to these features we can subvert the system.

4. Training in Adversarial Settings

As parameters θ of the hypothesis h are fine-tuned during learning, the training dataset analyzed is potentially vulnerable to manipulations by adversaries. This scenario corresponds to a *poisoning attack* [19], and is an instance of learning in the presence of non-necessarily adversarial but nevertheless noisy data [34]. Intrusion detection systems are a prevalent example of these settings. Poisoning attacks alter the training dataset by inserting, editing, or removing points with the intent of modifying the decision boundaries of the targeted model [32], thus targeting the learning system's integrity per our threat model from Section 3. It is somewhat obvious that an unbounded adversary can cause the learner to learn any arbitrary function h leading to complete unavailability of the system. Thus, all the attacks below bound the adversary in their attacks [35]. Modifications of the training data can be seen as altering the distribution D that generated the training data, thereby creating a mismatch between the distributions used for training and inference. In Section 6.1, we present a line of work that builds on that observation to propose learning strategies robust to *distribution drifts* [36].

Upon surveying the field, we note that works almost exclusively discuss poisoning attacks against classifiers (supervised models trained with labeled data). Yet, as we strive to generalize our observations to other types of ML tasks (see Section 2), we note that the strategies described below may apply, as many RL algorithms employ supervised components. This is for instance the case for AlphaGo [15].

4.1. Targeting Integrity

Kearns et al. theoretically analyzed the accuracy of learning classifiers when the adversary is allowed to modify training samples with probability β [32]. For large datasets, this adversarial capability can be interpreted as the ability to modify a fraction β of the training data. One of their fundamental results states that achieving an error rate of ϵ at inference requires $\beta \leq \frac{\epsilon}{1+\epsilon}$ for any learning algorithm. For example, to achieve 90% accuracy ($\epsilon = 0.1$) the adversary manipulation rate must be less than 10%. The efforts below explore this result from a practical standpoint and introduce poisoning attacks against ML algorithms. We organize our discussion around the adversarial capabilities highlighted in the preceding section. Unlike some attacks at inference (see Section 5.2), training time attacks almost always require some degree of knowledge about the learning procedure, in order to disrupt it through manipulations of the data.

Label manipulation: When adversaries are able only to modify the labeling information contained in the training dataset, the attack surface is limited: they must find the most harmful labels to perturb in the data given partial or full knowledge of the learning algorithm ran by the defender.

The baseline strategy is to perturb the labels (i.e., randomly draw new labels) for a fraction of the training data. In practice, Biggio et al. showed that this was sufficient to degrade the inference performance of SVM classifiers [37], as long as the adversary randomly flips about 40% of the training labels. It is unclear whether this attack would generalize to multi-class classifiers, with more than 2 output classes (they only considered binary tasks, where swapping the labels is guaranteed to be very harmful to the model).

Heuristics improve the adversary's chances of success. Biggio et al. [37] observe that poisoning points classified with confidence by the model later further degrades the model's performance during inference. In fact, they reduce the ratio of poisoned points needed to reduce accuracy by about 10 percentage points when compared to random label flipping. A similar attack approach has been applied in the context of healthcare [38]. These attacks require that a new ML model be learned for each new candidate poisoning point in order to measure the candidate's impact on the updated model's performance during inference. This high computation cost can be explained by the largely unknown relationship between performance metrics respectively computed on the training and test data. However, for models where such a relationship is better understood, it is possible to find near-optimal sets of labels that need to be flipped—as demonstrated by Xiao et al. with SVMs [39].

Our take-away 4.1. *Bounds on the generalization error (the difference in performance of a model on training and testing distributions) are often loose [40], in which case it is difficult to quantify the impact of poisoning attacks on inference.*

Input manipulation: In this threat model, the adversary can corrupt the input features of training points processed by the model, in addition to its labels. These works assume knowledge of the learning algorithm and training set.

Direct poisoning of the learning inputs: The attack surface of a ML model is often exacerbated when learning is performed online, that is, with new training points added by observing the environment in which the system evolves. Most efforts in this area focus on clustering models, where the intuitive strategy for adversaries is to slowly displace the center of the cluster to have points misclassified at inference.

Kloft et al. insert poisoned points in a training set used for anomaly detection, and demonstrate how this gradually shifts the decision boundary of a centroid model, i.e. a model that classifies a test input as malicious when it is too far from the empirical mean of the training data [22]. This model is learned in an online fashion—new training data is collected at regular intervals and the parameter values θ are computed on a sliding window of that data. Poisoning points are found by solving a linear programming problem that maximizes the displacement of the centroid. This formulation exploits the simplicity of centroid models, which essentially evaluate Euclidean distances to compute the empirical mean of the training data. This attack is unlikely to apply when the relationship between training data and the model is not as explicit. The approach was later revisited in the context of malware clustering [41]: malware is modified to include additional behavioral features that place it between two existing clusters in the model's input domain, thus eventually decreasing the distance between clusters.

In the settings of offline learning, Biggio et al. introduce an attack that identifies poisoning points by gradient ascent on the test error of the model [56]. Adding these inputs to the training set results in a degraded classification accuracy at inference. Their approach is (at least in theory) specific to SVMs, because it relies on the existence of a closed-form formulation of the model's test error, which in their case follows from the assumption that support vectors[2] do not change as a result of the insertion of poisoning points. Mei et al. [57] fall under this class of approaches but derive the gradient ascent formulation differently, by relaxing a bilevel optimization problem (similarly to label flipping attacks like [39]). Later, this same gradient ascent strategy was adapted to feature selection algorithms like LASSO [58].

Such manipulations of the learning inputs are also effective ways to target reinforcement learning agents. Behzadan et al. showed that gradient ascent techniques designed in the context of adversarial examples (see Section 5 for a detailed presentation of these techniques) were able to induce the agent into learning the wrong policy [59].

