Received March 7, 2020, accepted April 5, 2020, date of publication April 13, 2020, date of current version May 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2987435

Machine Learning Security: Threats, Countermeasures, and Evaluations

MINGFU XUE[1], (Member, IEEE), CHENGXIANG YUAN[1], HEYI WU[2], YUSHU ZHANG[1], (Member, IEEE), AND WEIQIANG LIU[3], (Senior Member, IEEE)

[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[2]Nanjing Upsec Network Security Technology Research Institute Co., Ltd., Nanjing 211100, China
[3]College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Corresponding authors: Mingfu Xue (mingfu.xue@nuaa.edu.cn) and Weiqiang Liu (liuweiqiang@nuaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602241, and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20150758.

ABSTRACT Machine learning has been pervasively used in a wide range of applications due to its technical breakthroughs in recent years. It has demonstrated significant success in dealing with various complex problems, and shows capabilities close to humans or even beyond humans. However, recent studies show that machine learning models are vulnerable to various attacks, which will compromise the security of the models themselves and the application systems. Moreover, such attacks are stealthy due to the unexplained nature of the deep learning models. In this survey, we systematically analyze the security issues of machine learning, focusing on existing attacks on machine learning systems, corresponding defenses or secure learning techniques, and security evaluation methods. Instead of focusing on one stage or one type of attack, this paper covers all the aspects of machine learning security from the training phase to the test phase. First, the machine learning model in the presence of adversaries is presented, and the reasons why machine learning can be attacked are analyzed. Then, the machine learning security-related issues are classified into five categories: training set poisoning; backdoors in the training set; adversarial example attacks; model theft; recovery of sensitive training data. The threat models, attack approaches, and defense techniques are analyzed systematically. To demonstrate that these threats are real concerns in the physical world, we also reviewed the attacks in real-world conditions. Several suggestions on security evaluations of machine learning systems are also provided. Last, future directions for machine learning security are also presented.

INDEX TERMS Artificial intelligence security, poisoning attacks, backdoor attacks, adversarial examples, privacy-preserving machine learning.

I. INTRODUCTION

Machine learning techniques have made major breakthroughs in recent years and have been widely used in many fields such as image classification, self-driving cars, natural language processing, speech recognition, and smart healthcare. In some applications, e.g., image classification, the accuracy of machine learning even exceeds that of humans. Machine learning has also been applied in some security detection scenarios, e.g., spam filtering, malicious program detection, which enables new security features and capabilities.

However, recent studies show that machine learning models themselves face many security threats: 1) *Training data poisoning* can result in a decrease in model accuracy or lead to other error-generic/error-specific attack purposes; 2) A well-designed *backdoor* in the training data can trigger dangerous consequences of a system; 3) A carefully-crafted disturbance in the test input (*adversarial examples*) can make the model go wrong; 4) *Model stealing attack*, *model inversion attack* and *membership inference attack* can steal the model parameters or recover the sensitive training data. All of the above security threats can lead to serious consequences to machine learning systems, especially in security and safety critical applications, such as autonomous driving, smart security, smart healthcare, etc.

In recent years, machine learning security has attracted widespread attentions [1], [2]. There are a large amount of research works on the security of deep learning algorithms

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Zhang.

since Szegedy *et al.* [1] highlighted the threat of adversarial examples in deep learning algorithms. However, machine learning security is not a new concept [3], and earlier works can be traced back to Dalvi *et al.* [4] in 2004. These earlier works, e.g., [4], [5], studied the so-called adversarial machine learning on non-deep machine learning algorithms in the context of spam detection, PDF malware detection, intrusion detection and so on [3]. Most of these earlier attacks are called *evasion attacks*, while a few others are referred as *poisoning attacks*.

Motivated by these issues, in the paper, we present a comprehensive survey on the security of machine learning. To date, only a few review and survey papers have been published on machine learning privacy and security issues. In 2010, Barreno *et al.* [6] review earlier evasion attacks on non-deep learning algorithms, and illustrated on a spam filter. Akhtar and Mian [7] review the adversarial example attacks on deep learning in the field of computer vision. They discuss adversarial example attacks and focus on computer vision. Yuan *et al.* [8] present a review on adversarial examples for deep learning, in which they summarize the adversarial example generation methods and discuss the countermeasures. Riazi and Koushanfar [9] analyze the provably secure privacy-preserving deep learning techniques. They discuss privacy protection techniques in machine learning and focus on cryptographic primitives-based privacy-preserving methods. The above review works all focus on only one type of attack, mostly adversarial examples attacks. Biggio and Roli [3] present a review on the wild patterns (also called adversarial examples) in adversarial machine learning over the last decade including the security of earlier non-deep machine learning algorithms and recent deep learning algorithms in the field of computer vision and cybersecurity. Particulary, evasion attacks and poisoning attacks are discussed, and corresponding defenses are presented [3]. Liu *et al.* [10] analyze security threats and defenses on machine learning. They focus on security assessment and data security. Papernot *et al.* [11] systematize the security and privacy issues in machine learning. Particularly, they describe the attacks with respect to three classic security attributes, i.e., confidentiality, integrity, and availability, while they discuss the defenses in terms of robustness, accountability and privacy [11].

The differences between this survey and these few existing review/survey papers are summarized as follows:

1) Instead of focusing on one stage, one type of attack, or one specific defense method, this paper systematically covers all the aspects of machine learning security. From the training phase to the test phase, all types of attacks and defenses are reviewed in a systematic way.
2) The machine learning model in the presence of adversaries is presented, and the reasons why machine learning can be attacked are analyzed.
3) The threats and attack models are described. Furthermore, the machine learning security issues are classified into five categories covering all the security threats of machine learning, according to the life cycle of a machine learning system, i.e., training phase and test phase. Specifically, five types of attacks are reviewed and analyzed: 1) *data poisoning*; 2) *backdoor*; 3) *adversarial examples*; 4) *model stealing attack*; 5) *recovery of sensitive training data*, which includes *model inversion attack* and *membership inference attack*.
4) The defense techniques according to the life cycle of a machine learning system are reviewed and analyzed. Moreover, the challenges of current defense approaches are also analyzed.
5) Several suggestions on security evaluations of machine learning algorithms are provided, including design-for-security, evaluating using a set of strong attacks, and evaluation metrics.
6) Future research directions on machine learning security are presented, including: attacks under real physical conditions; privacy-preserve machine learning techniques; intellectual property (IP) protection of DNN; remote or lightweight machine learning security techniques; systematic machine learning security evaluation method; the underlying reasons behind these attacks and defenses on machine learning.

The rest of this paper is organized as follows. The machine learning model in the presence of adversaries, and the reasons why machine learning can be attacked, are described in Section II. The threat models and attack approaches are reviewed in Section III. The defense techniques and challenges are analyzed in Section IV. The security evaluations of machine learning algorithms are discussed in Section V. Future directions on machine learning security, are presented in Section VI. We conclude this paper in Section VII.

II. MACHINE LEARNING MODEL IN THE PRESENCE OF ADVERSARIES

A. OVERVIEW OF MACHINE LEARNING

The overview of a machine learning system is shown in Fig. 1. We describe the machine learning systems from the following aspects.

Stages: Generally, a machine learning system can be separated into two stages: 1) *training phase*, where an algorithm learns from the training data to form a model with model parameters; 2) and *test phase*, where the trained model is applied to a specific task, such as classification, to give a predicted label for the input data.

Algorithm: an algorithm is used to learn from the training set to obtain a model with parameters. We divide the machine learning algorithms into two categories, *NN algorithms* and *non-NN algorithms*. We use the term *NN algorithms* to represent the Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and other Neural Network (NN) algorithms which make major breakthroughs in recent years and have significantly improved the performance of machine learning systems. On the other hand, we use the term *non-NN algorithms* to represent the other

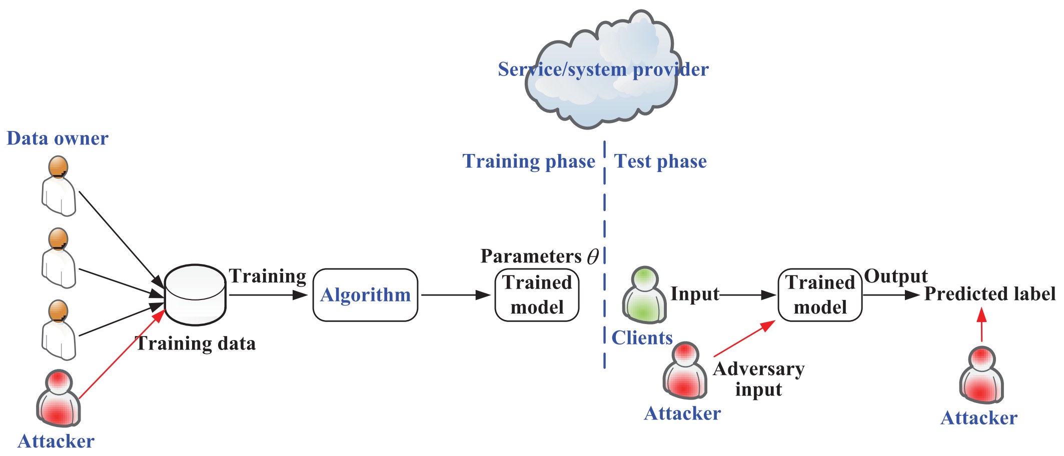


FIGURE 1. Overview of machine learning systems, which illustrates the two phases, the learning algorithm, and different entities.

traditional machine learning algorithms, such as Support Vector Machine (SVM), *k*-means, Naive Bayes, etc.

Entities in Adversarial Models: A normal machine learning system consists of the following entities, *data owner*, *system/service provider*, and *clients*, while in the adversarial model, there are also *attackers*, as shown in Fig. 1. The *data owners* are the owners of the massive training data which are usually private. The *system/service provider* is the provider who constructs the algorithm, trains the model and then performs the task or provides the service. The *clients* are the users who use the service, e.g., through the provided prediction APIs. The *attacker* can be an external adversary, or an curious person inside the system who is interested in the secret information of other entities.

