SoK: Towards the Science of Security and Privacy in Machine Learning

Nicolas Papernot*, Patrick McDaniel*, Arunesh Sinha†, and Michael 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 rapid and 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 recent 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. We conclude by formally exploring the opposing relationship between model accuracy and resilience to adversarial manipulation. Through these explorations, we show that there are (possibly unavoidable) tensions between model complexity, accuracy, and resilience that must be calibrated for the environments in which they will be used.**

I. Introduction

The coming of age of the science of machine learning (ML) coupled with advances in computational and storage capacities have transformed the technology landscape. For example, ML-driven data analytics have fundamentally altered the practice of health care and financial management. Within the security domain, detection and monitoring systems now consume massive amounts of data and extract actionable information that in the past would have been impossible. Yet, 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 the security and privacy in ML.

Such calls have not gone unheeded. A number of activities have been launched to understand the threats, attacks and defenses of systems built on machine learning. However, work in this area is fragmented across several research communities including machine learning, security, statistics, and theory of computation, and there has been few efforts to develop a unified lexicon or science spanning these disciplines. This 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. We introduce a unified threat model that considers the attack surface and adversarial goals and capabilities of systems built on machine learning. The security model serves as a roadmap in the following sections for exploring attacks and defenses of ML systems. We draw major themes and highlight results in the form of take-away messages about this new area of research. We conclude by providing a theorem of the "no free lunch" properties of many ML systems—identifying where there is a tension between complexity and resilience to adversarial manipulation, how this tension impacts the accuracy of models and the effect of the size of datasets on this trade-off.

In exploring security and privacy in this domain, it is instructive to view systems built on machine learning through the prism of the classical confidentiality, integrity, and availability (CIA) model. In this work, confidentiality is defined with respect to the model or its training 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 it, e.g., patient data. The latter class of attacks have a potential to impact the privacy of the data source, e.g., the privacy of patient clinical data used to train medical diagnostic models is often of paramount importance. Conversely, we define attacks on the integrity as those that induce particular outputs or behaviors of the adversary's choosing. Where those adversarial behaviors attempt to prevent access to 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 are somewhat less well developed. Here we consider several defensive goals. First, we consider methods at training and inference time that are robust to distribution drifts–the property that ensures that the model performs adequately when the training and

runtime input distributions differ. Second, we explore models that provide formal privacy preserving guarantees–the property that the amount of data exposed by the model is bounded by a privacy budget (expressed in terms of differential privacy). Lastly, we explore defenses that provide 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 machine learning. This model, presented in Section III, 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 as informed elements of PAC learning theory. Section IV details the challenges of in adversarial settings and Section V considers trained and deployed systems. Section VI presents desirable properties to improve the security and privacy of ML.
- In Section VII, we introduce a *no free lunch* theorem for adversarial machine learning. It characterizes the tradeoff between accuracy and robustness to adversarial efforts, when learning from limited data.

Note that ML systems address a great many different problem domains, e.g., classification, regression and policy learning. However, for brevity and ease of exposition, we focus much of the current paper on ML classification. We further state that the related study of the implications of ML and AI on safety in societies is outside the scope of this paper, and refer interested readers to the comprehensive review by Amodei et al. [1].

The remainder of this paper focuses on a systematization of the knowledge of security and privacy in ML. While there is an enormous body of work in many aspects of this research, for space we focus on attacks and defenses. As a result of a careful analysis of the threat model, we have selected seminal and representative works that illustrate the branches of this research. While we attempt to be exhaustive, it is a practical impossibility to cite all works. For instance, we do not cover trusted computing platforms for ML [2]. We begin below by introducing the basic structure and lexicon of ML systems.

II. About Machine Learning

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

A. An Overview of Machine Learning Tasks

Machine learning provides automated methods of analysis for large sets of data [3]. Tasks solved with machine learning techniques are commonly divided into three types. These are characterized by the structure of the data analyzed by the corresponding learning algorithm.

Supervised learning: Methods that induce an association between inputs and outputs based on training examples in the form of inputs labeled with corresponding outputs are *supervised learning* techniques. If the output data is categorical, the task is called *classification*, and real-valued output domains define *regression* problems. Classic examples of supervised learning tasks include: object recognition in images [4], machine translation [5], and spam filtering [6].

Unsupervised learning: When the method is given unlabeled inputs, its task is *unsupervised*. Unsupervised learning considers problems such as *clustering* points according to a similarity metric [7], *dimensionality reduction* to project data in lower dimensional subspaces [8], and *model pre-training* [10]. For instance, clustering may be applied to anomaly detection [11].

Reinforcement learning: Methods that learn a policy for action over time given sequences of actions, observations, and rewards fall in the scope of *reinforcement learning* [12], [13]. Reinforcement learning can be viewed as the subfield of ML concerned with planning and control. 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 [14].

Readers interested in a broad survey of ML are well served by many books covering this rich topic [3], [15], [16]. Work on ML security and privacy to date has for the most part conducted in supervised settings, especially in the context of classification tasks, as reflected by our presentation in Sections IV and V below. Since security issues are just as relevant for unsupervised and reinforcement learning tasks, we strive to present results in more general settings when meaningful.

B. Data Collection: Three Use Cases

Before one can learn a model that solves a task of interest, training data must be collected. This consists in gathering a generally large set of examples of solutions to the task that one wants to solve with machine learning. For each of the task types introduced above, we describe one example of a task and how the corresponding training dataset would be collected.

The first example task is to classify software executables in two categories: malicious and benign. This is a supervised classification problem, where the model must learn some mapping between inputs (software executables) and categorical outputs (this binary task only has two possible classes). The training data comprises a set of labeled instances, each an executable clearly marked as malicious or benign [17].

Second, consider the task of extracting a pattern representative of normal activity in a computer network. The training data could consist of TCP dumps [18]. Such a scenario is commonly encountered in anomaly-based network intrusion detection [19]. Since the model's desired outputs are not given along with the input—that is, the TCP dumps are not associated with any pattern specification—the problem falls under the scope of unsupervised learning.

Finally, consider the same intrusion-detection problem given access to metrics of system state indicator (CPU load, free memory, network load, etc.) [20]. This variant of the intrusion detector can then be viewed as an agent and the system state

indicator as rewards for actions taken based on a prediction made by the intrusion detector (e.g., shut down part of the network infrastructure). In this form, the scenario then falls under the reinforcement learning tasks.

C. Machine Learning Empirical Process

We describe the general approach taken to create a machine learning model solving one of the tasks described above.

Training: Once the data is collected and pre-processed, a ML model is chosen and trained. Most[1] ML models can be seen as parametric functions $h_\theta(x)$ taking an input x and a parameter vector θ. The input x is often represented as a vector of values called *features*. The space of functions $\{\forall\theta, x \mapsto h_\theta(x)\}$ is the set of candidate hypotheses to model the distribution from which the dataset was sampled. A *learning algorithm* analyzes the training data to find the value(s) of parameter(s) θ. When learning is supervised, the parameters are adjusted to reduce the gap between model predictions $h_\theta(x)$ and the expected output indicated by the dataset. In reinforcement learning, the agent adjusts its policy to take actions that yield the highest reward. The model performance is then validated on a test dataset, which must be disjoint from the training dataset in order to measure the model's *generalization*. For a supervised problem like malware classification (see above), the learner computes the model *accuracy* on a test dataset, i.e. the proportion of predictions $h_\theta(x)$ that matched the label y (malware or benign) associated with the executable x in the dataset. When learning is done in an online fashion, 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: i.e., the value of parameters θ are fixed, and the model computes $h_\theta(x)$ for new inputs x. In our running example, the model would predict whether an executable x is more likely to be malware or benign. 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.

D. A Theoretical Model of Learning

Next, we formalize the semantics of supervised ML algorithms. We give an overview of the Probably Approximately Correct (PAC) model, a theoretical underpinning of these algorithms, here and later use the model in Sections IV, V, and VI to interpret attacks and defenses. Such an interpretation helps discover, from specific attacks and defenses, general principles of adversarial learning that apply across all supervised ML.