Indirect poisoning of the learning inputs: Adversaries with no access to the pre-processed data must instead poison the model's training data before its pre-processing (see Figure 2). For instance, Perdisci et al. prevented Polygraph, a worm signature generation tool [60], from learning meaningful signatures by inserting perturbations in worm traffic flows [61]. Polygraph combines a flow tokenizer together with a classifier that determines whether a flow should be in the signature. Polymorphic worms are crafted with noisy traffic flows such that (1) their tokenized representations will share tokens not representative of the worm's traffic flow, and (2) they modify the classifier's threshold for using a signature to flag worms. This attack forces Polygraph to generate signatures with tokens that do not correspond to invariants of the worm's behavior.

4.2. Targeting Privacy and Confidentiality

During training, the confidentiality and privacy of the data and model are not impacted by the fact that ML is used, but rather the extent of the adversary's access to the system hosting them. This is a traditional access control problem, which falls outside the scope of our discussion.

2. Support vectors are the subset of training points that suffice to define the decision boundary of a support vector machine.

5. Inferring in Adversarial Settings

Adversaries may also attack ML systems at inference time, once the deployed model's parameters have been fixed. For instance, the attacker may be targeting an intrusion detection system's whose rules were learned and fixed. Thus, the attacker is interested in mounting a variant of its attack that will instantly evade detection at runtime. Strong *white-box* attackers have access to the model internals (e.g., its architecture and parameters), whereas *black-box* adversaries are limited to interacting with the model as an oracle (e.g., by submitting inputs and observing the model's predictions). In practice, capabilities range on a spectrum between these two extrema. This perspective is used to structure the present section, of which Figure 4 provides an overview. Note that most privacy and confidentiality attacks are motivated in the black-box setting, and seek to expose respectively properties of the data or the model itself (see Section 5.2).

5.1. White-box adversaries

White-box adversaries have varying degrees of access to the model h as well as its parameters θ. This strong threat model allows the adversary to conduct particularly devastating attacks. While it is often difficult to obtain, white-box access is not always unrealistic. For instance, ML models trained on data centers are compressed and deployed to smartphones [62], in which case reverse engineering may enable adversaries to recover the model's internals (e.g., its parameter values) and thus obtain white-box access.

Integrity: To target a system's prediction integrity at inference, adversaries perturb the inputs of the ML model. This can be interpreted as modifying the distribution that generates data at inference. We first describe strategies that require *direct* manipulation of model inputs, and then consider indirect perturbations *resilient to the pre-processing stages* of the system's data pipeline (as illustrated in Figure 2).

Direct manipulation of model inputs: Here, adversaries directly alter the feature values processed by the model. For instance, the adversary's goal may be to have a classifier assign the wrong class to inputs [19]. Szegedy et al. coined the term *adversarial example* to refer to such inputs [24]. Similar to concurrent work [42], they formalize the search for adversarial examples as a minimization problem:

$$\arg\min_r h(x+r) = l \quad \text{s.t.} \quad x^* = x + r \in D \qquad (1)$$

The input x, correctly classified by h, is perturbed with r to produce an adversarial example x^* that remains in the input domain D but is assigned the target label l. When the target l is chosen, the attack is a *source-target misclassification* [21] (also referred to as *targeted* in the literature). When l can be any label different from $h(x)$, the attack is said to be a simple misclassification (or sometimes to be *untargeted*). Attacks constructing adversarial examples differ from one another by the approximation they use to solve Equation 1 when the model h is not convex.

The first class of attack techniques applies existing optimizers. For instance, Szegedy et al. use the L-BFGS

			Integrity		Privacy		
Knowledge of model h_θ	Access to model input x and output $h(x)$	Access to training data	Misprediction	Source-target misprediction	Membership	Training data extraction	Model extraction
White-Box	Full	No	[42], [43], [44]	[24], [21], [44], [45], [46]	[47]	[48]	
	Through pipeline only	No	[49], [50], [23]	[50], [23]			
Black-Box	Yes	No	[51]		[30], [47]	[52], [29]	[53]
	Input x only	Yes	[26], [24], [43]	[24]			[53]
		No	[25], [54]	[55]			
	Through pipeline only	No	[50]				

Figure 4. **Attacks at inference:** all of these works are discussed in Section 5 and represent the threat models explored by the research community.

algorithm [63] to solve Equation 1, which handles the input domain constraint by design. They were the first to find that a wide range of ML models, including deep neural networks with state-of-the-art accuracy on vision tasks were misled by perturbations r imperceptible to humans. Carlini et al. [46] revisited this approach with a different optimizer, Adam, by encoding the domain constraint as a change of variable.

Other techniques make assumptions to solve Equation 1 efficiently. This is notably the case of the *fast gradient sign method* introduced by Goodfellow et al. [43]. A linearization assumption reduces the computation of an adversarial example x^* to: $x^* = x + \epsilon \cdot sign(\nabla_{\vec{x}} J_h(\theta, x, y))$, where J_h is the cost function used to train the model h. Despite the approximation made, a model with close to state-of-the-art performance on MNIST, a widely-used corpus of 70,000 handwritten digits used for validating ML systems for pattern recognition [64], misclassifies 89.4% of this method's adversarial examples. This empirically validates the hypothesis that erroneous model predictions on adversarial examples are likely due to the linear extrapolation made by components of ML models (e.g., individual neurons of a DNN) for inputs far from the training data.

In Equation 1, the minimization of perturbation r can be expressed with different metrics. These formulations all lead to different attacks [21], [44], [65]. The choice of an appropriate metric (often a ℓ_p norm) is problem-dependent. For instance, when crafting malware that evades detection by a ML system, it is easier to introduce perturbations r that only modify a limited subset of the features, rather than making small changes to all features [49]. To this effect, Papernot et al. introduced a Jacobian-based adversarial example algorithm that minimizes the L_0 norm of r [21], i.e. the number of features perturbed. On average, only 4% of the features of an MNIST test set input are perturbed to have it classified in a chosen target class with 97% success—whereas most of the approaches introduced previously perturbed the entire input (albeit by smaller changes) to achieve this success rate.

The variety of algorithms that find adversarial directions suggests that model errors form a continuous space rather than being scattered in small pockets throughout the models' output surface. Warde-Farley and Goodfellow showed that adversarial examples form a space of dimension at least two [66]. Later, Tramèr et al. introduced the Gradient Aligned Adversarial Subspace method, which uses a first-order approximation similar to the one used to define the fast gradient sign method to directly estimate the dimensionality of the dense space formed by adversarial examples [67].