B. WHY MACHINE LEARNING CAN BE ATTACKED

First, the working paradigm of machine learning makes it be vulnerable to various types of attacks. In the training phase, the massive training data and computational complexity of the training process in deep learning network lead to: 1) the training procedure is outsourced [12]; 2) pre-trained models from third parties which are served as intellectual properties (IPs), are integrated into the network; 3) a large amount of data comes from untrusted users or third parties without undergoing effective data validations. However, the above working paradigms also bring new security threats.

The so called machine-learning-as-a-service has also been increasingly used, in which the machine learning model works on a server or on the cloud, while clients can query the model through prediction APIs. The vast amounts of data used to train this model are often sensitive, and the parameters of the model are of great commercial value and are therefore confidential. These are the targets of an attacker in the test phase.

Second, reasons behind these attacks remain open problems. There are a few discussions about the reasons behind these successful attacks on machine learning models, however, they still lack consensus. Goodfellow *et al.* [2] indicate that the linearity of DNN model in high dimensional space is a possible reason for being vulnerable to adversarial example attacks. Studies [1], [8] also suggest that training data incompletion is one of the reasons for the existence of adversarial examples. They conclude that the adversarial examples are corner cases with low probability in the test set, which indicates that the training data is not enough and is not complete. Yeom *et al.* [13] study the effect of *overfitting* and *influence* on recovering sensitive training data or attribute by an adversary. They show that overfitting plays an important role to make the attacker be able to carry out membership inference attacks. Due to the unexplained nature of machine learning models, the essential reasons for these attacks, e.g., is the adversarial example a bug or an intrinsic property of the model, why sensitive training data can be recovered through normal quires, are still open problems.

III. ATTACKS ON MACHINE LEARNING

In this section, we will review the threats and attacks faced by machine learning systems. As shown in Fig. 2, to date, all the security threats along the life cycle of machine learning systems can be divided into five categories: 1) *Training set poisoning*; 2) *Backdoor in the training set*; 3) *Adversarial example attacks*; 4) *Model theft*; 5) *Recovery of sensitive training data* (including *model inversion attack* and *membership inference attack*). The first two attacks occur during the training phase, while the last three attacks occur during the

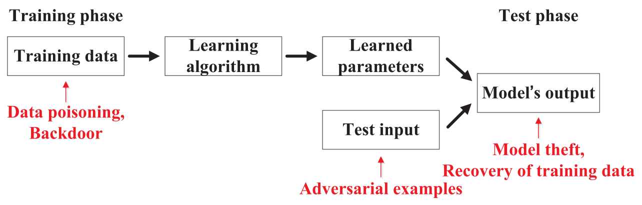


FIGURE 2. Attacks on machine learning systems.

test phase. We will review these five attacks in the following sections respectively, and present discussions about attacks in Section III-F.

A. TRAINING SET POISONING

The malicious manipulation of training set aiming at misleading the prediction of a machine learning model, is called *poisoning attack*. Studies have shown that a small percentage of carefully constructed poisoning training data can make a dramatic decrease in the performance of the machine learning model. The overview of poisoning attacks is shown in Fig 3. In this paper, we divide the poisoning works in terms of whether it is targeting a neural network (NN) model. The summary on training set poisoning approaches is presented in Table 1 in terms of the type, the targets, the working mechanism, the effect, the advantages and disadvantages of the investigated methods.

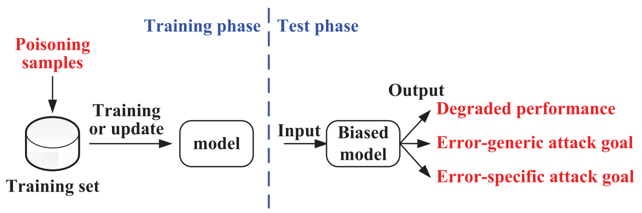


FIGURE 3. Overview of poisoning attacks.

1) POISONING ATTACKS TARGETING NON-NN MODELS

Targeting Anomaly Detection or Security Detection Applications: Machine learning has been widely used in many security detection applications, such as abnormal detection and malware detection. These are clearly the valuable targets for poisoning attacks. Rubinstein *et al.* [14] propose three poisoning attacks on Principal Component Analysis (PCA)-subspace method based anomalies detection system in backbone networks. It is shown that by introducing a small fraction of poisoned data, the performance of the detector decreases dramatically. This method is simple and effective [14], however, it focus on binary classification problems, which is not generic to other learning algorithms. Li *et al.* [15] use the Edge Pattern Detection (EPD) algorithm to design poisoning attack, named chronic poisoning attack, on machine learning based intrusion detection systems (IDS). The method can poison several learning algorithms including SVM, LR and NB [15]. However, the method in [15] uses a long-term slow poisoning procedure and is complicated to implement.

Targeting Biometric Recognition Systems: Machine learning techniques are also applied in adaptive biometric recognition systems so as to adapt the changes of the users' biometric traits, e.g., aging effects. However, the updating process can be exploited by an attacker to compromise the security of the system [16]. Biggio *et al.* [16] propose a poisoning attack targeting a PCA-based face recognition system. By submitting a set of carefully designed fake faces (i.e., poisoned samples) and claiming to be the victim, the system template will be gradually compromised due to the adaptive updating process. At last, the attacker can impersonate the victim with his own face. In [16], it is assumed that each user only stores one template in the system and the attacker has complete knowledge of the system, such as the feature extraction algorithm, the matching algorithm, the template update algorithm, and even the victim's template, which are difficult to obtain in practice. In [17], Biggio *et al.* improve the above attack targeting a more realistic face recognition system, where the system stores multiple templates per user, using different matching algorithms, and the attacker only has an estimate of the victim's face image. However, it is demonstrated that the attack success rate depends on the attacker-victim pair [17].

Targeting SVM: Biggio *et al.* [18] propose poisoning attacks against Support Vector Machines (SVM), where crafted training data is injected to increase the test errors of the SVM classifier. They use a gradient ascent strategy based on the SVM's optimal solution to construct the poisoning data. This method construct poisoning data use optimization formulation and can be kernelized [18], but it needs the full knowledge of the algorithm and the training data.

Targeting Clustering Algorithms: Clustering algorithms have been widely used in data analysis and security applications, e.g., market segmentation, web pages classification, malware detection [19]. However, the clustering process itself can be subverted by a smart attacker. Biggio *et al.* [19] demonstrate that an attacker can poison the clustering process by introducing a small amount of poisoning samples to the training data. Moreover, these poisoning samples can be effectively obfuscated and hidden in the input data. This approach is evaluated on malware samples clustering and handwritten digits clustering. A similar poisoning approach is proposed by Biggio *et al.* [20] which targets behavioral malware clustering by adding crafted poisoning samples with poisoning behaviors to training data [20]. In general, these methods [19], [20] first calculate the distance between two clusters, and then insert poisoning data between two clusters, which can confuse the boundaries between two clusters. As a result, clustering algorithms will incorrectly merge two different clusters into a single cluster [20]. These methods [19], [20] are generic and can attack most of the clustering algorithms. However, these methods require the attacker to know the target clustering algorithm, the training data, and the feature space, etc.

Targeting Algorithms or Processing Methods: Poisoning attacks can also be used to attack specific machine learning algorithms or processing methods. Xiao *et al.* [21] investigate the poisoning attacks on several feature selection methods, e.g., ridge regression, the elastic net, and least absolute shrinkage and selection operator (LASSO). They demonstrate on PDF malware detection and show that poisoning attacks can compromise these feature selection methods significantly [21]. Li *et al.* [22] propose a poisoning attack on the collaborative filtering system. They demonstrate that an attacker with full knowledge of the system can construct

TABLE 1. Summary on training set poisoning approaches.

Targeting NN or non-NN	Target system/model	Approaches	Working mechanism	Effect	Advantages	Disadvantages
Targeting non-NN models	Anomaly detection or security detection applications	Poisoning PCA-subspace based anomalies detection [14]; Chronic poisoning against IDS [15]	Poison the training data of the model	Reduce the performance of anomaly detection algorithms	Simple and effective [14]; Effective on several learning models [15]	Not generic [14]; Long-term slow poisoning and complicated to implement [15]
	Biometric recognition systems	Poisoning PCA-based face recognition systems [16]; Poisoning face templates [17]	Submitting a set of fake faces and claiming to be the victim	Replace the victim's template in the system	Stealthily and gradually	Require the knowledge of the target system and/or the victim
	SVM	Poisoning SVM [18]	Use a gradient ascent strategy based on the SVM's optimal solution	Increase the test errors of the SVM classifier	Use optimization formulation and can be kernelized	Need the full knowledge of the algorithm and the training data
	Clustering algorithms	Poisoning data clustering [19]; Poisoning behavioral malware clustering [20]	Poisoning attack based on the distance between clusters	Mislead clustering algorithms	Generic to clustering algorithms	Require the knowledge of the target algorithm
	Machine learning algorithms or processing methods	Poisoning feature selection [21]; Poisoning collaborative filtering [22]; Poisoning regression learning [23]	Optimization-based or statistical-based poisoning attacks	Mislead feature selection algorithms, collaborative filtering system or linear regression	Powerful and effective	Not generic, and/or require the knowledge of the target algorithm
	Online learning or data sanitization defenses	Poisoning breaks data sanitization defenses [24]	Optimization-based poisoning attacks	Bypass data sanitization defenses	Strong, effective under defenses	High computational overhead
		Online data poisoning [25] [26]	Optimization-based poisoning attacks	Reduce the performance of online learning	Can work under online learning scenarios	High computational overhead
Targeting NN models	\	Generative poisoning against NN [27]	GAN-based poisoning attack	Reduce the accuracy of the NN model	Generate poisoning data fast	Require interactions with the model frequently
		Poisoning of deep learning with back-gradient optimization [28]	Optimization-based poisoning attack	Increase multi-class classification error	Extend to multi-class problems and generalize well	High computational overhead
Specific application-oriented	\	Poisoning healthcare [29]; Poisoning graph-based recommender systems [30]; Poisoning crowd sensing systems [31]	Optimization-based poisoning attacks	Mislead healthcare system, recommender system or crowd sensing systems	Apply poisoning attacks to practical application scenarios	Require the knowledge of the target system

poisoning data to achieve the goal and imitate normal users' behaviors to avoid being noticed. In [21], [22], they formalize the poisoning attack as an optimization problem and solve it use the gradient ascent strategy. Jagielski *et al.* [23] investigate poisoning attacks on linear regression, and use statistical properties of the data to generate poisoning data. These methods [21]–[23] are not generic, and/or require the attacker to have the knowledge of the target algorithm, such as the feature set, the learning algorithm, etc.