PAC model of learning: PAC learning model has a very rich and extensive body of work [22]. Briefly, the PAC model states that data points (x, y) are samples obtained by sampling from a fixed but unknown probability distribution D over the space $Z = X \times Y$. Here, X is the space of feature values and Y is the space of labels (e.g., $Y = \{0, 1\}$ for classification or $Y = \mathbb{R}$ for regression). The mapping from X to Y is captured by a function $h : X \to Y$ associated with a loss function $l_h : X \times Y \to \mathbb{R}$ which captures the error made by the prediction $h(x)$ when the true label is y. Examples include the hinge loss [23] used in SVMs or the cross-entropy loss [3]. The learner aims to learn a function h^* from a family of functions $\mathcal{H}$ such that the the expected loss (also called risk) $r(h) = E_{x,y\sim D}[l_h(x, y)]$ is minimal, that is, $h^* \in \operatorname{argmin}_{h\in\mathcal{H}} r(h)$.

Of course, in practice D is not known and only samples $\vec{z} = z_1, \ldots, z_n$ (the training data) is observed. The learning algorithm then uses the empirical loss $\hat{r}(h, \vec{z}) = \frac{1}{n}\sum_{i=1}^{n} l_h(x_i, y_i)$, where $z_i = (x_i, y_i)$ as a proxy for the expected loss and finds the h that is close to $\hat{h} \in \operatorname{argmin}_{h\in\mathcal{H}} \hat{r}(h, \vec{z})$. Thus, all supervised learning algorithm perform this *empirical risk minimization* (ERM) with the loss function varying across different algorithms. The PAC guarantee states that:

$$P(|r(h^*) - r(\hat{h})| \leq \epsilon) \geq 1 - \delta \tag{1}$$

where the probability is over samples $\vec{z}$ used to learn $\hat{h}$. This guarantee holds when two pre-conditions are met: **[Condition 1: Uniform bound]** given enough samples (called the sample complexity, which depends on ϵ, δ above) that enable a uniform bound of the difference between the actual and empirical risk for all functions in $\mathcal{H}$, and **[Condition 2: Good ERM]** $\hat{h}$ is close to the true empirical risk minimizer $\hat{h}$.

Statistical learning is primarily concerned with the uniform bound pre-condition above, wherein a good ERM is assumed to exist and the goal is to find the sample complexity required to learn certain classes of function with certain loss function.

The training step in supervised learning algorithms performs the ERM step. The accuracy measured on the test data in machine learning typically estimates the ϵ (or some error value correlated with ϵ). In particular, the train test procedure relies on the assumption that training and test data arise from the same, though unknown, distribution D and more importantly the distribution faced in an actual deployment in the inference step is also D. Later we show that most attacks arise from an adversarial modification of D either in training or in inference resulting in a mismatch between the distribution of data used in the learning and the inference phases.

Another noteworthy point is that the PAC guarantee (and thus most ML algorithms) is only about the expected loss. Thus, for most data points (x, y) that lie in low probability regions, the predictor $\hat{h}(x)$ can be far from the true y yet the output $\hat{h}$ could have high accuracy (as measured by test accuracy) because the accuracy is an estimate of the expected loss. In the extreme, a learning accuracy of 100% could be achieved by predicting correctly in the positive probability region with lot of misclassification in the zero probability regions (or more precisely sets with measure zero; a detailed discussion of this fact is present in a recent paper [24]). An adversary may exploit such misclassification to its advantage. We elaborate on the two points above later in Section VII.

[1] A few models are non-parametric: for instance the nearest neighbor [21].

III. Threat Model

The security of any system is measured with respect the adversarial goals and capabilities that it is designed to defend against–the systems' *threat model*. In this section we taxonomize the definition and scope of threat models in machine learning systems and map the space of security models. We begin by identifying the threat surface of systems built on machine learning to inform where and how an adversary will attempt to subvert the system under attack. For the purpose of exposition of the following Sections, we expand upon previous approaches at articulating a threat model for ML [25], [26].

A. The ML Attack Surface

The attack surface of a system built with data and machine learning is reflective of its purpose. However, one can view systems using ML within a generalized data processing pipeline (see Figure 1, 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. To illustrate, consider a generic pipeline, autonomous vehicle, and network intrusion detection systems in Figure 1 (middle and bottom). These systems collect sensor inputs (video image, network events) from which model features (pixels, flows) are extracted and used within 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 and processing of data, corrupt the model, or tamper with the outputs.

Recall that the training of the model is performed using either an offline or online process. In an offline setting, training data is collected or generated. 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. The training data may also include additional features not available at runtime (referred to as privileged information in some settings [27]). 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 (such as may be encountered in reinforcement learning) can be quite damaging, where the adversary can slowly alter the model with crafted inputs submitted at runtime (e.g., using false training [28]). Online attacks such as these have been commonly observed in domains such as SPAM detection and network intrusion detection [28].

B. 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.

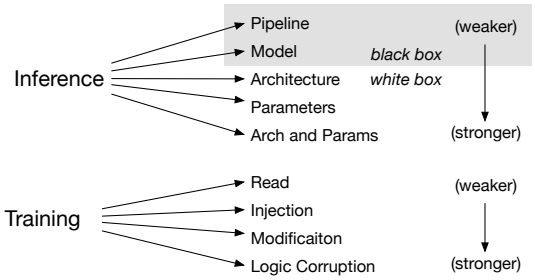

Fig. 2. **Adversarial Capabilities:** adversaries attack ML systems at inference time by exploiting model internal information (white box) or probing the system to infer system vulnerabilities (black box). Adversaries use read or write access to the training data to mimic or corrupt the model.

The term capabilities refers to the whats and hows of the available attacks, and indicates the attack vectors one may use on a threat surface. For instance, in the network intrusion detection scenario, an internal adversary may have access to the model used to distinguish attacks from normal behavior, whereas a weaker eaves-dropping adversary would only have access to TCP dumps of the network traffic. Here the attack surface remains the same for both the attacks, but the former attacker is assumed to have much more information and is thus a strictly "stronger" adversary. We explore the range of attacker capabilities in machine learning systems as they relate to inference and training phases (see Figure 2).

Inference Phase: Attacks at inference time—*exploratory attacks* [25]—do not tamper with the targeted model but instead either cause it to produce adversary selected outputs (incorrect outputs, see Integrity attacks below) or simply use the attack to collect evidence about the model characteristics (reconnaissance, see privacy below). As explored at length in Section V, the effectiveness of such attacks are largely determined by the amount of 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, e.g., ML algorithm h, model parameters θ, network structure, or summary, partial, or full training data. Grossly speaking, this information can be divided into attacks that use information about the model architecture (algorithm and structure of the hypothesis h), model parameters θ (weights), or both. The adversary exploits available information to identify where a model may be exploited. 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 to into that space, e.g., adversarial example crafting [30].

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 a oracle attack, the adversary explores a model by providing a series of carefully crafted inputs and observing outputs [31]. Oracle attacks work because a good

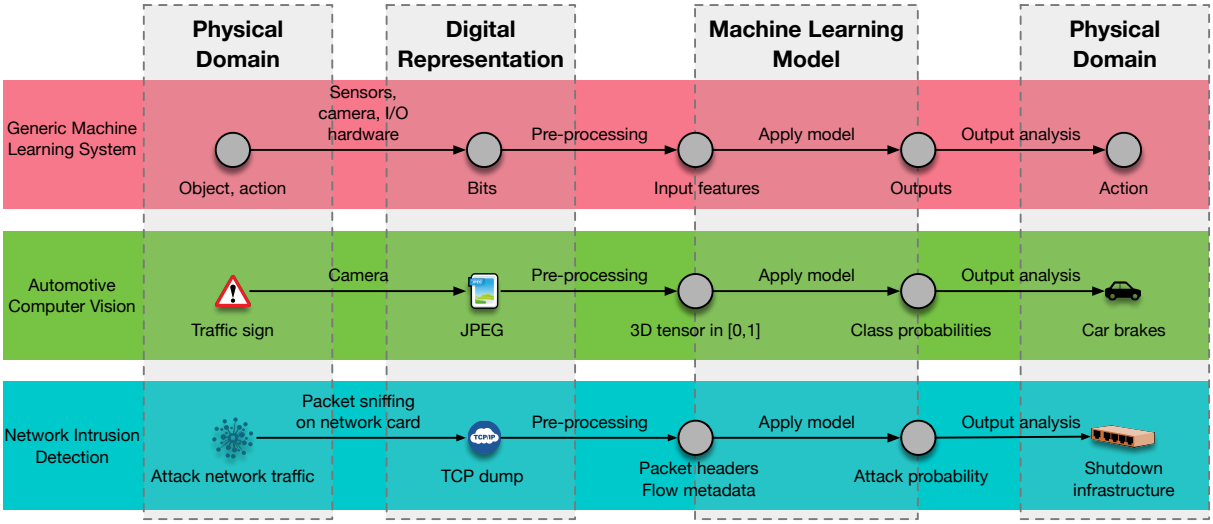

Fig. 1. **System's attack surface:** the generic model (top row) is illustrated with two example scenarios (bottom rows): a computer vision model used by an automotive system to recognize traffic signs on the road and a network intrusion detection system.