Our take-away 5.1. *Models often extrapolate linearly from the limited subspace covered by the training data [43]. Algorithms can exploit this regularity in directing search toward prospective adversarial regions.*

Indirect manipulation of model inputs: When the adversary cannot directly modify feature values used as model inputs, it must find perturbations preserved by the data pipeline that precedes the classifier in the overall targeted system. Strategies operating in this threat model construct adversarial examples in the *physical domain* stage of Figure 2.

Kurakin et al. showed how printouts of adversarial examples produced by the fast gradient sign algorithm were still misclassified by an object recognition model [50]. They fed the model with photographs of the printouts, thus reproducing the typical pre-processing stage of a computer vision system's data pipeline. They also found these physical adversarial examples to be resilient to pre-processing deformations like contrast modifications or blurring. Sharif et al. applied the approach introduced in [24] to find adversarial examples that are printed on glasses frames, which once worn by an individual result in its face being misclassified by a face recognition model [23]. Adding penalties to ensure the perturbations are physically realizable (i.e., printable) in Equation 1 is sufficient to conduct misclassification attacks (the face is misclassified in any wrong class), and to a more limited extent source-target misclassification attacks (the face is misclassified in a chosen target class).

Our take-away 5.2. *To be resilient to the pipeline's deformations, adversarial examples in physical domains need to introduce adapted, often larger, perturbations. This suggests that enforcing physical realizability and domain constraints may reduce a model's error space.*

Beyond classification: Alfeld et al. [45] examine autoregressive models, where the prediction x_t of a time series depends on previous k realizations of x, that is, $x_t = \sum_{i=1}^{k} c_i x_{t-i}$; such models are prevalent in market prediction. An adversary manipulates the input data with the goal of achieving their desired prediction, given budget constraints. The authors formulate the adversary's manipulation problem as quadratic optimization and provide efficient solutions.

Adversarial examples also apply to reinforcement learning. Huang et al. demonstrated that a RL agent is vulnerable to adversarial perturbations of its environment after it has been trained [68]. Using the fast gradient method (see above), the adversary induces the agent into misbehaving immediately or at a later time—effectively creating “sleeper agents” that behave correctly for several time steps after the environment was perturbed before taking wrong actions.

Our take-away 5.3. *Although research has focused on classification, adversarial example algorithms extend to other settings like reinforcement learning: e.g., the adversary perturbs a frame in a video game to force an agent to take wrong actions.*

Privacy and Confidentiality: Confidentiality attacks in the white-box threat model are trivial because the adversary already has access to the model parameters. As discussed in Section 3, adversaries targeting the privacy of data manipulated by a ML system are interested in recovering information about either the training data. The simplest attack against data consists in performing a membership test, i.e. determining whether a particular input was used in the training dataset of a model. Stronger opponents may seek to extract fully or partially unknown training points. Few attacks operate in the white-box threat model, as the black-box model (see below) is more realistic for privacy.

Ateniese et al. infer statistical information about the training data from a trained model h_θ [48], that is, whether its training data verified a certain statistical property. Their attack generates several datasets, where some exhibit the statistical property and others do not. A model is trained on each dataset independently. The adversary then trains a *meta-classifier*: it takes as its inputs these models and predicts if their dataset verified the statistical property. The meta-classifier is then applied to the model of interest h_θ to fulfill the initial adversarial goal. One limitation is that all classifiers must be trained with the same technique than the model h_θ being attacked.

5.2. Black-box adversaries

When attacking *black-box* systems, adversaries do not know the model internals. This prohibits the strategies described in Section 5.1: for instance, integrity attacks require that the attacker compute gradients defined using the model h and its parameters θ. However, black-box access is perhaps a more realistic threat model, as all it requires is access to the output responses. For instance, an adversary seeking to penetrate a computer network rarely has access to the specifications of its intrusion detection system—but they can often observe how it responds to network events. Similar attacks are key to performing reconnaissance in networks to determine their environmental detection and response policies. We focus on strategies designed irrespectively of the domain ML is being applied to, albeit heuristics specific to certain applications exist, e.g., spam filtering [69], [70].

A common threat model for black-box adversaries is the one of an *oracle*, borrowed from the cryptography community: the adversary may issue queries to the ML model and observe its output for any chosen input. This is particularly relevant in the increasingly popular environment of ML as a Service cloud platforms, where the model is potentially accessible through a query interface. Without access to the training data or ML algorithm, querying the target model and knowledge of the class of target models allows the adversary to reconstruct the model with similar amounts of query data as used in training [71] Thus, a key metric when comparing different attacks is the wealth of information returned by the oracle, and the number of oracle queries.

Integrity: Lowd et al. estimate the cost of misclassification in terms of the number of queries to the black-box model [72]. The adversary has oracle access to the model. A cost function is associated with modifying an input x to a target instance x^*. The cost function is a weighted $l_1$difference between x^* and x. The authors introduce ACRE learnability, which poses the problem of finding the least cost modification to have a malicious input classified as benign using a polynomial number of queries to the ML oracle. It is shown that continous features allow for ACRE learnability while discrete features make the problem NP-hard. Because ACRE learnability also depends on the cost function, it is a different problem from reverse engineering the model. Following up on this thread, Nelson et al. [73] identify the space of convex inducing classifiers—those where one of the classes is a convex set—that are ACRE learnable but not necessarily reverse engineerable.

Direct manipulation of model inputs: Adversaries with access to class probabilities can recover many details of the underlying black-box model, as shown by model extraction attacks (see paragraph below). In these settings, Xu et al. apply a genetic algorithm. The fitness of genetic variants obtained by mutation is defined in terms of the oracle's class probability predictions [51]. The approach evades a random forest and SVM used for malware detection. However, computing the genetic variants is a challenge for problems where the number of input features is larger.

When the adversary cannot access probabilities, it is more difficult to extract information about the decision boundary, a pre-requisite to find input perturbations that result in erroneous predictions. In the following works, the adversary only observes the first and last stage of the pipeline from Figure 2: e.g., the input (which they produce) and the class label in classification tasks. Szegedy et al. first observed *adversarial example transferability* [24]: i.e., the property that adversarial examples crafted to be misclassified by a model are likely to be misclassified by a different model. This transferability property holds even when models are trained on different datasets.