Strong Data Poisoning and Online Learning Poisoning: The evolution of poisoning attacks and corresponding defense techniques is also a spiraling process. Koh *et al.* [24] propose a strong poisoning attack which can bypass existing data sanitization defenses. Specifically, they placed the poisoning samples near one another to evade the anomaly detection, and set some constraints to the optimization problem to evade detection.

Most of prior works of poisoning attack are in the offline learning situation. Wang and Chaudhuri [25] propose a poisoning attack targeting online learning situation, in which the training data is inputted in the form of stream. They formulate their attack as an optimization problem, and solve it using the gradient ascent strategy. Besides, this method only modifies the training data at specific positions in an input stream to reduce the search space thus improves the attack efficiency [25]. Similarly, Zhang and Zhu [26] propose an improved online poisoning attack which requires limited knowledge of the training process. They formulate such attack as stochastic optimal control, and propose two attack algorithms, model predictive control (MPC)-based attack, and reinforcement learning-based attack.

The strong poisoning attack [24] can be effective under several data sanitization methods. The poisoning attack methods in [25], [26] can succeed in online learning scenarios. However, these methods [24]–[26] generate poisoning data by solving optimization problems, which require expensive computational overhead.

2) POISONING ATTACKS TARGETING NN MODELS

In recent years, the popularity of neural networks has also led to poisoning attacks against neural network models. Yang *et al.* [27] use Generative Adversarial Networks (GAN) to perform poisoning attacks. Specifically, an auto-encoder is used as the generator to generate poisoning data, and the target model is used as a discriminator to calculate the effect of poisoning data on the model. This method accelerates the poisoning data generation thus can find out the effective poisoning data quickly, but this method needs to interact with the target model frequently.

Muñoz-González *et al.* [28] propose poisoning attacks on deep learning by using back-gradient optimization. They extend the poisoning attacks from binary algorithms to multiclass problems. The gradient is computed through automatic differentiation. They also reverse the learning process to reduce the complexity of the attack [28]. Moreover, it is shown that, similar to adversarial examples in the test phase, the poisoning examples in the training phase can also generalize well across different learning models [28]. However, this method optimizes one poisoning data each time which requires high computational overhead [23].

3) POISONING ATTACKS IN SPECIFIC APPLICATION SCENARIOS

Some specific valuable application scenarios have become the targets of poisoning attacks. Mozaffari-Kermani *et al.* [29] propose poisoning attacks targeting healthcare datasets. Both generic errors and specific errors can be achieved, which are catastrophic in healthcare systems. Fang *et al.* [30] propose poisoning attacks on recommender systems. They generate fake users with crafted rating scores based on an optimization problem, and inject them to the recommender system. In this way, the attacker can make a target instance be recommended to as many people as possible. Miao *et al.* [31] propose a poisoning attack framework towards crowd sensing systems. The malicious workers can be disguised as normal ones to evade detection, while achieving the maximum attack utility meanwhile. In these works, the poisoning attacks are applied to specific scenarios, such as healthcare systems [29], recommender systems [30] and crowd sensing systems [31]. These works formalize poisoning attacks into different optimization problems to design corresponding poisoning attack strategies. However, these methods [29]–[31] require the attackers to know the learning algorithm used by the target system or the information about the training data.

B. BACKDOOR IN THE TRAINING SET

Recently, researchers show that an attacker can create a backdoor hidden in the training data or in the pre-trained model. The overview of backdoor attacks is shown in Fig 4. The backdoor does not affect the normal function of the model, but once a specific trigger condition arrives, the backdoor instance will be misclassified by the model as the target label specified by the attack. Such backdoor attack is stealthy because of the unexplained nature of the deep learning models. The summary of backdoor attack works is shown in Table 2.

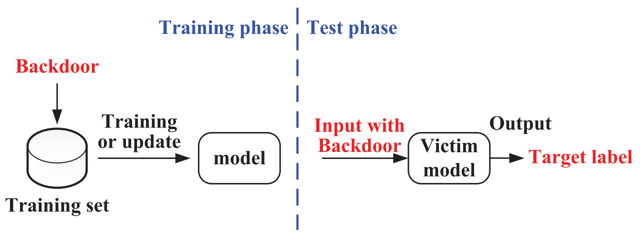


FIGURE 4. Overview of backdoor attacks.

1) BACKDOOR ATTACKS

Gu *et al.* [12] propose a maliciously trained network, named BadNet. BadNet can cause bad behaviors of the model when a specific input arrives. They demonstrate the effectiveness of BadNet on handwritten digit classifier and road sign classifier. Ji *et al.* [32] study backdoors on learning systems. The backdoors are introduced by primitive learning modules (PLMs) supplied from third parties. The malicious PLMs which are integrated into the machine learning system can cause the system malfunction once a predefined trigger condition is satisfied. They demonstrate such attack on a skin cancer screening system while the attacker doesn't require the knowledge about the system and the training process [32]. However, in [32], the attacker directly manipulates the parameters of the model to insert backdoors. This assumption is difficult to satisfy in practice.

Chen *et al.* [33] propose a backdoor attack on deep learning models by using data poisoning. Specifically, poisoning samples are injected into the training dataset so as to implant a backdoor. Their attack can work under a weak attack model, which means it doesn't require knowledge about the model and the training set [33]. Only 50 poisoning samples are injected while the attack success rate is above 90% [33]. Liao *et al.* [34] propose backdoor attacks in CNN models by injecting stealthy perturbations. Specific embedded pattern will be recognized as a target label defined by the attacker.

TABLE 2. The summary of backdoor attack works.

Scenario	Approaches	Working mechanism	Effect	Advantages	Disadvantages
Backdoor attacks	BadNet [12]; PLMs [32]; Attack on deep learning [33]; Attack on CNNs [34] [35]; Attack on federated learning [36]	Add well-designed backdoor signals to the clean data; Manipulate the model's parameters directly	Target model misclassifies backdoor instances	Stealthy and don't affect the performance of the model	Need to participate in the training process, or re-train the model
Trojan attacks	PoTrojan [37]; Trojan attack on NN [38]	Add Trojan to the inner structure of NN [37]; Re-train the model to embed Trojan [38];	Target model produces malicious output	Without re-training [37]; Inverse the NN to generate the trigger [38];	Require the full knowledge of the model [37]; Requires re-training [38]

Barni *et al.* [35] propose backdoor attacks on CNNs, in which they corrupt samples of a target class without label poisoning. They evaluated the attack on MNIST digits classifier and traffic signs classifier.

Backdoor attacks can even attack the state-of-the-art secure training models and training processes. Bagdasaryan *et al.* [36] illustrate backdoor attacks on Federated Learning, which is considered to be a secure privacy-preserving learning framework. They demonstrate that a malicious participant can introduce stealthy backdoor function into the global model by using model replacement.

In these methods [12], [33]–[36], the attacker first adds the well-designed backdoor signal to the clean data to generate poisoning data, and then injects the poisoning data into the training set to re-train the target model. After re-training, a specific backdoor is embedded into the target model. These methods [12], [33]–[36] are stealthy and can perform backdoor attacks without affecting the performance of the model, e.g., the accuracy drop on normal inputs is less than 1% in [34]. However, these methods need to participate in the training process of the model, or re-train the model.

2) TROJAN ATTACKS IN MACHINE LEARNING

In some literatures, such backdoors are also called *Trojans*. Liu *et al.* [39] propose a preliminary concept of neural Trojans in neural network IPs, which are malicious functions inserted in the neural networks. They propose three neural Trojan mitigation techniques, input preprocessing, re-training, and anomaly detection (see Section IV-B for details), without given the detailed description of the neural Trojan implementations [39]. Zou *et al.* [37] propose neural-level Trojans in pre-trained NN models, named PoTrojan. PoTrojan remains inactive and will only be trigged with rare conditions, which can make the model produce malicious output. Specifically, they design two kinds of triggers (single-neural PoTrojans and multiple-neural PoTrojans), and design two kinds of payloads based on whether an attacker can access to the training data of the target label [37]. Liu *et al.* [38] insert Trojans to NN by a two-step implementation. First, they inverse the NN to generate the Trojan trigger. Then, they re-train the NN model to inject malicious payloads to the NN. In these methods [37], [38], the attackers need to have the full knowledge of the target neural network, which is difficult to obtain in practice.

C. ADVERSARIAL EXAMPLE ATTACKS

Adversarial example is a disturbance to the input data carefully constructed by an attacker to cause the machine learning model to make a mistake. The term ''adversarial example'' is introduced by Szegedy *et al.* [1] in 2014 targeting deep learning algorithms. However, the similar concept and methods are far more ancient, which are called adversarial machine learning targeting non-deep machine learning algorithms [3], [4]. In these earlier works, these attacks are referred as *evasion attacks* mainly targeting at spam filtering, malware detection, intrusion detection, and so on.

The adversarial example attacks can be further divided into two categories [3]: *error-generic attack*, which just makes the model go wrong; and *error-specific attack*, which aims at making the model incorrectly identify the adversarial example as coming from a specific class. The summary of works on adversarial example attacks is presented in Table 3.