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 V-B).

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, partial or all of the training data. Here, depending on the quality and volume of training data, the adversary can create a substitute model (also referred to as a surrogate or auxiliary model) to use 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 system [32]. 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) or altering the training data directly (modification). 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. We refer to these attacks as logic corruption. Obviously, any adversary that can alter the learning logic (and thus controls the model itself) is very powerful and difficult to defend against.

C. Adversarial Goals

The last piece of a threat model is an articulation of the goals of the adversary. We adopt a classical approach to modeling adversarial goals by modeling desired ends as impacting confidentiality, integrity, and availability (called a CIA model), and adding a fourth property, privacy. Interestingly, a 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.

As is often the case in security, ML systems face three types of risks: failure to provide integrity, availability, and privacy. Integrity and privacy can both be understood at the level of the ML model itself, as well as for the entire system deploying it. Availability is however ill defined for a model in isolation but makes sense for the ML-based system as a whole. 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. Put another way, the attacks achieving these goals attempt to extract information about the model or training data as highlighted above. When the model itself represents intellectual property, it requires that the model and its parameters be confidential, e.g., financial market systems [33]. In other contexts it is imperative that the privacy of the training data be preserved, e.g., medical applications [34]. Regardless of the goal, the attacks and defenses for them relate to exposing or preventing the exposure of the model and training data.

Machine learning models have enough capacity to capture and memorize elements of their training data [35]. 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) [36], 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 [35].

Integrity and Availability: Attacks on integrity and ability 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 [37].

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 [38]. However, researchers have shown that the integrity of ML systems may be compromised by adversaries capable of manipulating model inputs [30] or its training data [39]. 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–the asset being 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 completely stopping traffic on the highway. More broadly, machine learning models may also not perform correctly when some of their input features are corrupted or missing [40]. Thus, by denying access to these features we can subvert the system.

IV. 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* [25], and is an instance of learning in the presence of non-necessarily adversarial but nevertheless noisy data [41]. Poisoning attacks alter the training dataset by inserting, editing, or removing points with the intent of modifying the decision boundaries of the targeted model [39], thus targeting the learning system's integrity per our threat model from Section III. 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 [42]. Also, in the PAC model, 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 VI-A, we present a line of work that builds on that observation to propose learning strategies robust to *distribution drifts* [43].

Upon surveying the field, we note that works almost exclusively discuss poisoning attacks against classifiers (supervised models trained with labeled data). However, as we strive to generalize our observations to other types of machine learning tasks (see Section II), we note that the strategies described below may apply, as for instance many algorithms used for reinforcement learning use supervised submodels.

A. Targeting Integrity

Kearns et al. formally analyzed PAC-learnability when the adversary is allowed to modify training samples with probability β [39]. For large datasets this adversarial capability can be interpreted as the ability to modify a fraction β of the training data. One of the fundamental results in the paper states that achieving ϵ learning accuracy (in the PAC model) 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 explored this result from a practical standpoint and introduced 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 V-B), training time attacks require some degree of knowledge about the learning model, in order to find manipulations of the data that are damaging to the learned model.

Label manipulation: When adversaries are only able to modify the labeling information contained in the training dataset, the attack surface is limited: they must find the most harmful label given partial or full knowledge of the learning algorithm ran by the defender. The baseline strategy is to randomly perturb the labels, i.e. select a new label for a fraction of the training data by drawing from a random distribution. Biggio et al. showed that this was sufficient to degrade inference performance of classifiers learned with SVMs [44], 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 $k > 2$ output classes (these results only considered problems with $k = 2$ classes, where swapping the labels is guaranteed to be very harmful to the model). The authors also demonstrate that perturbing points classified with confidence by the model in priority is a compelling heuristic to later degrade the model's performance during inference. It

reduces the ratio of poisoned points to 30% for comparable drops in inference accuracy on the tasks also used to evaluate random swaps. A similar attack approach has been applied in the context of healthcare [45]. As is the case for the approach in [44], this attack requires that a new ML model be learned for each new candidate poisoning point in order to measure the proposed point's impact on the updated model's performance during inference. This high computation cost is due to the largely unknown relationship between performance metrics respectively computed on the training and test data.

Our take-away IV.1. *Search algorithms for poisoning points are computationally expensive because of the complex and often poorly understood relationship between a model's accuracy on training and test distributions.*

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

Direct poisoning of the learning inputs: Kloft et al. show that by inserting points in a training dataset used for anomaly detection, they can gradually shift the decision boundary of a simple 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 [28]. The detection model is learned in an online fashion—new training data is collected at regular intervals and the parameter values θ are computed based on a sliding window of that data. Therefore, injection of poisoning data in the training dataset is a particularly easy task for adversaries in these online settings. Poisoning points are found by solving a linear programming problem that maximizes the displacement of the centroid (empirical mean of the training data). This formulation is made possible by the simplicity of the centroid model, which essentially evaluates an Euclidean distance.

Our take-away IV.2. *The poisoning attack surface of a ML system is often exacerbated when learning is performed online, i.e. new training points are added by observing the environment in which the system evolves.*

In the settings of offline learning, Biggio et al. introduce an attack that also inserts inputs in the training set [46]. These malicious samples are crafted using a gradient ascent method that identifies inputs corresponding to local maxima in the test error of the model. Adding these inputs to the training set results in a degraded classification accuracy at inference. Their approach is 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. introduce a more general framework for poisoning, which finds the optimal changes to the training set in terms of cardinality or the Frobenius norm, as long as the targeted ML model is trained using a convex loss (e.g., linear and logistic regression or SVMs) and its input domain is continuous [47]. Their attack is formulated as two nested optimization problems, which are solved by gradient descent after reducing them to a single optimization problem using the inner problem's Karush-Kuhn-Tucker conditions.

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 1). For instance, Perdisci et al. prevented Polygraph, a worm signature generation tool [48], from learning meaningful signatures by inserting perturbations in worm traffic flows [49]. 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. Later, Xiao et al. adapted the gradient ascent strategy introduced in [46] to feature selection algorithms like LASSO [50].

B. Targeting Privacy

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

V. Inferring in Adversarial Settings

Adversaries may also attack ML systems at inference time. In such settings, they cannot poison the training data or tamper with the model parameters. Hence, the key characteristic that differentiates attackers is their capability of accessing (but not modifying) the deployed model's internals. *White-box* adversaries possess knowledge of the internals: e.g., the ML technique used or the parameters learned. Instead, *black-box* access is a weaker assumption corresponding to the capability of issuing queries to the model or collecting a surrogate training dataset. Black-box adversaries may surprisingly jeopardize the integrity of the model output, but white-box access allows for finer-grain control of the outputs. With respect to privacy, most existing efforts focus on the black-box (oracle) attacks that expose properties of the training data or the model itself.

A. 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 and thus obtain white-box access.
Integrity: To target a white-box system's prediction integrity, adversaries perturb the ML model inputs. In the theoretical

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

Knowledge of model h_θ	Access to model input x and output $h(x)$	Access to training data	Integrity		Privacy		
			Misprediction	Source-target misprediction	Membership	Training data extraction	Model extraction
White-Box	Full	No	[51], [52], [53]	[30], [26], [54]		[55]	
	Through pipeline only	No	[56], [57], [37]	[37]			
Black-Box	Yes	No	[58]		[36]	[59]	[60]
	Input x only	Yes	[32], [30], [52]				[60]
		No	[31], [61]				
	Through pipeline only	No	[57]				

Fig. 3. **Attacks at inference:** all of these works are discussed in Section V and represent the threat models explored by the research community.

PAC model, this can be interpreted as modifying the distribution that generates data during inference. Our discussion of attacks against classifiers is two-fold: (1) we describe strategies that require *direct* manipulation of model inputs, and (2) we consider indirect perturbations *resilient to the pre-processing stages* of the system's data pipeline. Although most of the research efforts study classification tasks, we conclude with a discussion of regression and reinforcement learning.