Assuming the availability of surrogate data to the adversary, Laskov et al. explored the strategy of training a substitute model for the targeted one [26]. They exploit a semantic gap to evade a malware PDF detector: they inject additional features that are not interpreted by PDF renderers. As such, their attack does not generalize well to other application domains or models.

Papernot et al. used adversarial example transferability to conduct black-box attacks [25]. They show how attackers can force a remotely hosted ML model to misclassify inputs without access to its architecture, parameters, or training data. The attack trains a substitute model using synthetic inputs generated by the adversary and labeled by querying the oracle. The substitute model is then used to craft adversarial examples that transfer back to (i.e., are misclassified by) the originally targeted model. They force MetaMind, an online deep learning API, to misclassify inputs at rates above 84%. In a follow-up work [54], the attack is generalized to many ML models. For instance, the attack on a logistic regression hosted by Amazon has a success rate of 96%.

This strategy is more likely to create adversarial examples that transfer (i.e., evade the targeted model) in a target class chosen by the adversary if they simultaneously evade several substitute models with different architectures [55].

Our take-away 5.4. *Adversarial examples transfer from one model to another because their individual error spaces are high-dimensional, making it likely that these error spaces will intersect. In addition, two models solving the same ML task with comparable performance are likely to have similar decision boundaries*

Data pipeline manipulation: Kurakin et al. [50] showed experimentally that transferability holds in spite of preprocessing stages of the system's data pipeline. Indeed, they evaluated their physical adversarial examples (i.e., printouts of an adversarial image) both on the model that they were targeting and a different model used by a smartphone app to recognize objects. Their results show that the physically printed perturbations reliably fool both the model they were originally targeting and the second *black-box* model.

Privacy and Confidentiality: In black-box settings, adversaries targeting privacy may pursue the goals discussed in white-box settings: membership attacks and training data extraction. In addition, since the model internals are now unknown to them, extracting model parameters themselves is now a valid goal when targeting model confidentiality.

Membership attacks: This type of adversary is looking to test whether or not a specific point was part of the training dataset analyzed to learn the model's parameters. Shokri et al. show how to conduct this type of attack against black-box models [30]. Their strategy exploits differences in the model's confidence on points that were or were not seen during training. For each class of the targeted black-box model, they train several shadow models. Each shadow model is trained to solve the membership inference test with synthetic samples of the corresponding class. The procedure that generates synthetic data is initialized with a random input and performs hill climbing by querying the original model to find modifications of the input that yield a classification with strong confidence in the class of interest. These synthetic inputs are assumed to be statistically similar to inputs contained in the black-box model's training data.

Hayes et al. extend the attack to the unsupervised learning of*generative models* [47]. Modern generative approaches often learn a neural network that models the distribution of training points. These models are used to generate synthetic data, with numerous applications such as compression or learning with little supervision (i.e., few training labels). The attack of Hayes et al. relies on a specific framework for generative models named *generative adversarial networks* (GAN) [74]. They show that the *discriminator* model in the GAN framework (used during training to evaluate the performance of the generative model by comparing its outputs to samples from the training distribution) can be leveraged in similar ways to the shadow models of Shokri et al. [30]. Given a set of training points, their attack identifies points which the generative model is likely to have been trained on by observing the confidence of the discriminator model.

Training data extraction: Fredrikson et al. present the model inversion attack [52]. For a medicine dosage prediction task, they show that given access to the model and auxiliary information about the patient's stable medicine dosage, they can recover genomic information about the patient. Although the approach illustrates privacy concerns that may arise from giving access to ML models trained on sensitive data, it is unclear whether the genomic information is recovered because of the ML model or the strong correlation between the auxiliary information that the adversary also has access to (the patient's dosage) [75]. Model inversion enables adversaries to extract training data from model predictions [29]. However, the inputs extracted are not specific points of the training dataset, but rather an average representation of the inputs that are classified in a class—similar to what is done by saliency maps [76]. The demonstration is convincing in [29] because each class corresponds to a single individual.

Model extraction: Among direct confidentiality considerations like intellectual property, extracting ML model also has privacy implications—as models have been shown to memorize training data at least partially. Tramer et al. show how to extract parameters of a model from the observation of its predictions [53]. Their attack consists in applying equation solving to recover parameters θ from sets of observed input-output pairs $(x, h_\theta(x))$. While simple, the approach is difficult to scale to scenarios where the adversary loses access to the probabilities returned for each class, i.e. when it can only access the label. This leaves room for future work to make such extraction techniques more practical.

6. Towards Robust, Private, and Accountable Machine Learning Models

After presenting attacks conducted at training in Section 4 and inference in Section 5, we cover efforts at the intersection of security, privacy, and ML that are relevant to their mitigation. We draw parallels between the seemingly unrelated goals of: (a) robustness to distribution drifts, (b) learning privacy-preserving models, and (c) fairness and accountability. Many of these remain largely open problems, thus we draw insights useful for future work.

6.1. Robustness of models to distribution drifts

To mitigate the integrity attacks presented in Section 5, ML needs to be robust to *distribution drifts*: i.e., situations where the training and test distributions differ. Indeed, adversarial manipulations are instances of such drifts. During inference, an adversary might introduce positively connotated words in spam emails to evade detection, thus creating a test distribution different from the one analyzed during training [70]. The opposite, modifying the training distribution, is also possible: the adversary might include an identical keyword in many spam emails used for training, and then submit spam ommiting that keyword at test time.

Defending against training-time attacks: Most defense mechanism at training-time rely on the fact that poisoning samples are typically out of the expected input distribution.

Rubinstein et al. [77] pull from robust statistics to build a PCA-based detection model robust to poisoning. To limit the influence of outliers to the training distribution, they constrain the PCA algorithm to search for a direction whose projections maximize a univariate dispersion measure based on robust projection pursuit estimators instead of the standard deviation. In a similar approach, Biggio et al. limit the vulnerability of SVMs to training label manipulations by adding a regularization term to the loss function, which in turn reduces the model sensitivity to out-of-diagonal kernel matrix elements [37]. Their approach does not impact the convexity of the optimization problem unlike previous attempts [78], [79], which reduces the impact of the defense mechanism on performance.