1) EARLIER EVASION ATTACKS ON NON-DEEP LEARNING ALGORITHMS

Dalvi *et al.* [4] first highlighted the adversarial classification problem in which an adversary tries to make the classifier produce wrong predictions. Moreover, they formulate the classification as a game between the classifier and the adversary, and present an optimal strategy. Lowd and Meek [40] introduce the adversarial learning problem, in which the adversary tries to reverse engineering the classifier through sending a number of queries. In this way, the adversary can find the ''malicious'' instances that the classifier cannot recognize. Nelson *et al.* [5] use statistical machine learning to attack spam filter to make it useless or achieve focused targets. Barreno *et al.* [6] review earlier attacks on machine learning systems, and provide a formal method to describe the interaction between the attacker and the defender. They illustrate their taxonomy on a spam filter, SpamBayes.

Biggio *et al.* [41] convert the evasion attack into an optimization problem, and solve it using the gradient-descent based approach. They evaluate the security of the model by increasing the adversary's knowledge and capabilities. They demonstrated that malicious examples generated by the gradient-based method can evade the PDF malware detection. Šrndić and Laskov [42] experimentally explore the performance of classifiers (PDFRATE) under evasion attacks and reveal that PDFRATE's classification performance drops significantly even under simple attacks. Demontis *et al.* [43] implement several evasion attacks on a linear classifier, the Android malware detection tool (Drebin), according to the attackers with different knowledge and abilities.

In these evasion attacks [4], [5], [40]–[43], attackers modify the features that have the greatest impact on the detector based on the feedback from the detector [56]. The malicious instances generated by these methods can evade detection of security related applications. However, these evasion attacks require the attackers to know the features extracted by the target algorithm or the feature extraction algorithm of the target system [8], which are difficult to obtain in practice. Besides, the attack success rate depends on how much knowledge the attacker has.

Compared with the above methods that require the attacker to have the knowledge of the target system, Xu *et al.* [44] and Dang *et al.* [45] propose evasion attacks without the knowledge of the target system. According to the detection score returned by the malware detector, Xu *et al.* [44] stochastically modify the malware to find a malicious example that can evade detection but retain malicious behavior. Similarly, Dang *et al.* [45] propose an evasion attack, named EvadeHC. EvadeHC first randomly modifies the malware to generate

TABLE 3. Summary of works on adversarial example attacks.

Scenario	Type/Targets	Approaches	Working mechanism	Effect	Advantages	Disadvantages
Earlier evasion attacks on non-NN models	Game theory	Adversarial classification [4]	Modify features that have the greatest impact on the detector	Malicious instances cannot be detected by the detector	Evade detection of security related applications	Require knowing the feature extraction algorithm
	Through reverse engineering the classifier	Adversarial learning [40]				
	Spam filtering	Nelson et al. [5]				
	PDF malware detection	Gradient-based evasion attack [41]; Šrndić and Laskov [42];				
	Android malware detection	Attack a linear classifier [43]				
	Malware detection (black-box)	Stochastic search-based attack [44]; EvadeHC [45]	Stochastic search-based; Heuristic search-based	Malicious instances cannot be detected by the detector	Generic; do not require knowledge about systems	Require iterations and feedback from the system
Adversarial example attacks on NN models	First work of adversarial examples on NN	Intriguing properties of NN [1]	Gradient-based or optimization-based adversarial example attacks	Adversarial examples mislead the model and cannot be perceived by humans	Perturbations are small and impercepti-ble; do not require knowing the feature extraction algorithm of the system	Attacks may fail in the real world
	One-step/one-shot methods	FGSM [2]				
	Take multiple small steps iteratively	BIM [46]				
	Perturbing only a few pixels	JSMA [47]; One pixel attack [48]				
	High-confidence adversarial examples	Carlini and Wagner [49], [50]				
	Universal adversarial perturbations	Universal perturbations [51]				
	Targeting deep learning based security detection	Malware classification [52]				
	In biometric authentication systems	Face recognition [53]				
In real world conditions	\	Road sign recognition attack [54]	Simulate physical conditions in adversarial examples generation process	Attack successfully in physical conditions	Robust to physical conditions	Perturbations are large and conspicuous
	\	Cellphone camera attack [46]				
	\	Face recognition system attack [53]				
	\	3D objects attack [55]				

a malware collection. Then, according to the binary results (i.e., reject or accept) returned by the detector, EvadeHC uses the hill-climbing algorithm to search for malware that can evade the detection from the malware collection. These methods [44], [45] are generic to binary classification problems, but these methods need multiple iterations to obtain effective adversarial examples. Hence, the computational overhead of these methods is high. Moreover, these attacks need the feedback of the detector which limits the practicality.

2) ADVERSARIAL EXAMPLE ATTACKS ON DEEP LEARNING

The pipeline of adversarial example attacks on deep learning algorithms is shown in Fig. 5. Szegedy *et al.* [1] first discovered that DNNs are surprisingly susceptible to adversarial examples which are small imperceptible perturbations to images. Such adversarial examples attacks can make the DNN model misclassification.

The earlier adversarial attack methods in the literature are called one-step/one-shot methods, e.g., the Fast Gradient Sign Method (FGSM) proposed by Goodfellow *et al.* [2]. In the latter improvement works, instead of taking a single large step to increase the loss of a classifier, the researchers tried to take multiple small steps iteratively so as to adjust the direction after each step [7], e.g., the Basic Iterative Method (BIM) [46].

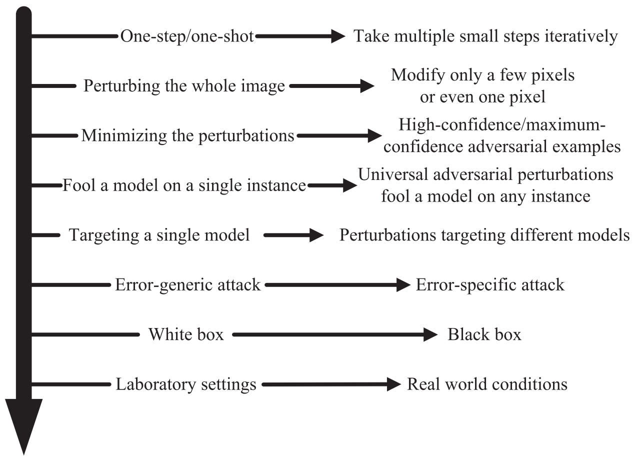

One-step/one-shot
Take multiple small steps iteratively
Perturbing the whole image
Modify only a few pixels or even one pixel
Minimizing the perturbations
High-confidence/maximum-confidence adversarial examples
Fool a model on a single instance
Universal adversarial perturbations fool a model on any instance
Targeting a single model
Perturbations targeting different models
Error-generic attack
Error-specific attack
White box
Black box
Laboratory settings
Real world conditions

FIGURE 5. Pipeline of adversarial example attacks in the context of deep learning algorithms.

Further adversarial example works try to perturbing only a few pixels in an image rather than the whole image [7], e.g., the Jacobian-based Saliency Map Attack (JSMA) [47]. Papernot *et al.* [47] propose an adversarial example generation method in which the adversary only needs to know the structure of the model. As an extreme case, Su *et al.* [48]

propose a method that fools the classifier by modifying just one pixel of an image.

Note that, although the initial adversarial examples are minimal perturbations, in some scenarios, it is more reasonable to assume that an adversary wants to maximize the classifier's confidence on the wrong predictions rather than only minimizing the perturbations [3]. The reason is that, the initial works on adversarial examples aim at analyzing the sensitivity of deep learning algorithms to minimal perturbations, however, in order to analyze the security of a deep learning algorithm under attacks, it is more reasonable to use the maximum-confidence adversarial attacks which can reflect the security of an algorithm under more powerful attacks [3]. For example, Carlini and Wagner [49], [50] show that several recent defense techniques against minimal-perturbations attacks can be bypassed by high-confidence adversarial examples. As a result, they suggest to use high-confidence adversarial examples for security evaluations [49], [50].

After the methods which introduce perturbations to fool a network on a single image each time (e.g., FGSM [2], DeepFool [57]), Moosavi-Dezfooli *et al.* [51] propose universal adversarial perturbations which can fool a network on any image.

Whereas earlier works are computing perturbations targeting a single model, more powerful attacks can generate perturbations generalize well across different models [7]. Studies have shown that there is a transferability between different models, especially models with similar structures.

The other three research directions about adversarial examples are from error-generic attack to error-specific attack, from white-box scenario to black-box scenario, and from laboratory conditions to real world conditions. For example, Papernot *et al.* [58] propose a black-box adversarial attack strategy, in which the attacker observes the outputs of the DNN according to chosen inputs, and then build a substitute model of the target DNN model. They use the substitute model to generate adversarial examples which are found to be effective in the target model as well [58].

As special cases, we discuss two specific applications. On the one hand, deep learning has been applied in some security applications. However, adversarial examples have been used to compromise these deep learning based security applications. For example, Grosse *et al.* [52] propose adversarial example attacks against DNN based malware classification. On the other hand, deep learning has also been successfully applied in biometric authentication systems, e.g., face recognition, voice control systems. However, by generating carefully crafted adversarial examples, an attacker can make the model incorrectly identified an attacker as a legal user thereby achieving the legal user's privilege (error-specific attack), or can make an attacker evades the system identification (error-generic attack) [53].

In these works [1], [2], [46]–[53], the adversarial examples are mainly generated based on the gradient or the optimization. The gradient-based approaches calculate the gradient of the loss function, and add perturbation to the input data according to the gradient to generate adversarial examples. The optimization-based approaches convert the adversarial examples generation problem into optimization problems. The goal of the optimization is that the perturbations in adversarial examples can not only mislead the model, but also not be perceived by humans. Compared with earlier evasion attacks on non-deep machine learning algorithms, these adversarial examples attacks on deep learning algorithms do not require the attackers to know the feature extraction algorithm used by the target system. In addition, the perturbations in the generated adversarial examples are small and imperceptible, and can achieve a high attack success rate. However, due to the influence of many physical factors (e.g., angle, distance, etc.), these digital adversarial examples have failed or have low attack success rates in the real world conditions [59].