Direct manipulation of model inputs: Here, adversaries alter the feature values processed by the ML model directly. When the model is a classifier, the adversary seeks to have it assign a wrong class to perturbed inputs [25]. Szegedy et al. coined the term *adversarial example* to refer to such inputs [30]. Similar to concurrent work [51], they formalize the search for adversarial examples as the following minimization problem:

$$\arg\min_{r} h(x+r) = l \quad \text{s.t.} \quad x + r \in D \tag{2}$$

The input x, correctly classified by h, is perturbed with r such that the resulting adversarial example $x^* = x + r$ remains in the input domain D but is assigned the target label l. This is a *source-target misclassification* as the target class $l \neq h(x)$ is chosen [26]. For non-convex models h like DNNs, the authors apply the L-BFGS algorithm [63] to solve Equation 2. Surprisingly, DNNs with state-of-the-art accuracy on object recognition tasks are misled by small perturbations r.

To solve Equation 2 efficiently, Goodfellow et al. introduced the *fast gradient sign method* [52]. A linearization assumption reduces the computation of an adversarial example x^* to:

$$x^* = x + \epsilon \cdot sign(\nabla_{\vec{x}} J_h(\theta, x, y)) \tag{3}$$

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[3] 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 its canonical form, the technique is designed for misclassification (in any class differing from the correct class), but it can be extended to choose the target class.

Our take-away V.1. *Adversarial examples exist in half-spaces of the model's output surface because of the overly linear extrapolation that models, including non-linear ones, make outside of their training data [52], [65].*

Follow-up work reduced the size of a perturbation r according to different metrics [53], [66]. Papernot et al. introduced a Jacobian-based adversarial example crafting algorithm that minimizes the number of features perturbed, i.e. the L_0 norm of r [26]. 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. This proves essential when the ML model has a discrete input domain for which only a subset of the features can be modified easily by adversaries. This is the case of malware detectors: in this application, adversarial examples are malware applications classified as benign [56].

Classifiers always output a class, even if the input is out of the expected distribution. It is therefore not surprising that randomly sampled inputs can be constrained to be classified in a class with confidence [52], [67]. The security consequences are not so important since humans would not classify these samples in any of the problem's classes. Unfortunately, training models with a class specific to rubbish (out of distribution) samples does not mitigate adversarial examples [52].

Indirect manipulation of model inputs: When the adversary cannot directly modify feature values used as inputs of the ML model, it must find perturbations that are 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 1.

Kurakin et al. showed how printouts of adversarial examples produced by the fast gradient sign algorithm were still misclassified by an object recognition model [57]. 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 [30] 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 [37]. Adding penalties to ensure the perturbations are physically realizable (i.e., printable) in Equation 2 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 V.2. *To be resilient to the pipeline's deformations, adversarial examples in physical domains*

[3]The MNIST dataset [64] is a widely used corpus of 70,000 handwritten digits used for validating image processing machine learning systems.

need to introduce adapted, often larger, perturbations.

As a natural extension to [67] (see above), it was shown that rubbish audio can be classified with confidence by a speech recognition system [68]. Consequences are not as important in terms of security then [37]: the audio does not correspond to any legitimate input expected by the speech system or humans.

Beyond classification: While most work has focused on attacking classifiers, Alfeld et al. [54] look at autoregressive models. An autoregressive model is one 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 widely used in market predictions. The adversary can manipulate the input data with the goal of achieving their desired prediction, given budget constraints for the adversary. The author's formulate the adversary's manipulation problem as a quadratic optimization problem and provide efficient solutions for it.

Adversarial behaviors were also considered in reinforcement learning, albeit not to address security but rather to improve the utility of models. Pinto et al. improve a model for grasping objects by introducing a competing model that attempts to snatch objects before the original model successfully grasps them [69]. The two models are trained, à la Generative Adversarial Networks [70], with competitive cost functions.

Our take-away V.3. *Although research has focused on classification problems, algorithms developed to craft adversarial examples naturally extend to other settings like reinforcement learning: e.g., the adversary perturbs a video game frame to force an agent to take wrong actions.*

Privacy: As discussed in Section III, adversaries targeting the privacy of a ML system are commonly interested in recovering information about either the training data or the learned model itself. The simplest attack 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 when considering privacy.

Ateniese et al. infer statistical information about the training dataset from a trained model h_θ [55]: i.e., they analyze the model to determine whether its training data verified a certain statistical property. Their attack generates several datasets, where some exhibit the statistical property and others don't. 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 finally 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 h_θ.

B. Black-box adversaries

When performing attacks against *black-box* systems, adversaries do not know the model internals. This prohibits the strategies described in Section V-A: 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 the intrusion detection system deployed– 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 [71], [29], e.g., spam filtering.

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. A PAC based work shows that with no 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 amount of query data as used in training [72] 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 [73]. 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. [74] 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: It has been hypothesized that in classification, adversaries with access to class probabilities for label outputs are only slightly weaker than white-box adversaries. In these settings, Xu et al. apply a computationally expensive genetic algorithm. The fitness of genetic variants obtained by mutation is defined in terms of the oracle's class probability predictions [58]. The approach evades a random forest and SVM used for malware detection.

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 1: e.g., the input (which they produce) and the class label in classification tasks. Szegedy et al. first observed *adversarial example transferability*: i.e., the property that adversarial ex-

amples 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, Srndic et al. explored the strategy of training a substitute model for the targeted one [32]. 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 the cross-model transferability of adversarial samples [30], [52] to design a black-box attack [31]. They demonstrated 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—are misclassified by—the originally targeted model. They force a DNN trained by MetaMind, an online API for deep learning, to misclassify inputs at a rate of 84.24%. In a follow-up work [61], they show that the attack generalizes to many ML models by having a logistic regression oracle trained by Amazon misclassify 96% of the adversarial examples crafted.

Our take-away V.4. *Black-box attacks make it more difficult for the adversary to choose a target class in which the perturbed input will be classified by the model, when compared to white-box settings.*

Data pipeline manipulation: Using transferability, Kurakin et al. [57] demonstrated that physical adversarial example (i.e., printouts of an adversarial example) can also mislead an object recognition model included in a smarphone app, which differs from the one used to craft the adversarial example. These findings suggest that black-box adversaries are able to craft inputs misclassified by the ML model despite the pre-processing stages of the system's data pipeline.

Privacy: In black-box settings, adversaries targeting privacy may pursue the goals already 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.

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 parameter values. Shokri et al. show how to conduct this type of attack, named *membership inference*, against black-box models [36]. Their strategy exploits differences in the model's response to points that were or were not seen during training. For each class of the targeted black-box model, they train a shadow model, with the same ML technique. Each shadow model is trained to solve the membership inference test for 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 a class of the problem. These synthetic inputs are assumed to be statistically similar to the inputs contained in the black-box model's training dataset.

Training data extraction: Fredrikson et al. present the model inversion attack [59]. 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 observed model predictions [35]. However, the input extracted is not actually a specific point 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 [35] because each class corresponds to a single individual.

Model extraction: Among other considerations like intellectual property, extracting ML model 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 [60]. Their attack is conceptually simple: it consists in applying equation solving to recover parameters θ from sets of observed input-output pairs $(x, h_\theta(x))$. However, the approach does not scale to scenarios where the adversary looses access to the probabilities returned for each class, i.e. when it can only access the label. This leaves room for future work to improve upon such extraction techniques to make them practical.

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

After presenting attacks conducted at training in Section IV and inference in Section V, we cover efforts at the intersection of security, privacy, and ML that are relevant to the mitigation of these previously discussed attacks. 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.

A. Robustness of models to distribution drifts

To mitigate the integrity attacks presented in Section V, 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 [29]. 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. Within the PAC framework, a

distribution drift violates the assumption that more training data reduces the learning algorithm's error rate. We include a PAC-based analysis of learning robust to distribution drifts.

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 [44]. 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 [25]. 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 (see Section VII). The authors also mention an attack detection strategy based on isolating a special holdout set to detect poisoning attempts. Lastly, they suggest the use of disinformation with for instance honeypots [80] and randomization of the ML model behavior.