Barreno et al. make proposals to secure learning [19]. These include the use of regularization in the optimization problems solved to train ML models. This removes some of the complexity exploitable by an adversary. Alternatively, they also propose using obfuscation or disinformation: the defender keeps a holdout set of the data or some details of the model secret. However, this violates security fundamentals, such as the ones stated by Kerckhoffs [80].

Recently, Steindhardt et al. extended this line of work by augmenting the defender with a detection model that attempts to remove data points outside a *feasible set* (i.e., outliers) before the model is learned [81].

Defending against inference-time attacks: The difficulty in attaining robustness to adversarial manipulations at inference stems from the inherent complexity of ML models' output surfaces. Yet, a paradox arises from the observation that this complexity of ML hypotheses is necessary to confer modeling capacity sufficient to train robust models, which may indicate a fundamental disadvantage for the defender. Defending against inference attacks remains largely an open problem. We explain why mechanisms that smooth model outputs in infinitesimal neighborhoods of the training data fail to guarantee integrity. Then, we present defenses effective against larger perturbations.

Defending by gradient masking: Most integrity attacks in Section 5 rely on the adversary being able to find small perturbations that lead to significant changes in the model's output. Thus, a natural defense strategy is to reduce the sensitivity of models to small changes made to their inputs. This sensitivity is estimated by computing first order derivatives

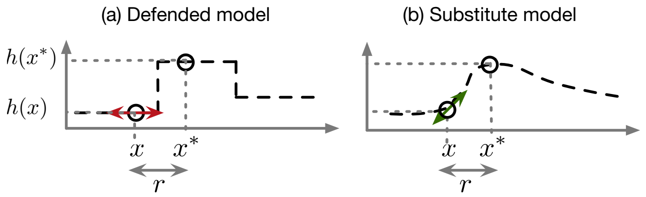

Figure 5. **Evading infinitesimal defenses using transferability:** the defended model is smooth in neighborhoods of training points: i.e., gradients of the model outputs with respect to its inputs are zero and the adversary does not know in which direction to look for adversarial examples. However, the adversary can use the substitute model's gradients to find adversarial examples that transfer back to the defended model. Note that this effect would be exacerbated by models with more than one dimension.

of the model h with respect to its inputs. These gradients are minimized during the learning phase: hence the *gradient masking* terminology. We detail why this intuitive strategy is bound to have limited success.

Gu et al. introduce *deep contractive networks*, a class of models trained using a smoothness penalty [82]. The penalty is defined with the norm of the model's Jacobian matrix, and is approximated layer by layer to preserve computational efficiency. Although contractive models are more robust to adversaries, the penalty reduces their capacity, with consequences on their performance and applicability.

The approach introduced in [83] does not involve the expensive computation of gradient-based penalties. The technique is an adaptation of *distillation* [62], a mechanism designed to compress large models into smaller ones while preserving prediction accuracy. In a nutshell, the large model labels data with class probabilities, which are then used to train the small model. Instead of compression, the authors apply distillation to increase the robustness of DNNs to adversarial samples. They report that the additional entropy in probability vectors (compared to labels) yields models with smoother output surfaces. In experiments with the fast gradient sign method [84] and the Jacobian attack [83], larger perturbations are required to achieve misclassification of adversarial examples by the distilled model. However, Carlini and Wagner [46] identified a variant of the attack in [24], which distillation fails to mitigate.

A simpler variant of distillation, called label smoothing [85], improves robustness to adversarial samples crafted using the fast gradient sign method [86]. It replaces hard class labels (a vector where the only non-null element is the correct class index) with soft labels (each class is assigned a value close to $1/N$ for a N-class problem). Yet, this variant was found to not defend against more precise but computationally expensive Jacobian-based iterative attack [21].

These results suggest limitations of defense strategies that seek to conceal gradient-based information exploited by adversaries. In fact, defensive distillation can be evaded using a black-box attack [25]. We here detail the reason behind this evasion. When applying defense mechanisms that smooth a model's output surface, as illustrated in Figure 5.(a), the adversary cannot craft adversarial examples because the gradients it needs to compute (e.g., the derivative of the model output with respect to its input) have values close to zero. In [25], this is referred to as *gradient masking*. The adversary may instead use a substitute model, illustrated in Figure 5.(b), to craft adversarial examples, since the substitute is not impacted by the defensive mechanism and will still have the gradients necessary to find adversarial directions. Due to the adversarial example transferability property [24] described in Section 5, the adversarial examples crafted using the substitute are also misclassified by the defended model. This attack vector is likely to apply to any defense performing gradient masking, i.e. any mechanism defending against adversarial examples in infinitesimal neighborhoods of the training points.

Our take-away 6.1. *Any defense that tampers with adversarial example crafting heuristics (e.g., by masking gradients used by adversaries) but does not mitigate the underlying erroneous model predictions can be evaded using a transferability-based black-box attack.*

Defending against larger perturbations: Szegedy et al. [24] first suggested injecting adversarial samples, correctly labeled, in the training set as a means to make the model robust. They showed that models fitted with this mixture of legitimate and adversarial samples were regularized and more robust to adversaries using their attack.

This strategy was later made practical by Goodfellow et al. [43]: the fast gradient sign method defines a differentiable and efficiently-computed adversarial objective during training. The defender minimizes the error between the model's predictions on adversarial examples (computed using the current parameter candidates throughout training) and the original labels. For instance, the misclassification rate of a MNIST model is reduced from 89.4% to 17.9% on adversarial examples [43]. Huang et al. [65] developed the intuition behind adversarial training. They formulate a min-max problem between the adversary applying perturbations to each training point to maximize the model's classification error, and the learning procedure attempting to minimize this error. The performance improvements over previous efforts are however often statistically non-significant.