There are also several useful toolboxes for adversarial examples generation, e.g. Cleverhans [60], AdvBox [61], Adversarial-Playground [62], which can promote the research in this field. Cleverhans, a software library developed by Papernot *et al.* [60], provides standard implementations of different adversarial example generation methods. AdvBox which is a toolkit developed by Baidu, supports TensorFlow, PaddlePaddle, and Caffe2 framework to generate adversarial examples for deep learning models [61]. Adversarial-Playground is a web-based visualization tool developed by Norton and Qi [62]. It can implement common adversarial example generation methods against CNN.

3) ADVERSARIAL ATTACKS IN REAL WORLD CONDITIONS

The above adversarial attacks are demonstrated in the laboratory conditions. In order to convince the community that the adversarial example is a real concern in practice, some researchers have illustrated their adversarial attacks in practical real world conditions [7]. We will review and analyze these works from the following four aspects: 1) Cellphone camera attack; 2) Face recognition system attack; 3) Road sign recognition attack; 4) and 3D objects attack.

Road Sign Recognition Attack: Object recognition is an important task of autonomous vehicles, which needs to identify road signs, pedestrians, etc. However, Evtimov *et al.* [54] demonstrate that adversarial examples are robust under various physical conditions, such as changes in view angles, distance and resolution. They propose two practical attack methods [54]. The first is a poster-printing attack, in which the attacker prints the adversarial example generated by C&W attacks and other algorithms as a poster, and then overlaid it on the real road sign. The second is the sticker perturbation attack, in which the attacker prints the perturbations on the paper, and then pastes it on the real road sign. These perturbations in the physical conditions can successfully make the model misclassification, e.g., the stop sign is recognized as the speed limit sign [54].

Cellphone Camera Attack: Kurakin *et al.* [46] printed the adversarial examples generated by FGSM, BIM, and so on.

Then, they use a mobile phone to take pictures of the printed adversarial examples. At last, they use TensorFlow Android Camera Demo to classify these images. It was shown that most of these images were misclassified [46]. This indicates that the adversarial examples are robust under printing and taking pictures.

Face Recognition System Attack: Face recognition is an important technique in computer vision, and has been widely used in video surveillance and access control systems. Sharif *et al.* [53] propose adversarial examples working in physical conditions, which can allow an attacker to evade system identification or impersonate others. An eyeglass frame with added perturbations is printed and worn by an attacker. Then the attacker is verified by a face recognition system. It is demonstrated that the attacker who wears such glasses will be incorrectly identified as another person, thus avoiding the detection of the face recognition system [53].

3D Objects Attack: The 3D object in real world is a difficult target for generating adversarial examples since it involves many angles. Athalye *et al.* [55] propose an Expectation Over Transformation (EOT) framework which can construct adversarial examples over different object transformations, and thus can print adversarial 3D objects. Their experiments demonstrate that a 3D-printed turtle will be classified as a rifle by ImageNet.

These adversarial example attacks [46], [53]–[55] take into account the effects of various physical factors, such as angle of view, distance, illumination, etc., to make the generated adversarial examples be robust to physical conditions. Hence, the generated adversarial examples can succeed in real physical conditions. However, compared with digital adversarial examples, the perturbations added in these physical adversarial examples are larger and more conspicuous, which are easy to be noticed visually.

D. MODEL EXTRACTION ATTACK

Recent studies show that an adversary can steal the machine learning model by observing the output labels and confidence levels with respect to the chosen inputs. This attack, also known as *model extraction attack* or *model stealing attack*, has become an emerging threat. The summary of model extraction attack works is presented in Table 4.

Tramèr *et al.* [63] first proposed the model extraction attack, i.e., an attacker tries to steal the machine learning model through multiple user inquiries. When inputting normal queries through prediction APIs, the model will return a predicted label with a confidence level. Based on this service, they demonstrate the model extraction attack on three types of models: logistic regression, decision trees and neural networks [63]. Two online machine learning services are used for evaluation, Amazon and BigML.

Yi *et al.* [64] propose a model stealing method by constructing a functionally equivalent model based on deep learning. It works in a black-box scenario, where the adversary can only obtain the predicted labels from the target model and use deep learning to infer and then build an equivalent model [64]. Specifically, they use the input data to query the target model, and use the results returned by the target model to label the input data [64]. The labeled data is used to train a model that have similar functions as the target model. Chandrasekaran *et al.* [65] show that model extraction is similar to active learning. They formulate model extraction into query synthesis active learning, and propose model extraction attacks with no auxiliary information. These methods [63]–[65] train a model similar to the target model by black-box accessing to the target model, which do not require the attackers to have the knowledge of the target model. However, existing model extraction attacks need to query the target model many times. If the system limits the number of queries, these model extraction attacks may not be able to complete.

Wang and Gong [66] aim at stealing the hyperparameters of machine learning models by using a learner. This method sets the gradient of the model to be 0, and then calculates hyperparameters of the model by solving linear equations [66]. The method [66] demonstrates that model extraction attacks can steal hyperparameters from many machine learning models, e.g., SVM, logistic regression, ridge regression, and neural networks. However, this method requires attackers to know the learning algorithm, the training data, etc. Milli *et al.* [67] present an algorithm to learn a model through querying the gradient information of the target model for specific inputs. It is shown that gradient information can quickly reveal the model parameters. They conclude that gradient is a more efficient learning primitive than the predicted label [67]. However, this heuristic method introduces high computational overhead and they only evaluate their model extraction attacks on a two-layer neural network.

E. RECOVERY OF SENSITIVE TRAINING DATA

In addition to the above *model extraction attacks*, the other two privacy-related attacks on machine learning are: (i) *Membership inference attack*, in which the attacker tries to determine if a specific sample data is used when training the model; (ii) *Model inversion attack*, in which the attacker infers some information about the training data. Similar to model extraction attack, the membership inference attack and model inversion attack also aim at the popular machine-learning-as-a-service. These three privacy-related attacks on machine learning models are illustrated in Fig. 6. The attacks of recovering sensitive training data are summarized in Table 5.

1) MODEL INVERSION ATTACK

Fredrikson *et al.* [68] first propose the model inversion attack in pharmacogenetics tasks. By using black-box access and auxiliary information about a patient, the attacker can recover the patient's genomic information. Fredrikson *et al.* [69] further propose a model inversion attack by exploiting the confidence values of predictions. They demonstrate the attack on decision tree based lifestyle surveys, and NN based facial recognition. For example, in the facial recognition model,

TABLE 4. Summary of model extraction attack works.

Approaches	Working mechanism	Effect	Advantages	Disadvantages
Tramèr et al. [63]; Shi et al. [64]; Chandrasekaran et al. [65]	Extraction through prediction APIs; Train a model similar to the target model; Use active learning [65]	Can generate a model with the same functionality as the target model	Extract model parameters without the knowledge of the target model	Need to query the target model frequently
Wang and Gong [66]	Calculate hyperparameters by solving linear equations	Steal the hyperparameters of models	Suitable for various machine learning algorithms	Require the knowledge of the target algorithm and the training set
Milli et al. [67]	Query gradients of a model to chosen inputs, and heuristic model reconstruction	Can reconstruct the target model	More efficiently than querying labels based methods	High computational overhead and only evaluate on a two-layer NN

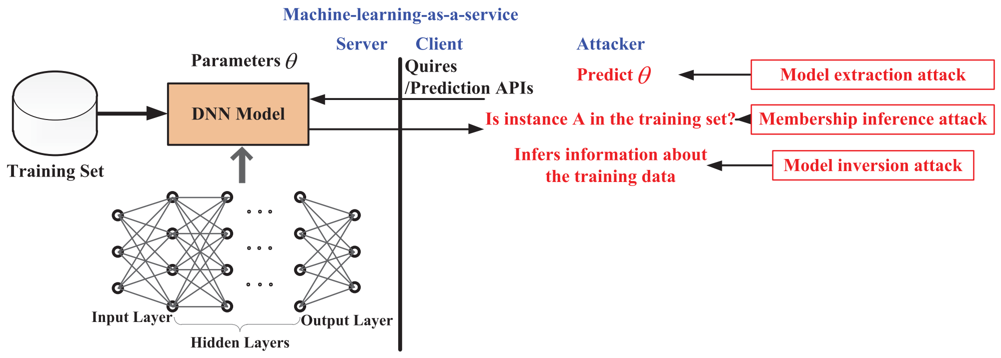


FIGURE 6. Overview of three privacy-related attacks on machine learning models: model extraction attack, membership inference attack, and model inversion attack.

TABLE 5. Summary of the attacks on recovering sensitive training data.

Approaches	Working mechanism	Effect	Advantages	Disadvantages
Model inversion attack [68] [69]	Estimate sensitive attributes through black-box access to the model	Infer sensitive attributes of the training data	Can infer sensitive attributes of data in sensitive applications	Require the knowledge of non-sensitive attributes of the data
Membership inference attack [70]	Train a model to distinguish training data and non-training data	Infer whether a given data is in the training data or not	Generic	Depends on the overfitting of the model
GAN based strong attacks [71]	GAN-based privacy information stealing	Can steal private information from collaborative deep learning	Generic; make differential privacy ineffective	The attacker needs to be an insider of the collaborative deep learning framework

the attacker who knows the user's name can recover a recognizable image of the user's face [69]. The method in [69] has less false positives than the method in [68], however, these methods [68], [69] require the attacker to know the non-sensitive attributes of the data, which may be difficult to obtain in practice [72].

2) MEMBERSHIP INFERENCE ATTACK

Liu *et al.* [73] illustrate security threats in cognitive systems. Specifically, they show that attacker can access to confidential training data or replicate the processing model by using only the public accessible services of the model. Shokri *et al.* [70] propose the so called membership inference attack, in which the adversary can estimate whether a given data is in the training set of a target model. Particularly, they use the target model's prediction of training and non-training data to train a membership inference model [70]. According to the output of the target model, the generated membership inference model can identify the differences in the prediction of the target model on its training data and the data that hasn't been used for its training. The membership inference attack proposed in [70] is generic, but the success of member inference attacks depends on the overfitting of the model [13], [70]. If it is a well-generalized model, the success rate of the membership inference attack is low.