Defending against inference-time attacks: The difficulty in attaining robustness to adversarial manipulations at inference, i.e. malicious test distribution drifts, stems from the inherent complexity of output surfaces learned by ML techniques. Yet, a paradox arises from the observation that this complexity of ML hypotheses is necessary to confer modeling capacity sufficient to train robust models (see Section VII). Defending against attacks at inference 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 more effective strategies that make models robust to larger perturbations of their inputs

Defending by gradient masking: Most integrity attacks in Section V rely on the adversary being able to find small perturbations that lead to significant changes in the model output. Thus, a natural class of defenses seeks to reduce the sensitivity of models to small changes made to their inputs. This sensitivity is estimated by computing first order derivatives 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 because of adversarial example transferability.

Gu et al. introduce a new ML model, which they name *deep contractive networks*, trained using a smoothness penalty [81]. The penalty is defined with the Frobenius norm of the model's Jacobian matrix, and is approximated layer by layer to preserve computational efficiency. This approach was later generalized

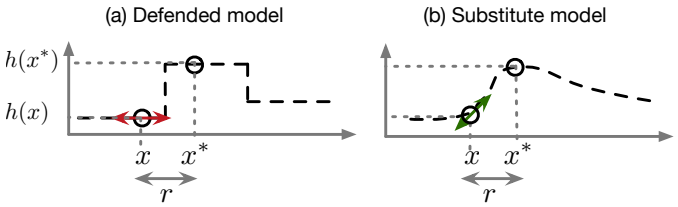

Fig. 4. **Evading infinitesimal defenses using transferability:** the defended model is very smooth in neighborhoods of training points: i.e., gradients of the model outputs with respect to its inputs are zero and the adversary does not know in which direction to look for adversarial examples. However, 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.

to other gradient-based penalties in [82], [83]. Although Gu et al. show that contractive models are more robust to adversaries, the penalty greatly reduces the capacity of these models, with consequences on their performance and applicability.

The approach introduced in [84] 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 [85] and the Jacobian attack [84], larger perturbations are required to achieve misclassification of adversarial examples by the distilled model. However, [86] identified a variant of the attack of [26] which distillation fails to mitigate on one dataset.

A simpler variant of distillation, called label smoothing [87], improves robustness to adversarial samples crafted using the fast gradient sign method [88]. 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). However this variant was found to not defend against more computation expensive but precise attacks like the Jacobian-based iterative attack [26].

These results suggest limitations of defense strategies that seek to conceal gradient-based information exploited by adversaries. In fact, Papernot et al. report that defensive distillation can be evaded using a black-box attack [31]. We here detail the reason behind this evasion. When applying defense mechanisms that smooth a model's output surface, as illustrated in Figure 4.(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 [31], this is referred to as *gradient masking*. The adversary may instead use a substitute model, illustrated in Figure 4.(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 [30] described

in Section V, 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 VI.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. [30] 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 future adversaries. The efficiency of the fast gradient sign method allows for the integration of an adversarial objective during training. The adversarial objective minimizes the error between the model's prediction on the adversarial example and the original sample label. This *adversarial training* continuously makes the model more robust to adversarial examples crafted with the latest model parameters. Goodfellow et al. show that this reduces the misclassification rate of a MNIST model from 89.4% to 17.9% on adversarial examples [52].

Huang et al. [66] developed the intuition behind adversarial training, i.e. penalize misclassification of adversarial examples. 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 the model's misclassification error:

$$\min_{h} \sum_{i} \max_{\|r^{(i)}\| \leq c} l(h(x_i + r^{(i)}), y_i) \tag{4}$$

They solve the problem using stochastic gradient descent [89]. Their experimentation shows improvements over [52], but they are often statistically non-significative. The non-adaptiveness of adversarial training explains some of the results reported by Moosavi et al. [53], where they apply the defense with an attack and evaluate robustness with another one.

Our take-away VI.2. *In adversarial training, it is essential to include adversarial examples produced by all known attacks, as the defensive training is non-adaptive.*

Interpreting robust learning in the PAC model: As stated earlier, inference attacks can be interpreted in the PAC model as the adversary choosing a different data distribution during inference from the one used in training. Thus, an approach to handle such adversaries is to modify the distribution D that is used to generate the training data so that training data samples reflect a probability distribution that maybe encountered during inference, i.e., a distribution that places more probability mass on possibly adversarially classified data. The additions of adversarial samples in the training data [30], [53] or modifying the training loss function [66] can be viewed as modifying the training distribution D towards such an end.

Recall that the PAC model captures the fact that learning algorithms only optimize for expected loss and hence existence of mis-classfied instances can never be ruled out completely. Thus, a formal approach of modifying the training data must also consider the adversary's cost in modifying the distribution in order to tractably deal with adversarial manipulation. Game theory is a natural tool to capture such defender-adversary interaction. Next, we use the Stackelberg game framework to capture the adversarial interaction. It is a model for defender-adversary interaction where the defender lays out her strategy (in this paper the classifier) and the adversary responds optimally (choose a least cost evasion). Stackelberg games also allow for scalability compared to the corresponding simultaneous move game [90], [91].

Bruckner et al. [92] first recognized that test data manipulation can be seen in the PAC framework as modifying the distribution D to $\dot{D}$. Then, the learner's expected cost for output h is $E_{x,y\sim\dot{D}}[l_h(x,y)]$.

The basic approach in all game based adversarial learning technique is to associate a cost function $c : X \times X \to \mathbb{R}$ that provides the cost $c(x, x')$ for the adversary in modifying the feature vector x to x'. The game involves the following two stages (we provide a generalization of the statistical classification game by Hardt et al. [93]):

1) The defender publishes its hypothesis $\hat{h}$. She has knowledge of c, and training samples drawn according to D.
2) The adversary publishes a modification function Δ.

The defender's loss is $E_{x,y\sim D}[l_h(\Delta(x), y)]$ and the adversary's cost is $E_{x,y\sim D}[l^a_h(\Delta(x), y)+c(x, \Delta(x))]$ where $l^a_h(x)$ is a loss function for the adversary that captures the loss when the prediction is $h(x)$ and the true output is y. It is worth pointing out that $\Delta : X \to X$ can depend on the hypothesis $\hat{h}$. This game follows the test data manipulation framework described earlier. The function Δ induces a change of the test distribution D to some other distribution $\dot{D}$[4] and the defender's loss function can be written as $E_{x,y\sim\dot{D}}[l_h(x,y)]$. Thus, just like the PAC framework, the costs for both players are stated in terms of the unknown D (or $\dot{D}$) and then the empirical counterparts of these functions are:

$$\frac{1}{n}\sum_{i=1}^{n} l_h(\Delta(x_i), y_i) \text{ and } \frac{1}{n}\sum_{i=1}^{n} l^a_h(\Delta(x_i), y_i) + c(x_i, \Delta(x_i))$$

The Stackelberg equilibrium computation problem is stated below. This problem is the analogue of the empirical risk minimization that the learner solves in the standard setting.

$$\min \sum_{i=1}^{n} l_h(\Delta(x_i), y_i)$$
$$\Delta \in \operatorname*{argmin}_{\Delta} \sum_{i=1}^{n} l^a_h(\Delta(x_i), y_i) + c(x_i, \Delta(x_i))$$

Unlike the standard empirical risk minimization, this problem is a bi-level optimization problem and is in general NP Hard.

[4]If (X, Y) is a random value distributed according to D, then the distribution of $(\Delta(X), Y)$ is $\dot{D}$.

Bruckner et al. [92] add a regularizer for the learner and the cost function $c(x, x')$ as the l_2 distance between x and x'. They solve the problem with a sequential quadratic approach.

Following the approach of Lowd et al. [73], Li et al. [94] use a cost function that is relevant for many security settings. The adversary is interested in classifying a malicious data point as non-malicious. Thus, the cost function only imposes costs for modifying the malicious data points.

B. 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* [95], 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:

$$Pr[A(T) \in S] \leq e^{\varepsilon} Pr[A(T') \in S] + \delta \tag{5}$$

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 bound defined by ε does not hold.

Training: The behavior of a ML system needs to be randomized in order to provide privacy guarantees. 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 [96]. In the scenario where users send reports 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 [97]. Another way to obtain randomized training data is to first learn an ensemble of teacher models on data partitions, and then use these models to make noisy predictions on public unlabeled data, which is used to train a private student model. This strategy was explored in [98], [99].