Although adversarial training defends against attacks on which the model is trained, it is weak in the face of adaptive adversaries. For instance, Moosavi et al. [44] use a different heuristic to find adversarial examples when training and attacking. Their evaluation shows that the model is no longer robust in these settings.

Our take-away 6.2. *Even with heuristics like adversarial training, it is impossible to fully cover the ML task's data domain. Hence, future proposals for defending against adversarial examples will almost certainly need to improve the ability of models to be uncertain when predicting far from their training subspace [87].*

6.2. Learning and Inferring with Privacy

One way of defining privacy-preserving models is that they do not reveal any additional information about the subjects involved in their training data. This is captured by *differential privacy* [88], a rigorous framework to analyze the privacy guarantees provided by algorithms. Informally, it formulates privacy as the property that an algorithm's output does not differ significantly statistically for two versions of the data differing by only one record. In our case, the record is a training point and the algorithm the ML model.

A randomized algorithm is said to be (ε, δ) differentially private if for two neighboring training datasets T, T', i.e. which differ by at most one training point, the algorithm A satisfies for any acceptable set S of algorithm outputs:

$$Pr[A(T) \in S] \leq e^{\varepsilon} Pr[A(T') \in S] + \delta \qquad (2)$$

The parameters (ε, δ) define an upper bound on the probability that the output of A differs between T and T'. Parameter ε is a privacy budget: smaller budgets yield stronger privacy guarantees. The second parameter δ is a failure rate for which it is tolerated that the ε bound does not hold.

To provide any form of meaningful privacy, such as differential privacy, it is necessary to randomize part of the ML system's pipeline. This may be done either in the preprocessing stages preceding the model (this falls outside the scope of this paper), while training the model, or at inference by randomizing the predictions that the model outputs.

Training: At training, random noise may be injected to the data, the cost minimized by the learning algorithm, or the values of parameters learned.

An instance of training data randomization is formalized by local privacy [89]. In the scenario where users send data to a centralized server that trains a model with the data collected, *randomized response* protects privacy: users respond to server queries with the true answer at a probability q, and otherwise return a random value with probability $1-q$. Erlingsson et al. showed that this allowed the developers of a browser to collect meaningful and privacy-preserving usage statistics from users [90].

Chaudhuri et al. show that *objective perturbation*, i.e. introducing random noise in the cost function (that measures the error between the model predictions and the expected outputs) minimized during learning can provide ε-differential privacy [91]. This noise is drawn from an exponential distribution and scaled using the model sensitivity[3] Bassily et al. provide improved algorithms and privacy analysis, along with references to many of the works intervening in private learning through cost minimization [92]

Our take-away 6.3. *Precisely quantifying the learning algorithm's sensitivity to training points is necessary to establish differential privacy guarantees. For non-convex models (e.g., neural nets), current loose bounds on sensitivity require that learning be heavily randomized to protect data—often at the expense of utility.*

3. In differential privacy research, sensitivity denotes the maximum change in the model output when one of its training points is changed. This is not identical to the sensitivity of ML models to adversarial perturbations (see Section 5).

Shokri et al. showed that large-capacity models like deep neural networks trained with multi-party computations from noisy parameter values provide differential privacy guarantees [93]. In centralized settings (a single entity trains the model), an approach introduced by Abadi et al. guarantees stronger differential privacy bounds. It randomly perturbs gradients computed by the learning algorithm before they are applied to update parameter values [94]. Under different assumptions, namely the availability of public and unlabeled data whose privacy does not need to be protected, it is possible to further improve the strength of the privacy protection on sensitive (labeled) data [95], [96]. To begin, an ensemble of teacher models is learned on (disjoint) partitions of the training data. These teachers make predictions on the public unlabeled data, and their outputs are aggregated in a noisy fashion to produce a single label prediction with differential privacy guarantee. This newly labeled dataset serves as a training set for a student model. Given that this model was trained on private labels, it may be deployed publicly.

Inference: To provide differential privacy, the ML's behavior may also be randomized at inference by introducing noise to predictions. Yet, this degrades the accuracy of predictions, since the amount of noise introduced increases with the number of inference queries answered by the ML model. Note that different forms of privacy, that often fall under the realm of test data confidentiality, can be provided during inference. For instance, Dowlin et al. use homomorphic encryption [97] to encrypt the data in a form that allows a neural network to process it without decrypting it [98]. Although, this does not provide differential privacy, it protects the confidentiality of each individual input when the model owner is not trusted by a model user. The main limitations are the performance overhead and the restricted set of arithmetic operations supported by homomorphic encryption, which introduce additional constraints in the architecture design of the ML model.

6.3. Fairness and Accountability in ML

The opaque nature of ML generates concerns regarding a lack of fairness and accountability of model predictions. This is key in applications like banking or healthcare [99]. In addition, legal frameworks like the European Data Protection Regulation require that companies provide an explanation for algorithm predictions if they were made using data considered sensitive or private [100]. In the interest of space, we do not provide a comprehensive survey of rapid technical progress made towards fairness and accountability, which would necessitate a dedicated SoK. We focus here on work relevant to previously discussed notions of security (e.g., data poisoning) and privacy (e.g., differential privacy).

Fairness: In the ML pipeline from Figure 2, *fairness* is relevant to the action taken in the physical domain based on the model prediction. It should not nurture discrimination against specific individuals [101], [102]. Training data is one source of bias in ML. For instance, a dishonest data collector might adversarially attempt to manipulate the learning into producing a model that discriminates against certain groups. Historical data also inherently reflects social biases [103]. Another source of bias is the learning algorithm itself, which can be adapted to provide guarantees for subpopulations of the training data [104]. These guarantees express a particular definition of fairness: e.g., equal or unbiased treatment [105]. They however introduce trade-offs between the performance of a model and its fairness [101], [106].

To learn fair models, Zemel et al. learn an intermediate representation that encodes a sanitized variant of the data, as first discussed in [107]. Edwards et al. showed that fairness could be achieved by learning in competition with an adversary trying to predict the sensitive variable from the fair model's prediction [108]. They find connections between fairness and privacy, as their approach also applies to the task of removing sensitive annotations from images. We expect future work at the intersection of fairness and topics discussed in this paper to be fruitful. For instance, connections between fairness and security have recently been drawn; techniques like adversarial example algorithms were applied to estimate how representative of a class a particular input is, which led to the identification of racial biases in popular image datasets [109].