3) GAN BASED STRONG ATTACKS

In order to protect the user's private data, researchers have proposed collaborative deep learning framework recently, where each party trains his model locally and only a small subset of parameters are shared. Differential privacy is also introduced to obfuscate the parameters for protection [74]. However, Hitaj *et al.* [71] propose a Generative Adversarial Networks (GAN) based strong attack which can break the above distributed or federated framework. The adversary trains a GAN to generate equivalent samples of the private training set, where the generated samples have same distribution as the private training set. It is shown that the differential privacy based collaborative deep learning framework is ineffective when facing such attacks [71]. However, the attacker in [71] needs to be an insider of the collaborative deep learning framework which has permission to obtain model parameters from the service provider.

F. DISCUSSION ABOUT ATTACKS

In the last decade, most of the attacks on machine learning were adversarial example attacks, while the other four types of attacks were significantly less. Among them, the adversarial example studies on images are the majority, while the adversarial example studies on speech and text are relatively less. Privacy-related attacks have emerged in recent years and have received increasing attentions. We summarize the treads of machine learning attacks as follows:

1) The attacks move towards more practical, and real physical conditions, such as the adversarial example attacks in real world conditions as described in Section III-C.3. For example, attacks against the face recognition system on a mobile phone or in surveillance cameras, or attacks on the road sign recognition system of driverless cars.
2) The attacks are getting stronger and stronger, and can even subvert humans' conventional cognition. For example, the state-of-the-art adversarial examples can not only make the model output wrong predictions (e.g., incorrectly identify the stop sign as a speed limit sign), but can also make the model be not aware that this is a road sign [75] or be not aware that this is a person [76]. For example, by pasting the printed adversarial example picture on the clothes, human can hide himself in front of a person detector [76]. This type of attack can be used to evade the surveillance systems.
3) Attacks toward biometric authentication systems are emerging. In the era of intelligent Internet of Things, authentication and control are two key features. There are many biometric-based authentication and control systems, such as fingerprint-based and voice-based systems. However, the above attacks can successfully break through these biometric authentication systems, thus threaten the security of the control system. For example, intelligent speech forgery can fool the automatic speaker verification system and thus hack into a system.

IV. DEFENSES

In this section, we will review and analyze the defenses against the above attacks according to the life cycle of machine learning systems. The existing defense techniques for machine learning, which covers the countermeasures against the above five security threats, are summarized in Fig. 7. The defense approaches against the above five attacks are analyzed in Section IV-A∼Section IV-E, respectively. We summarize the defense techniques in Section IV-F.

A. DEFENSES AGAINST POISONING ATTACKS

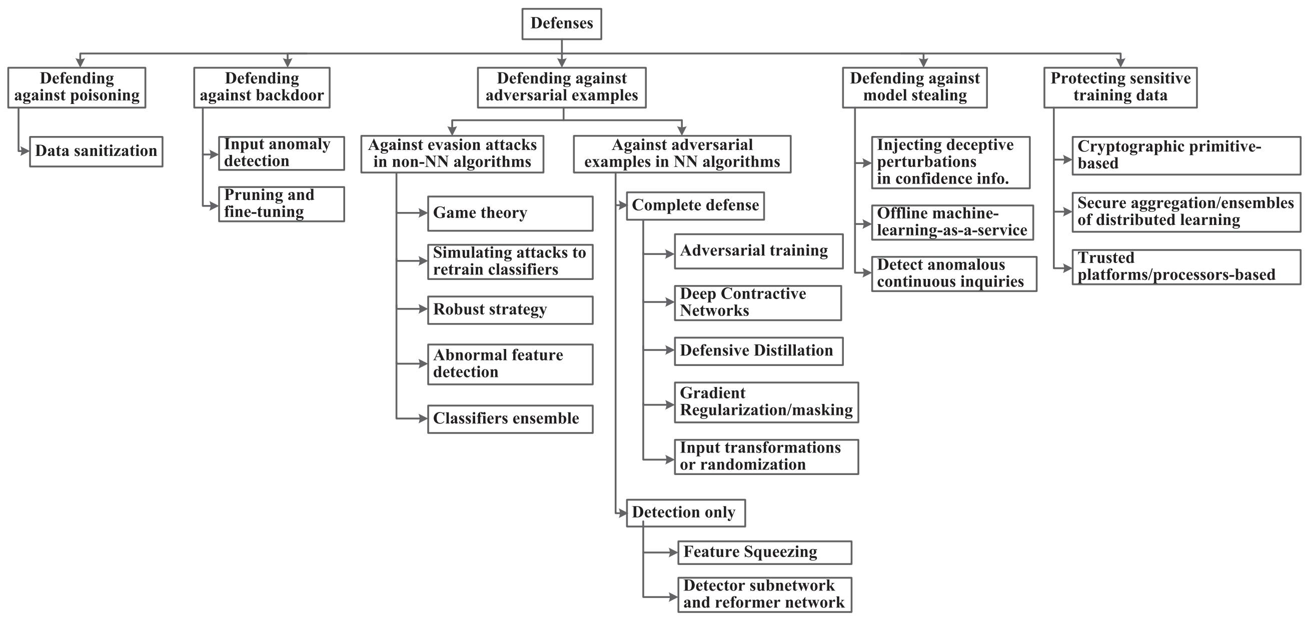
We discuss the defense against poisoning attacks based on whether it is targeted on the NN models. The summary of existing defense techniques against poisoning attacks is presented in Table 6.

1) DEFENSES AGAINST POISONING ATTACKS IN NON-NN MODELS

Defenses in Anomaly Detection or Security-Related Detection: Rubinstein *et al.* [14] propose a defending technique, named ANTIDOTE, against poisoning attacks on an anomaly detector. ANTIDOTE uses robust statistics to mitigate the influence of outliers and can reject the poisoned samples. Biggio *et al.* [77] consider poisoning attacks mitigation as outliers detection problems, which are small in number and have shifted distribution compared to the normal training data. Therefore, they use Bagging Classifiers which is an ensemble method to mitigate the influence of these outliers (poisoning samples) in the training set. Specifically, they use different training data to train multiple classifiers, and combine the predictions of multiple classifiers to reduce the influence of outliers in the training set [77]. They evaluate the ensemble method on spam filter and a web-based IDS against poisoning attacks [77]. However, training multiple classifiers will introduce significant overhead. Chen *et al.* [78] propose a defense technique, named KUAFUDET, against poisoning attacks in malware detection systems. KUAFUDET uses a self-adaptive learning framework and uses a detector to filter suspicious false negatives which will then be fed into the training procedure [78]. These methods [14], [77], [78] can improve the robustness of learning algorithms and mitigate the influence of outliers on the trained model, but these methods mainly focus on binary classification problems and introduce additional overhead.

Defending SVM: Zhang and Zhu [79] propose a game theory based defense for distributed SVM. They use game theory to analyze the conflicting interests between the attacker and the learner. Nash equilibrium is used to predict the outcome of the learner in adversarial settings [79]. This method can prevent wrong updates and prevent the poisoning data from reducing the performance of the distributed SVM. However, such game theory based defense technique requires expensive computational overhead.

Defenses in Algorithms and Processing Methods: Liu *et al.* [80] propose a robust Linear Regression method against poisoning attacks. The technique first uses low-rank matrix factorization then uses principle component regression to prune poisoned samples. Jagielski *et al.* [23] propose a defense algorithm, named TRIM, for Regression Learning. TRIM uses an iterative method to estimate regression parameters, while a trimmed loss function is used to remove and isolate suspicious poisoning points. Steinhardt *et al.* [81] construct defense approaches against poisoning attacks by using outlier removal and risk minimization. Baracaldo *et al.* [82] propose a data provenance based defense technique for



Defenses
Defending against poisoning
Data sanitization
Defending against backdoor
Input anomaly detection
Pruning and fine-tuning
Defending against adversarial examples
Against evasion attacks in non-NN algorithms
Game theory
Simulating attacks to retrain classifiers
Robust strategy
Abnormal feature detection
Classifiers ensemble
Against adversarial examples in NN algorithms
Complete defense
Adversarial training
Deep Contractive Networks
Defensive Distillation
Gradient Regularization/masking
Input transformations or randomization
Detection only
Feature Squeezing
Detector subnetwork and reformer network
Defending against model stealing
Injecting deceptive perturbations in confidence info.
Offline machine-learning-as-a-service
Detect anomalous continuous inquiries
Protecting sensitive training data
Cryptographic primitive-based
Secure aggregation/ensembles of distributed learning
Trusted platforms/processors-based

FIGURE 7. Summary of defense techniques for machine learning.

TABLE 6. Summary of defense techniques against poisoning attacks.

Targeting NN or non-NN	Target	Approaches	Working mechanism	Effect	Advantages	Disadvantages
Defenses in non-NN models	Anomaly detection or security detection	Defending anomaly detectors [14]; In adversarial classification tasks [77]; In malware detection systems [78]	Using robust statistics; Bagging ensembles; Using self-adaptive learning and introducing a camouflage detector	Mitigate the influence of outliers	Improve the robustness of learning algorithms	Mainly focus on binary classification problems; Additional overhead
	SVM	Game-theoretic defense in distributed SVM [79]	Game theory and rejection method	Reject malicious updates	Prevent poisoning and wrong updates	Not generic; High computational overhead
	Algorithms and processing methods	Robust linear regression [80]; Regression learning [23]; Certified defenses [81]; Data provenance [82]	Low-rank matrix factorization and principle component regression; Iterative method; Outlier removal and risk minimization; Data provenance based on contextual information	Remove poisoning data	Improve the robustness of learning algorithms	Not generic
Defenses in NN models	\	In NN [27];	Check the loss of the model	Detect the poisoning attack	Simple, generic	Not fully evaluated in the experiment
		In collaborative deep learning systems [83]	Statistical-based: identify features with abnormal distributions	Automatically identify malicious users	Defend against poisoning attacks without affecting model performance	Defense performance is affected by the number of malicious users
Defenses in special applica-tions	\	In healthcare [29]	Monitor the accuracy deviations on the training set	Detect poisoning data	Generic	High computational overhead

online and re-trained applications. Practically, they take advantage of the origin and transformation information of the training data to detect poisoning points. These methods [23], [80]–[82] can improve the robustness of learning algorithms, but these methods are not generic and only suitable for specific algorithms.