Chaudhuri et al. show that *objective perturbation*, i.e. introducing random noise—drawn from an exponential distribution and scaled using the model sensitivity[5]—in the cost function minimized during learning, can provide ε-differential privacy [100]. Bassily et al. provide improved algorithms and privacy analysis, along with references to many of the works intervening in private empirical risk minimization [101]

Our take-away VI.3. *Learning models with differential privacy guarantees is difficult because the sensitivity of models is unknown for most interesting ML techniques.*

Despite noise injected in parameters, Shokri et al. showed that large-capacity models like deep neural networks can be trained with multi-party computations from perturbed parameters and provide differential privacy guarantees [102]. Later, Abadi et al. introduced an alternative approach to improve the privacy guarantees provided: the strategy followed is to randomly perturb parameters during the stochastic gradient descent performed by the learning algorithm [103].

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 can be provided during inference. For instance, Dowlin et al. use homomorphic encryption [104] to encrypt the data in a form that allows a neural network to process it without decrypting it [105]. Although, this does not provide differential privacy, it protects the confidentiality of each individual input. 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.

C. Fairness and Accountability in Machine Learning

The opaque nature of ML generates concerns regarding a lack of fairness and accountability of decisions taken based on the model predictions. This is especially important in applications like credit decisions or healthcare [106].

Fairness: In the pipeline from Figure 1, *fairness* is relevant to the action taken in the physical domain based on the model prediction. It should not nurture discrimination against specific individuals [107]. Training data is perhaps the strongest 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 [108]. To learn fair models, Zemel et al. first learn an intermediate representation that encodes a sanitized variant of the data, as first discussed in [109]. 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 [110]. 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.

Accountability: *Accountability* explains model predictions using the ML model internals h_θ. This is fundamentally relevant to understanding model failures on adversarial examples. Few models are interpretable by design, i.e., match human reason-

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

ing [111]. Datta et al. introduced *quantitative input influence* measures to estimate the influence of specific inputs on the model output [112]. Another avenue to provide accountability is to compute inputs that the machine learning model's components are most sensitive to. An approach named *activation maximization* synthesizes an input that highly activates a specific neuron in a neural network [113]. The challenge lies in producing synthetic inputs easily interpreted by humans [114] but faithfully representing the model's behavior.

VII. No Free Lunch in Adversarial Learning

We begin by pointing out that if a classifier is perfect, i.e., predicts the right class for every possible input, then it cannot be manipulated. Thus, the presence of adversarial examples is a manifestation of the classifier being inaccurate on many inputs. Hence the dichotomy between robustness to adversarial examples and better prediction is a false one. Also, it is well-known in ML that, given enough data, more complex hypothesis classes (e.g., non-linear classifier as opposed to linear ones) provide better prediction (see Figure VII). As a result, we explore the interaction between prediction loss, adversarial manipulation at inference and complexity of hypothesis class.

Recall the theoretical characterization for data poisoning by Kearns and Li [39] (see Section IV). While poisoning attacks can be measured by the percentage of data modified, mathematically describing an attack at inference is non-obvious. Thus, our *first result* in this section is an identification of the characteristics of an *effective attack* at inference. Our *second result* reveals that, given a positive probability of presence of an adversary, any supervised ML algorithm suffers from performance degradation under an effective attack. Finally, our *third result* is that increased capacity is required for resilience to adversarial examples (and can also give more precision as a by-product). But to prevent empirical challenges, e.g., overfitting, more data is needed to accompany the increase in capacity. Yet, in most practical settings, one is given a dataset of fixed size, which creates a tension between resilience and precision. A trade-off between the two must be found by empirically searching for the optimal capacity to model the underlying distribution. Note that this result (presented below) is analogous to the no free lunch result in data privacy that captures the tension between utility and privacy [115], [116].

In the PAC learning model, data x, y is sampled from a distribution D. Recall that the learner learns $\hat{h}$ such that $P(|r(h^*) - r(\hat{h})| \leq \epsilon) \geq 1 - \delta$. For this section, we assume that there is enough data[6] so that ϵ, δ are negligible, hence we assume $\hat{h}$ is same as h^* for all practical purpose. Recall that the *performance* of any learning algorithm is characterized by the expected loss of its output h: $E_{x,y\sim D}[l_h(x, y)]$. In the real world, it is often not known whether an adversary will be present or not. Thus, we assume that an adversary is present with probability $q \in (0, 1)$; where q is not extremely close to 0 or 1 so that both q and $1 - q$ are not negligible. Also,

[6]This assumption is practical in many applications today. Moreover, insufficient data presents fundamental information theoretic limitations [22] on learning accuracy problems in the benign (without adversaries) setting itself.

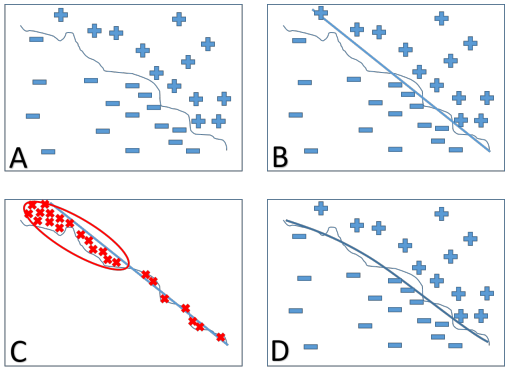

Fig. 5. Subfigure A shows the data available for learning and the true separator between the positive and negative region. The top left corner has few data points, which in the PAC model means that the data distribution D has low probability mass over that region. Subfigure B shows the model learned with an hypothesis class $\mathcal{H}$ of linear classifiers. Subfigure C shows all points misclassified by the linear model. Also shown is an adversarially chosen uniform distribution $\dot{D}$ restricted to the red oval in the top left corner; two observations are (1) the red crosses will cause a significant prediction loss with $\dot{D}$ and the linear model shown, and (2) the true separator in this red oval is highly non-linear (compared to rest of the space) and hence even the best linear classifier learned w.r.t. $\dot{D}$ will suffer significant expected loss. Subfigure D shows that a more complex non-linear classifier can be more accurate and can provide a lower expected loss against $\dot{D}$ (modulo over-fitting issues).

recall that an attack in the inference phase is captured by an adversarially modified distribution $\dot{D}$ of test data.

Our first result is the following definition that characterizes an effective attack by the adversary.

- **α-effective attack against $\mathcal{H}$ and D**: the best hypothesis $\dot{h}^*$ in the adversarial setting, i.e., $\dot{h}^* \in \text{argmin}_{h\in\mathcal{H}} E_{x,y\sim\dot{D}}[l_h(x, y)]$ still suffers a loss $E_{x,y\sim\dot{D}}[l_{\dot{h}^*}(x, y)]$ such that for the best hypothesis in the benign setting $h^* \in \text{argmin}_{h\in\mathcal{H}} E_{x,y\sim D}[l_h(x, y)]$

$$E_{x,y\sim\dot{D}}[l_{\dot{h}^*}(x, y)] = E_{x,y\sim D}[l_{h^*}(x, y)] + \alpha$$

This definition implies that for $\alpha > 0$ it is not trivial to defend against the modified distribution that the adversary presents compared to the benign scenario, since even the best hypothesis $\dot{h}^*$ against the modified distribution $\dot{D}$ suffers a greater loss than the best hypothesis h^* in the benign case.

Note that the definition above does not restrict the adversary to choose any particular $\dot{D}$, that is, the adversary is not restricted to a particular attack. The definition is parametrized by $\mathcal{H}$ and D, that is, the attack is effective against the given choice of hypothesis class $\mathcal{H}$ and data distribution D. We leave open the research question about whether a mathematical characterization of when such attacks exists or is feasible given the attackers cost–however, we illustrate in Figure VII-C why it is reasonable to assume that such attacks abound in practice.

For the sake of comparison, we contrast our definition with a possible alternate attack definition: one may call an attack effective if $E_{x,y\sim\dot{D}}[l_{h^*}(x, y)] = E_{x,y\sim D}[l_{h^*}(x, y)] + \alpha$, that is, the adversarial choice of $\dot{D}$ cause an increase in prediction loss against the deployed h^*. While such an attack is indeed

practical as shown in Figure VII-C, the impact of such an attack can be reduced if a different data distribution D is used or by gathering more data from the feature space (such as adding the adversarial samples back into training data). We present such an example in Appendix A. Our definition of α-effective attack leads to an impossibility of defense result that rules out any defense for any data distribution D and any defense measures based on gathering more data; further, later the same definition reveals the fundamental importance of the complexity of the hypothesis class used in adversarial settings.