Accountability: *Accountability* explains ML predictions with the model internals h_θ. Few models are interpretable by design, i.e., match human reasoning [110], [111]. Datta et al. introduced *quantitative input influence* measures to estimate the influence of specific inputs on the model output [112]. Later, influence functions were exploited to mount poisoning attacks against deep learning by introducing ambiguity in the model's training data [113]. Another avenue for accountability is to compute inputs that the ML model is most sensitive to. *Activation maximization* synthesizes inputs that highly activate specific neurons of a neural network [114]. The challenge lies in producing human-interpretable synthetic inputs that faithfully represent the model's behavior [115]. Activation maximization is also relevant to model failures like adversarial examples. In fact, salient inputs that maximally activate particular models are produced with heuristics similar to those commonly used to identify input directions that yield adversarial examples misclassified by a model. On the one hand, techniques used for accountability and transparency are likely to yield improved attack techniques because they increase the adversary's understanding of how the model's decisions are made. On the other hand, they also contribute to building a better understanding of the impact of training data on the model learned by ML algorithm, which is beneficial to privacy-preserving ML.

7. Conclusions

The security and privacy of machine learning is an active yet nascent area. We explored the attack surface of systems built upon ML. That analysis yields a natural structure for reasoning about their threat models. In the large, a vast body of work from diverse scientific communities paints a picture that many vulnerabilities of ML and the countermeasures used to defend against them are as yet unknown—yet a science for understanding them is slowly emerging.

Take-aways from this systematization of knowledge point towards varying—but related—notions of sensitivity. Characterizing the sensitivity of learning algorithms to their training data is essential to privacy-preserving ML. Similarly, controlling the sensitivity of deployed models to data they perform inference on is required for secure ML. Central to these two notions, the sensitivity of generalization error (i.e., the gap between performance on training and test data) remains poorly understood for many models (including neural nets) and calls for further research.

Acknowledgments

We thank Martín Abadi, Z. Berkay Celik, Ian Goodfellow, Damien Octeau, and Kunal Talwar for feedback on early versions of this document. We also thank Megan McDaniel for taking good care of our diet before the deadline. We would also like to thank Seda Guerses for shepherding our paper. Nicolas Papernot is supported by a Google PhD Fellowship in Security. Research was supported in part by the Army Research Laboratory, under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA), and the Army Research Office under grant W911NF-13-1-0421. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon.

References

[1] W. House, “Preparing for the future of artificial intelligence,” *Executive Office of the President, National Science and Technology Council, Committee on Technology*, 2016.

[2] C. P. Pfleeger and S. L. Pfleeger, *Analyzing Computer Security: A Threat/Vulnerability/Countermeasure Approach*. Prentice Hall, 2012.

[3] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, “Concrete problems in AI safety,” *arXiv preprint arXiv:1606.06565*, 2016.

[4] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, “Oblivious multi-party machine learning on trusted processors,” in *25th USENIX Security Symposium*, 2016.

[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[8] H. Drucker, D. Wu, and V. N. Vapnik, “Support vector machines for spam categorization,” *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[10] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.

[11] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?” *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[12] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[13] J. Hu and M. P. Wellman, “Nash Q-learning for general-sum stochastic games,” *Journal of Machine Learning Research*, vol. 4, pp. 1039–1069, 2003.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[16] C. M. Bishop, “Pattern recognition,” *Machine Learning*, 2006.

[17] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” 2016, Book in preparation for MIT Press (www.deeplearningbook.org).

[18] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[19] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, “Can machine learning be secure?” in *ACM Symposium on Information, Computer and Communications Security*, 2006, pp. 16–25.

[20] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, “Adversarial machine learning,” in *4th ACM Workshop on Security and Artificial Intelligence*, 2011, pp. 43–58.

[21] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in *1st IEEE European Symposium on Security and Privacy*, 2016.

[22] M. Kloft and P. Laskov, “Online anomaly detection under adversarial impact,” in *13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 405–412.

[23] 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 *23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1528–1540.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *International Conference on Learning Representations*, 2014.

[25] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against deep learning systems using adversarial examples,” *arXiv preprint arXiv:1602.02697*, 2016.

[26] N. Šrndić and P. Laskov, “Practical evasion of a learning-based classifier: A case study,” in *IEEE Symposium on Security and Privacy*, 2014, pp. 197–211.

[27] R. J. Bolton and D. J. Hand, “Statistical fraud detection: A review,” *Statistical Science*, vol. 17, pp. 235–249, 2002.

[28] T. C. Rindfleisch, “Privacy, information technology, and health care,” *Communications of the ACM*, vol. 40, no. 8, pp. 92–100, 1997.

[29] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[30] R. Shokri, M. Stronati, and V. Shmatikov, “Membership inference attacks against machine learning models,” *arXiv preprint arXiv:1610.05820*, 2016.

[31] D. M. Powers, “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,” *Journal of Machine Learning Technologies*, vol. 2, pp. 37–63, 2011.

[32] M. Kearns and M. Li, “Learning in the presence of malicious errors,” *SIAM Journal on Computing*, vol. 22, no. 4, pp. 807–837, 1993.

[33] A. Globerson and S. Roweis, “Nightmare at test time: Robust learning by feature deletion,” in *23rd International Conference on Machine Learning*, 2006, pp. 353–360.

[34] N. Manwani and P. S. Sastry, “Noise tolerance under risk minimization,” *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.

[35] B. Nelson and A. D. Joseph, "Bounding an attack's complexity for a simple learning model," in *First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, 2006.

[36] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 97–106.

[37] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian Conference on Machine Learning*, 2011, pp. 97–112.

[38] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 6, pp. 1893–1905, 2015.

[39] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *20th European Conference on Artificial Intelligence*, 2012, pp. 870–875.

[40] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[41] B. Biggio, K. Rieck, D. Ariu, C. Wressnegger, I. Corona, G. Giacinto, and F. Roli, "Poisoning behavioral malware clustering," in *Workshop on Artificial Intelligence and Security*, 2014, pp. 27–36.