2) DEFENSES AGAINST POISONING ATTACKS IN NN MODELS

Yang *et al.* [27] propose a countermeasure for NN against poisoning attacks based on calculating the loss of the model. An input data which introduces a larger loss than the threshold is considered as a suspicious data. This method [27] is simple and generic, but they only present one simple detection result without fully evaluating the defense method. Shen *et al.* [83] propose a system, named AUROR, to defend the collaborative deep learning systems. Since poisoning data has a strong influence in the distribution of the features learned by the model, AUROR filters out suspicious users by identifying anomalous features [83]. AUROR defends against poisoning attacks without affecting the performance of the target model, but the defense performance

TABLE 7. Summary of defenses against backdoor attacks.

Approaches	Working mechanism	Effect	Advantages	Disadvantages
Activation clustering [84]; Fine-Pruning [85]; Neural cleanse [86]; Strengthen the NN models [39]	Activation clustering-based detection; Fine-pruning; Neuron pruning, input filter, and unlearning based model patching; Adding an input preprocessor, input anomaly detection, and continuing re-training.	Detect and mitigate the backdoors	Generic for most DNNs	High computational overhead
STRIP [87]	Perturb the input and observe the prediction	Detect the backdoors	Run-time detection	May fail when facing adaptive attacks

of this method is affected by the number of malicious users.

3) DEFENSES AGAINST POISONING ATTACKS IN SPECIAL APPLICATION SCENARIOS

Mozaffari-Kermani *et al.* [29] propose a countermeasure defending healthcare systems by monitoring the accuracy deviations of the training set and the number of the added data. This method is generic, and can provide protection against poisoning attacks for different target models. However, this method needs to train the model periodically [29], which results in a high computational overhead.

B. DEFENSES AGAINST BACKDOOR ATTACKS

Table 7 summarizes defenses against backdoor attacks. Chen *et al.* [84] propose an activation clustering method for protecting DNN by detecting and removing backdoors. The proposed activation clustering method can detect poisonous training data, even under multiple backdoors scenario. Liu *et al.* [85] investigate two defense approaches for DNN against backdoor attacks, named pruning and fine-tuning. Then, they propose a method combining the pruning and fine-tuning, called fine-pruning, to mitigate the impact of a backdoor. Wang *et al.* [86] identify and mitigate the backdoors by three techniques, using a input filter to identify inputs with triggers, using neuron pruning to patch the model, and using an unlearning based model patching. Liu *et al.* [39] strengthen the NN models against Trojans via three approaches: input anomaly detection using SVM and decision trees; adding an input preprocessor, i.e., an autoencoder which is trained on the legitimate training data thus can preprocess input data based on their distribution; and continuing re-training which can make the model ''forget'' the Trojan. These defense methods [39], [84]–[86] are suitable for most DNNs, but these methods require expensive computational overhead when detecting and mitigating backdoors.

Gao *et al.* [87] propose a run-time Trojan detection technique for DNN, named STRIP. The idea is to perturb the input and observe the prediction randomness in terms of entropy. If it is a Trojan input, the prediction of this perturbation is almost unchanged [87]. If it is a clean input, the prediction change caused by this perturbation will be large. This method is fast and can detect Trojans at runtime, but this method may fail when facing adaptive attacks [88].

C. DEFENSES AGAINST ADVERSARIAL EXAMPLES ATTACKS

We discuss the defenses against adversarial examples attacks based on whether it is targeted on the NN models. The summary of defenses against adversarial examples is presented in Table 8.

1) EARLIER DEFENSES AGAINST EVASION ATTACKS IN THE CONTEXT OF NON-DEEP LEARNING ALGORITHMS

The earlier defense works against evasion attacks are mostly for non-deep learning algorithms in the context of spam filtering, malware detection, and IDS. Dalvi *et al.* [4] use game theory to formulate the adversarial classification problem, which can produce optimal classifier according to the adversary's strategy. Nelson *et al.* [5] propose two defense techniques in the context of spam filtering, measuring the impact of each email on system performance with and without that email, and using dynamic threshold settings so that the rankings will be invariant to the scores shift caused by an attack.

In adversarial machine learning scenario, it is generally expected that the classifiers are robust, which are not sensitive to the changes in the distribution of the data [89]. Globerson and Roweis [90] introduce a robust learning strategy to avoid feature over-weighting by exploiting the game theory to analyze robustness. Particularly, they develop classifiers that are resilient to feature deletions, and evaluated on handwritten digit recognition and spam detection tasks. Similarly, Kołcz and Teo [89] propose two methods, averaged classifiers and feature reweighting to improve the robustness, and evaluate on spam email classification task. Biggio *et al.* [91] experimentally evaluated the effectiveness of two techniques, random subspace method and bagging, on improving the robustness of the linear classifiers under adversarial machine learning. Šrndić and Laskov [92] propose a hierarchical document structure based malicious PDF detection, and evaluate the robustness of this method under several evasion attacks. Demontis *et al.* [43] propose an adversary-aware secure learning paradigm for Android malware detection which can mitigate the negative effects of evasion attacks.

In [93], Li *et al.* propose an iterative re-training method to improve the robustness of classifiers. They re-train the classifier by iteratively adding malicious instances that can evade detection to the training data. The iterative re-training method minimizes the risk of evasion attacks and improves the ability of the classifier to resist evasion attacks [93].

TABLE 8. Summary of defenses against adversarial examples.

Targeting NN or non-NN	Approaches	Working mechanism	Effect	Advantages	Disadvantages
Defenses in non-NN models	Game theory [4]; Defenses for Spam filter [5]; Feature weighting [89]; Feature deletion [90]; Multiple classifier system [91]; Malicious PDF detection [92]; Android malware detection [43]; Binary Domains [93]; SecDefender [94]; Feature Selection [95]; SecureDroid [96]; Conserved features [97]; Data transformations [98]	Game theory; Measuring the impact of each email and using dynamic threshold; Feature weighting; Feature deletion; Using multiple classifiers; Structural properties difference detection; Adversary-aware approach; Iterative re-training; Classifier re-training and security regularization; Feature selection; Using conserved features; Data transformations	Improves the ability of the model to resist evasion attacks	Simple and effective	Not generic; Mainly focus on binary classification problems
Defenses in NN models	Adversarial training [2], [99]	Adversarial training	Improve the robustness of the target model	Simple, effective, and generic	Increase training overhead
	Saddle point formulation [100]; Deep Contractive Network [101]; Defensive distillation [102]; Input gradient regularization [103]	Using saddle point formulation; Using a smoothness penalty; NN distillation; Input gradient regularization	Mitigate the effects of adversarial examples on the model	Generic to most NNs	May increase the training complexity or may be defeated by stronger attacks
	Input transformations [104]; Randomization [105]	Apply image transformations or randomization to the input of the model	Remove perturbations in adversarial examples	Simple; Do not modify the model	Attackers can bypass these defenses
	Feature squeezing [106]	Compare the predictions of the model on the original input and the squeezed input	Can detect adversarial examples	Simple, low computational overhead	Cannot defend against adaptive attackers
	MagNet [107]; Detector subnetwork [108]	Use detector network(s) to detect adversarial examples	Can detect (and reform [107]) adversarial examples	Generic to most NNs	Require training an additional detector network

In [94], Chen *et al.* propose a malware detection method, named SecDefender, which exploits the classifier re-training and security regularization to enhance the robustness of the classifier. The experimental results show that even if the attacker has the knowledge of the target classifier, SecDefender can also defend against evasion attacks effectively [94].

Zhang *et al.* [95] propose an adversarial feature selection method to defend against evasion attacks. The features selected by this method can improve the performance of the model, and also improve the model's capability to resist evasion attacks. The evaluation results on spam and PDF malware detection show that the adversarial feature selection method outperforms traditional feature selection methods [95]. In [96], Chen *et al.* select features that are difficult for attackers to manipulate to generate a secure model. Besides, they combine the results of multiple models to ensure that the feature selection method does not affect the performance of the classifier. Similarly, Liang *et al.* [97] train the secure model by extracting such features that cannot be modified unless compromising the malicious functionality of the malware. Bhagoji *et al.* [98] use dimensionality reduction to protect the model from evasion attacks. However, this method of reducing the feature dimensions may also affect the performance of the model.

These defense methods [4], [5], [43], [89]–[98] are simple and effective. However, most defense methods focus on binary classification problems, such as spam detection, malware detection and IDS. These methods may not be applicable to other machine learning tasks.

2) DEFENSES AGAINST ADVERSARIAL EXAMPLES IN THE CONTEXT OF DEEP LEARNING ALGORITHMS

The defense techniques against adversarial examples in the context of deep learning algorithms can be further divided into two types [7], *complete defense techniques* and *detection only techniques*. The complete defense techniques aim at making the model be able to identify the correct label of an adversarial example. On the other hand, the detection only techniques only need to identify whether the input instance is a potentially adversarial example.

Complete Defense Approaches: A common conclusion is that adversarial training can be used as the first line of defense against adversarial attacks [2], [7], [99]. Goodfellow *et al.* [2] propose a fast adversarial examples generation method, named "fast gradient sign method" (FGSM) and suggest to use these generated adversarial examples for adversarial training. It is shown that adversarial training can regularize the model thus can improve the robustness of the model [2]. The above adversarial training method was later expanded to large models and large datasets by Kurakin *et al.* [99]. Adversarial training is simple and can effectively improve the robustness of the target model. However, adversarial training inevitably increases the training data size and the training overhead. Moreover, the attacker can once again generate the adversarial examples based on the model that has already been adversarially trained, which will evolve into a training competition [7].