No free lunch: In our result below, we reveal that higher the probability that an adversary is present leads to higher expected loss for any learner. In fact, we prove a stronger result: we show the above statement holds even if the learner is allowed to combine outputs in a randomized manner. Thus, we define the set of hypothesis that can be formed by choosing a set of hypothesis and randomly choosing one hypothesis: let $h_q(\mathcal{S}) = h$ such that $h \in \mathcal{S}$ and h is chosen from $\mathcal{S}$ according to distribution q, then $R(\mathcal{H}) = \cup_{\mathcal{S} \subseteq \mathcal{H}, \mathcal{S} \text{ finite}} \cup_q \{h_q(\mathcal{S})\}$.

Theorem 1. *Fix any hypothesis (function) class $\mathcal{H}$ and distribution D, and assume that an attacker exists with probability q. Given the attacker uses an α-effective attack against $\mathcal{H}$ and D with $\alpha \geq \alpha_0 > 0$, for all hypothesis $h \in R(\mathcal{H})$ the learner's loss is at least*

$$E_{x,y \sim D}[l_{h^*}(x, y)] + q\alpha_0$$

This theorem is proved in Appendix B. While the above result is a lower bound negative result, next, we present a positive upper bound result that shows that increasing the complexity of the hypothesis class considered by the learner can lead to better protection against adversarial manipulation. Towards that end, we begin by defining the lowest loss possible against a given distribution D: $l_D = \min_h E_{x,y \sim D}[l_h(x, y)]$.

- **β-rich hypothesis class:** A hypothesis class $\mathcal{H}'$ with the following properties: (1) $\mathcal{H} \subset \mathcal{H}'$ and (2) $E_{x,y \sim D}[l_{h'}(x, y)] \leq \min\{l_D, E_{x,y \sim D}[l_{h^*}(x, y)] - \beta\}$ for $h' \in \text{argmin}_{h \in \mathcal{H}'} E_{x,y \sim D}[l_h(x, y)]$, $h^* \in \text{argmin}_{h \in \mathcal{H}} E_{x,y \sim D}[l_h(x, y)]$ and all D.

Intuitively, $\mathcal{H}'$ is a more complex hypothesis class that provides lower minimum loss against any possible distribution.

Theorem 2. *Fix any hypothesis (function) class $\mathcal{H}$ and distribution D and a β-rich hypothesis class $\mathcal{H}'$. Assume the attacker is present with probability q and $l_D << E_{x,y \sim D}[l_{h^*}(x, y)] - \beta$ and $l_{\dot{D}} << E_{x,y \sim \dot{D}}[l_{\dot{h}^*}(x, y)] - \beta$. Given the attacker that uses an α-effective against $\mathcal{H}$ and D with $\alpha = \alpha_0$ and the learner uses the β-rich hypothesis class $\mathcal{H}'$, there exists a $h \in \mathcal{H}'$ such that the loss for h is less than*

$$E_{x,y \sim D}[l_{h^*}(x, y)] + q\alpha_0 - \beta$$

This theorem is proved in Appendix C. There are a number of subtle points in the above result that we elaborate below:

- The result is an upper bound result and hence requires bounding the attacker's capabilities by imposing a bound on its effectiveness α (compare with Theorem 1).
- The attack used is effective against the less complex class $\mathcal{H}$ whereas the defender uses the more complex class $\mathcal{H}'$. Following the lower bound in Theorem 1, if the attacker were to use an effective attack against the class $\mathcal{H}'$ then the defender cannot benefit from using the rich class $\mathcal{H}'$.

Standard techniques to increase the hypothesis complexity include: considering more features, using non-linear kernels in SVM and, using a neural network with more neurons. In addition, a well known general technique—ensemble methods—is to combine any classifiers to form complex hypothesis, in which combinations of classifiers are used as the final output.

Complexity is not free. The above results reveal that more complex models can defend against adversaries; however, an important clarification is necessary. We assumed the existence of enough data to learn the model with high fidelity; as long as it is the case, increasing complexity leads to lower bias and better accuracy. Otherwise, learning may lead to over-fitting or high variance in the model. Thus, while the above theoretical result and recent empirical work [94] suggests more complex models for defeating adversaries, in practice, availability of data may prohibit the use of this general result.

Our take-away VII.1. *Adversaries can exploit fundamental limitations of simple hypothesis classes in providing accurate predictions in sub-regions of the feature space. Such attacks can be thwarted by moving to a more complex (richer) hypothesis class, but over-fitting issues must be addressed with the more complex class.*

VIII. Conclusions

The security and privacy of machine learning is an active yet nascent area. We have explored the attack surface of systems built upon machine learning. That analysis yields a natural structure for reasoning about their threat models, and we have placed numerous works in this framework as organized around attacks and defenses. We formally showed that there is often a fundamental tension between security or privacy and precision of ML predictions in machine learning systems with finite capacity. In the large, the vast body of work from the diverse scientific communities jointly paint a picture that many vulnerabilities of machine learning and the countermeasures used to defend against them are as yet unknown–but a science for understanding them is slowly emerging.

Acknowledgments

We thank Martín Abadi, Z. Berkay Celik, Ian Goodfellow, Damien Octeau, and Kunal Talwar for feedback on early versions of this document. Nicolas Papernot is supported by a Google PhD Fellowship in Security. We also thank Megan McDaniel for taking good care of our diet before the deadline. 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

REFERENCES

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

[2] 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 (USENIX Security 16)*, 2016.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[9] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks and Machine Learning*, 2011, pp. 52–59.

[10] 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.

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

[12] 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.

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

[14] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[15] C. M. Bishop, "Pattern recognition," *Machine Learning*, 2006.

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

[17] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns," in *12th USENIX Security Symposium (USENIX Security 06)*, 2006.

[18] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *IEEE International Conference on Communications*, vol. 5, 2006, pp. 2388–2393.

[19] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.

[20] J. Cannady, "Next generation intrusion detection: Autonomous reinforcement learning of network attacks," in *Proceedings of the 23rd national information systems security conference*, 2000, pp. 1–12.

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

[22] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[23] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[24] A. Sinha, D. Kar, and M. Tambe, "Learning adversary behavior in security games: A pac model perspective," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 214–222.

[25] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 2006, pp. 16–25.

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

[27] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Networks*, vol. 22, no. 5, pp. 544–557, 2009.

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

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

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proceedings of the 2014 International Conference on Learning Representations*. Computational and Biological Learning Society, 2014.

[31] 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.

[32] P. Laskov *et al.*, "Practical evasion of a learning-based classifier: A case study," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 197–211.

[33] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical science*, pp. 235–249, 2002.

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

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

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

[37] 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1528–1540.

[38] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

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

[40] A. Globerson and S. Roweis, "Nightmare at test time: robust learning by feature deletion," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 353–360.

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

[42] B. Nelson and A. D. Joseph, "Bounding an attack's complexity for a simple learning model," in *Proc. of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML), Saint-Malo, France*, 2006.

[43] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[44] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise." in *ACML*, 2011, pp. 97–112.

[45] 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.

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

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*. Computational and Biological Learning Society, 2015.

[53] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *arXiv preprint arXiv:1511.04599*, 2015.

[54] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[55] 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.

[56] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

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

[58] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.

[59] 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 (USENIX Security 14)*, 2014, pp. 17–32.

[60] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," *arXiv preprint arXiv:1609.02943*, 2016.

[61] 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.

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

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

[67] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *In Computer Vision and Pattern Recognition (CVPR 2015)*. IEEE, 2015.

[68] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th USENIX Security Symposium (USENIX Security 16), Austin, TX*, 2016.

[69] L. Pinto, J. Davidson, and A. Gupta, "Supervision via competition: Robot adversaries for learning tasks," *arXiv preprint arXiv:1610.01685*, 2016.

[70] 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.

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

[72] Y. Vorobeychik and B. Li, "Optimal randomized classification in adversarial settings," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 485–492.

[73] 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.

[74] 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, no. May, pp. 1293–1332, 2012.

[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*. ACM, 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] N. Provos *et al.*, "A virtual honeypot framework." in *USENIX Security Symposium*, vol. 173, 2004, pp. 1–14.

[81] 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.

[82] C. Lyu, K. Huang, and H.-N. Liang, "A unified gradient regularization family for adversarial examples," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 301–309.