[42] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 387–402.

[43] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3d International Conference on Learning Representations*, 2015.

[44] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[45] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *30th AAAI Conference on Artificial Intelligence*, 2016, pp. 1452–1458.

[46] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.

[47] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Evaluating privacy leakage of generative models using generative adversarial networks," *arXiv preprint arXiv:1705.07663*, 2017.

[48] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[49] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," in *22nd European Symposium on Research in Computer Security*, 2017.

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

[51] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Network and Distributed Systems Symposium*, 2016.

[52] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *23rd USENIX Security Symposium*, 2014, pp. 17–32.

[53] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium*, 2016, pp. 601–618.

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

[55] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[56] B. Biggio, B. Nelson, and L. Pavel, "Poisoning attacks against support vector machines," in *29th International Conference on Machine Learning*, 2012.

[57] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *AAAI*, 2015, pp. 2871–2877.

[58] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1689–1698.

[59] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," *arXiv preprint arXiv:1701.04143*, 2017.

[60] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 226–241.

[61] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif, "Misleading worm signature generators using deliberate noise injection," in *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 2006, pp. 15–pp.

[62] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS-14 Workshop on Deep Learning and Representation Learning*. arXiv:1503.02531, 2014.

[63] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[64] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 1998.

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

[66] D. Warde-Farley and I. Goodfellow, "Adversarial perturbations of deep neural networks," *Advanced Structured Prediction, T. Hazan, G. Papandreou, and D. Tarlow, Eds*, 2016.

[67] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[68] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[69] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters." in *CEAS*, 2004.

[70] D. Lowd and C. Meek, "Good word attacks on statistical spam filters." in *CEAS*, 2005.

[71] Y. Vorobeychik and B. Li, "Optimal randomized classification in adversarial settings," in *13th International Conference on Autonomous Agents and Multi-Agent Systems*, 2014, pp. 485–492.

[72] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 641–647.

[73] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. Tygar, "Query strategies for evading convex-inducing classifiers," *Journal of Machine Learning Research*, vol. 13, pp. 1293–1332, 2012.

[74] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[75] F. McSherry, "Statistical inference considered harmful," 2016. [Online]. Available: https://github.com/frankmcsherry/blog/blob/master/posts/2016-06-14.md

[76] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*. Springer, 2014, pp. 818–833.

[77] B. I. P. 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 Conference on Internet measurement*, 2009, pp. 1–14.

[78] G. Stempfel and L. Ralaivola, "Learning SVMs from sloppily labeled data," in *International Conference on Artificial Neural Networks*. Springer, 2009, pp. 884–893.

[79] L. Xu, K. Crammer, and D. Schuurmans, "Robust support vector machine training via convex outlier ablation," in *Twenty-First AAAI National Conference on Artificial Intelligence*, vol. 6, 2006, pp. 536–542.

[80] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, pp. 5–83, 1883.

[81] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," *arXiv preprint arXiv:1706.03691*, 2017.

[82] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proceedings of the 2015 International Conference on Learning Representations*. Computational and Biological Learning Society, 2015.

[83] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 37th IEEE Symposium on Security and Privacy*. IEEE, 2016.

[84] N. Papernot and P. McDaniel, "On the effectiveness of defensive distillation," *arXiv preprint arXiv:1607.05113*, 2016.

[85] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv preprint arXiv:1512.00567*, 2015.

[86] D. Warde-Farley and I. Goodfellow, "Adversarial perturbations of deep neural networks," in *Advanced Structured Prediction*, T. Hazan, G. Papandreou, and D. Tarlow, Eds., 2016.

[87] Y. Li and Y. Gal, "Dropout inference in bayesian neural networks with alpha-divergences," *arXiv preprint arXiv:1703.02914*, 2017.

[88] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[89] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011.

[90] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *21st ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 1054–1067.

[91] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.

[92] R. Bassily, A. Smith, and A. Thakurta, "Differentially private empirical risk minimization: Efficient algorithms and tight error bounds," *arXiv preprint arXiv:1405.7085*, 2014.

[93] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1310–1321.

[94] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[95] J. Hamm, P. Cao, and M. Belkin, "Learning privately from multiparty data," *arXiv preprint arXiv:1602.03552*, 2016.

[96] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*, 2016.

[97] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[98] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 201–210.

[99] I. W. P. Consortium *et al.*, "Estimation of the warfarin dose with clinical and pharmacogenetic data," *N Engl J Med*, vol. 2009, no. 360, pp. 753–764, 2009.

[100] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a" right to explanation"," *arXiv preprint arXiv:1606.08813*, 2016.

[101] J. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent tradeoffs in the fair determination of risk scores," *arXiv preprint arXiv:1609.05807*, 2016.

[102] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," *arXiv preprint arXiv:1703.06856*, 2017.

[103] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California Law Review*, vol. 104, 2016.

[104] M. B. Zafar, I. Valera, M. Gomez Rodriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," *arXiv preprint arXiv:1507.05259*, 2017.

[105] M. Kearns, "Fair algorithms for machine learning," in *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, 2017, pp. 1–1.

[106] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq, "Algorithmic decision making and the cost of fairness," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 797–806.

[107] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 2012, pp. 214–226.

[108] H. Edwards and A. Storkey, "Censoring representations with an adversary," *arXiv preprint arXiv:1511.05897*, 2015.

[109] P. Stock and M. Cisse, "Convnets and imagenet beyond accuracy: Explanations, bias detection, adversarial examples and model criticism," *arXiv preprint arXiv:1711.11443*, 2017.

[110] B. Letham, C. Rudin, T. H. McCormick, D. Madigan *et al.*, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model," *The Annals of Applied Statistics*, vol. 9, no. 3, pp. 1350–1371, 2015.

[111] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1135–1144.

[112] A. Datta, S. Sen, and Y. Zick, "Algorithmic transparency via quantitative input influence," in *Proceedings of 37th IEEE Symposium on Security and Privacy*, 2016.

[113] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *arXiv preprint arXiv:1703.04730*, 2017.

[114] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, 2009.

[115] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, pp. 1–23, 2016.