[83] I. Ororbia, G. Alexander, C. L. Giles, and D. Kifer, "Unifying adversarial training algorithms with flexible deep data gradient regularization," *arXiv preprint arXiv:1601.07213*, 2016.

[84] 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.

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

[86] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," *arXiv preprint arXiv:1607.04311*, 2016.

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

[88] 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.

[89] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[90] A. Sinha, T. H. Nguyen, D. Kar, M. Brown, M. Tambe, and A. X. Jiang, "From physical security to cybersecurity," *Journal of Cybersecurity*, p. tyv007, 2015.

[91] M. Tambe, *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[92] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 547–555.

[93] M. Hardt, N. Megiddo, C. Papadimitriou, and M. Wootters, "Strategic classification," in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ser. ITCS '16, 2016, pp. 111–122.

[94] B. Li and Y. Vorobeychik, "Feature cross-substitution in adversarial classification," in *Advances in Neural Information Processing Systems*, 2014, pp. 2087–2095.
[95] 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.
[96] 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.
[97] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 2014, pp. 1054–1067.
[98] J. Hamm, P. Cao, and M. Belkin, "Learning privately from multiparty data," *arXiv preprint arXiv:1602.03552*, 2016.
[99] 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.
[100] 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.
[101] R. Bassily, A. Smith, and A. Thakurta, "Differentially private empirical risk minimization: Efficient algorithms and tight error bounds," *arXiv preprint arXiv:1405.7085*, 2014.
[102] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1310–1321.
[103] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.
[104] 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.
[105] 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.
[106] 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.
[107] J. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores," *arXiv preprint arXiv:1609.05807*, 2016.
[108] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California Law Review*, vol. 104, 2016.
[109] 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.
[110] H. Edwards and A. Storkey, "Censoring representations with an adversary," *arXiv preprint arXiv:1511.05897*, 2015.
[111] 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.
[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] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, 2009.
[114] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, pp. 1–23, 2016.
[115] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer Berlin Heidelberg, 2008, pp. 1–19.
[116] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011, pp. 193–204.

APPENDIX

A. EXAMPLE OF DEFENSE USING ADDITIONAL DATA

In Figure 6 we show in sub-figure A the original data-set and the true classifier. The hypothesis class being used contains either a single linear separator or two linear separators. Thus, this hypothesis class can provide a classifier that is very close to the true classifier. However, for the data-set in A, the learned classifier is shown in sub-figure B, which is clearly far from optimal. This is not a problem of the hypothesis class; a different distribution of data shown in sub-figure C can provide for the learning of a much better classifier as shown. Another way is to add back adversarial examples as shown in sub-figure D (adversarial examples in red), which again makes the learned classifier much better.

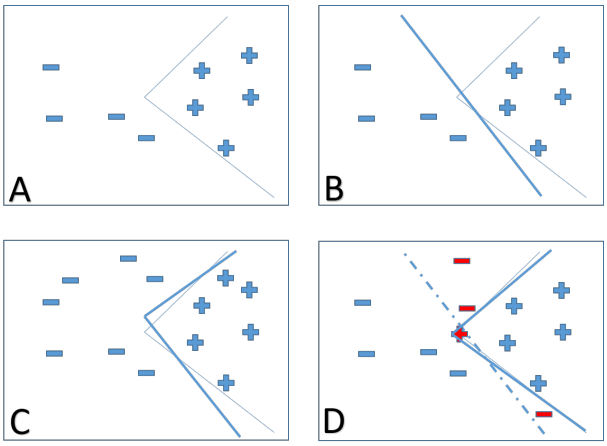

Fig. 6. Defense using additional data

B. PROOF OF THEOREM 1

Theorem 1. *Fix any hypothesis (function) class $\mathcal{H}$ and distribution D, and assume that an attacker exists with probability q. Given the attacker uses an α-effective attack against $\mathcal{H}$ and D with $\alpha \geq \alpha_0 > 0$, for all hypothesis $h \in R(\mathcal{H})$ the learner's loss is at least*

$$E_{x,y\sim D}[l_{h^*}(x,y)] + q\alpha_0$$

Proof. Choose any hypothesis $h \in R(\mathcal{H})$. The learner's loss is $E_{x,y\sim\dot{D}}[l_h(x,y)] = \int l_h(x,y)\dot{p}(x,y)dxdy$. For a randomized classifier h which randomizes over $h_1,\ldots,h_n$ with probabilities $q_1,\ldots,q_n$ respectively, the loss for any (x,y) is the expected loss $\sum_i q_i l_{h_i}(x,y)$. Thus, $E_{x,y\sim\dot{D}}[l_h(x,y)] = \sum_i q_i \int l_{h_i}(x,y)\dot{p}(x,y)dxdy = \sum_i q_i E_{x,y\sim\dot{D}}[l_{h_i}(x,y)]$. Now, from the definition of α-effective attack we have that $E_{x,y\sim\dot{D}}[l_{h_i}(x,y)] \geq E_{x,y\sim D}[l_{h^*}(x,y)] + \alpha_0$. Thus, $E_{x,y\sim\dot{D}}[l_h(x,y)] \geq E_{x,y\sim D}[l_{h^*}(x,y)] + \alpha_0$. Also, by definition $E_{x,y\sim D}[l_h(x,y)] \geq E_{x,y\sim D}[l_{h^*}(x,y)]$

Next, for any choice of h the adversary is present with probability q. Thus, the expected loss of any hypothesis is $qE_{x,y\sim\dot{D}}[l_h(x,y)] + (1-q)E_{x,y\sim D}[l_h(x,y)]$, which using the inequalities above is $\geq E_{x,y\sim D}[l_{h^*}(x,y)] + q\alpha_0$ □

C. Proof of Theorem 2

Theorem 2. *Fix any hypothesis (function) class $\mathcal{H}$ and distribution D and a β-rich hypothesis class $\mathcal{H}'$. Assume the attacker is present with probability q and $l_D << E_{x,y\sim D}[l_{h^*}(x,y)] - \beta$ and $l_{\dot{D}} << E_{x,y\sim \dot{D}}[l_{\dot{h}^*}(x,y)] - \beta$. Given the attacker that uses an α-effective against $\mathcal{H}$ and D with $\alpha = \alpha_0$ and the learner uses the β-rich hypothesis class $\mathcal{H}'$, there exists a $h \in \mathcal{H}'$ such that the loss for h is less than*

$$E_{x,y\sim D}[l_{h^*}(x,y)] + q\alpha_0 - \beta$$

Proof. Let the adversary's choice of distribution in his attack be $\dot{D}$. Choose the hypothesis $h \in H'$ such that $h' \in \operatorname{argmin}_{h\in\mathcal{H}'} E_{x,y\sim\dot{D}}[l_h(x,y)]$. The learner's loss is $E_{x,y\sim\dot{D}}[l_{h'}(x,y)]$. By definition of β-richness, we have

$$E_{x,y\sim\dot{D}}[l_{h'}(x,y)] \leq E_{x,y\sim\dot{D}}[l_{\dot{h}^*}(x,y)] - \beta$$

where $\dot{h}^* \in \operatorname{argmin}_{h\in\mathcal{H}} E_{x,y\sim\dot{D}}[l_h(x,y)]$. Also, by definition of α-effective attack against $\mathcal{H}$ we have

$$E_{x,y\sim\dot{D}}[l_{\dot{h}^*}(x,y)] = E_{x,y\sim D}[l_{h^*}(x,y)] + \alpha_0$$

for $h^* \in \operatorname{argmin}_{h\in\mathcal{H}} E_{x,y\sim D}[l_h(x,y)]$. Thus, we have

$$E_{x,y\sim\dot{D}}[l_{h'}(x,y)] \leq E_{x,y\sim D}[l_{h^*}(x,y)] + \alpha_0 - \beta$$

Also, by β-richness we have

$$E_{x,y\sim D}[l_{h'}(x,y)] \leq E_{x,y\sim D}[l_{h^*}(x,y)] - \beta$$

Next, the adversary is present with probability q. Thus, the expected loss of the hypothesis h' is $qE_{x,y\sim\dot{D}}[l_{h'}(x,y)] + (1-q)E_{x,y\sim D}[l_{h'}(x,y)]$, which using the inequalities above is $\leq E_{x,y\sim D}[l_{h^*}(x,y)] + q\alpha_0 - \beta$

□