Madry *et al.* [100] propose a robust deep learning strategy against adversarial attacks by using saddle point formulation. They also suggest that the model capacity has a significant impact on the robustness, therefore a larger model capacity is required against adversarial attacks. Gu and Rigazio [101] propose a defense model, named Deep Contractive Network (DCN). DCN uses a smoothness penalty similar to the contractive autoencoder in the training phase. As a result, it can effectively make the output of the model less sensitive to the input [101], thus can increase the robustness of the model to adversarial examples. Papernot *et al.* [102] propose a defense technique, named defensive distillation, which uses two networks, an initial network and a distilled network. The knowledge contained in probability vectors is transferred through distillation, thus can improve the generalization capabilities of DNNs so as to enhance their robustness to perturbations [102]. Ross and Doshi-Velez [103] propose a denfense technique by regularizing the input gradients of DNNs. The method penalizes the degree of change in the output to the change in the input. As a result, small adversarial perturbations cannot significantly change the predictions of the model. These methods [100]–[103] are generic to most neural networks, but these methods may increase the training complexity of the model or may be defeated by stronger attack methods [7]. For example, it is shown that the defensive distillation method [102] can be defeated by the Carlini and Wagner attack [49].

Guo *et al.* [104] use input transformations to counter adversarial images. They applied transformations, e.g., JPEG compression, bit-depth reduction, image quilting and total variance minimization, to the input images before feeding them to the ImageNet. Their experimental results show that after training on these transformed images, the convolutional network classifier can effectively defense the adversarial perturbations. This indicates that some image transformation operations can remove perturbations [104]. Similarly, Xie *et al.* [105] demonstrated that randomization in the test phase can mitigate the effects of adversarial examples. In particular, two randomization operations are applied to the input before fed into the model, random resizing and random padding. It is shown that these randomization operations can defend both one-step and iterative adversarial attacks [105]. These methods [104], [105] are simple and do not modify or re-train the target model. However, attackers can also apply image transformations to the generated adversarial examples to make them robust, thus ensure that the generated adversarial examples can bypass these defense methods.

Detection Only Approaches: Xu *et al.* [106] use feature squeezing operations to detect adversarial examples. Two feature squeezing operations are used, including reducing the color depth of each pixel and performing spatial smoothing. After that, they compare the predictions of the DNN model on the original input and the squeezed input [106]. If there is a large difference between the predictions, the input is considered as an adversarial example. The method is simple and has low computational overhead, but this method cannot defend against the adaptive attacker who has full knowledge of the defense technique [106]. Meng and Chen [107] propose a defense framework, named MagNet. One or more detector networks and a reformer network are used in MagNet. The instances that are found to have large perturbations are rejected. On the other hand, the instances with small perturbations are reformed towards the normal instance by the reformer network [107], and then the reformed instances are fed into the classifier for correct classification. Metzen *et al.* [108] harden the DNN model by augmenting a detector subnetwork which is trained on the classification task of identifying adversarial perturbations in the inputs. It is demonstrated that adding such a detector subnetwork to the main classification network can detect adversarial examples effectively [108]. Although the detector is only trained to detect one type of attack, it can generalize well to similar or weaker attacks. These methods [107], [108] use a detector to determine whether the input of the target model is an adversarial example, which are independent of the target model and can provide adversarial examples detection for different models. However, these detection approaches [107], [108] require training an additional detector to detect the input of the model which usually introduce high computational overhead.

D. DEFENSES AGAINST MODEL STEALING ATTACKS

In this section, we will discuss the defenses against model stealing attacks. The summary of defenses against model stealing attacks is presented in Table 9. An intuitive defense method against model stealing is that when the label is outputted, no confidence information is provided. However, this will affect the quality of the service. Lee *et al.* [109] protect the machine learning models by injecting deceptive perturbations/noises in the confidence information to mislead the adversary. As a result, the adversary can only use the labels for model stealing and require significantly more quires to steal the model [109]. This method protect the model without affecting the model accuracy. However, if the attacker increases the number of queries, the attack can still succeed.

Hanzlik *et al.* [110] propose an offline machine-learning-as-a-service framework, named MLCapsule. In MLCapsule framework, the model is allowed to be executed on the users' side so that users can protect the privacy of their data while the server can still control the model with intellectual property protection. However, this method needs to encrypt the user's data, which will introduce the additional computational overhead. Juuti *et al.* [111] propose a technique to detect DNN model stealing attacks, named PRADA. Since the adversary steals the model via prediction APIs, PRADA analyzes the distribution of quires and detects anomalous continuous quires. PRADA is generic and can provide detection of model extraction attacks for most models. However, this method is not robust to dummy queries [112] which are used to hide the anomalous queries of the attackers.

TABLE 9. Summary of defenses against model stealing attacks.

Approaches	Working mechanism	Effect	Advantages	Disadvantages
Deceptive perturbations [109]	Injecting deceptive perturbations in the confidence information	Makes the confidence information useless for attackers	Protect the model without affecting the model accuracy	Attackers can still succeed with increased number of queries
MLCapsule [110]	Execute the model on the isolated execution environments	Provide a secure environment to run models	Protect both the model security and the users' data privacy	Increase computational and hardware overhead
PRADA [111]	Monitor the distribution of continuous queries	Detect anomalous queries	Generic	Not robust to dummy queries

E. PRIVACY-PROTECTED MACHINE LEARNING TECHNIQUES AGAINST RECOVERY OF SENSITIVE TRAINING DATA

The defenses for machine learning models against recovery of sensitive training data can be broadly divided into three categories: (1) Cryptographic primitive-based approaches, e.g., differential privacy, homomorphic encryption; (2) Secure aggregation/ensembles of distributed learning, e.g., Federated Learning [113], Private Aggregation of Teacher Ensembles (PATE) [114]; (3) Trusted platforms/processors-based approaches. The summary of privacy-protected machine learning techniques against recovery of sensitive training data is presented in Table 10.

1) CRYPTOGRAPHIC PRIMITIVE-BASED APPROACHES

Abadi *et al.* [115] develop differential privacy based deep learning framework. In addition, they propose techniques to improve the efficiency of the differential privacy based training, so as to achieve a balance between the privacy, efficiency, software complexity, and model quality. The differential privacy-based method [115] is generic to most machine learning algorithms, but this method adds noise to the gradient in the process of training the model, which will affect the accuracy of the model. Jayaraman *et al.* [123] demonstrate that there is a trade-off between the privacy and the performance of the model in current privacy-preserving mechanisms. In other words, when protecting the privacy of the model, the current privacy-preserving mechanisms will also sacrifice the performance of the model [123]. Phong *et al.* [116] show that the distributed learning framework for privacy protection in [74] may still reveal secret data to the server. Therefore, they improve the technique in [74] by using asynchronous stochastic gradient descent to NN and introduce homomorphic encryption to the framework [116]. The homomorphic encryption-based method [116] use cryptographic primitive to ensure the security and privacy of the training data without affecting the accuracy of the model, but this method inevitably bring high computational overhead in the training process of the model. Rouhani *et al.* [117] propose a provably-secure learning framework, named DeepSecure, in which Yao's garbled circuit protocol is used to perform the secure computation. They also propose optimized implementation and low-overhead techniques to reduce the overhead. Although more efficient than homomorphic encryption-based method, this method is not easy to deploy in practical applications.

TABLE 10. Summary of privacy-protected machine learning techniques against recovery of sensitive training data.

Categories	Approaches	Working mechanism	Effect	Advantages	Disadvantages
Cryptographic primitive-based approaches	Deep learning with differential privacy [115]	Differential privacy-based	Prevent attackers from stealing the privacy of the training data	Generic	Affect the accuracy of the model
	Deep learning via additively homomorphic encryption [116]	Homomorphic encryption-based	Protect the gradients	Without affecting the accuracy of the model	High computational overhead
	DeepSecure [117]	Yao's garbled circuit and low-overhead techniques	Secure deep learning with low overhead	Efficient and scalable	Not easy to deploy in practice
Secure aggregation/ensembles of distributed learning	Distributed learning [74]; SecureML [118]	Jointly learn the model using asynchronously stochastic gradient descent [74]; Secure two-party computation [118]	Multiple users jointly learn a model without sharing their data	Protect the privacy of data for multiple users while achieve high model accuracy	High computational overhead
	Federated Learning [113]	Secure multi-party computation-based			
	PATE [114]; Scalable PATE [119]	Private aggregation of teacher ensembles	Prevent attackers from accessing data	Generic	Need to train an additional student model
Trusted platforms or processors-based approaches	Oblivious multi-party learning on trusted processors [120]; Deep learning on multi-source private data [121]	Run the model on trusted platforms	Prevent attackers from accessing the model	Low computational overhead	Require the support of hardware platforms
Specific application-oriented	Disguised-Nets [122]	Use image disguising techniques	Protect model and data privacy	Simple while maintaining good model performance	Only suitable for image data

2) SECURE AGGREGATION/ENSEMBLES OF DISTRIBUTED LEARNING

Distributed Learning Framework: Shokri and Shmatikov [74] propose a distributed learning framework for privacy protection, where multiple entities can jointly learn a NN model without sharing their own data and only share a small subset of the learned parameters. The key idea is that the stochastic gradient descent based deep learning algorithms can be executed in parallel [74]. Mohassel and Zhang [118] propose new protocols for privacy preserving machine learning models, including logistic regression, linear regression, and neural network. The protocol works in a two-server model, where two servers train their own models on the distributed private data by using secure two-party computation [118].

Secure Aggregation for Federated Learning: Bonawitz *et al.* [113] propose a secure Multi-Party Computation (secure aggregation) based Federated Learning framework. In a distributed learning model, the secure aggregation can protect the gradient information of each user's model.

Private Aggregation of Teacher Ensembles: Papernot *et al.* [114] propose a privacy-protection training model, named PATE. They trained multiple models using disjoint sensitive datasets, named teacher models. The student model is learned based on the output of a noisy aggregation of all the teachers, thus cannot access the data or parameters of an individual teacher model [114]. Papernot *et al.* [119] extend PATE to large scale tasks and uncurated datasets with errors. Specifically, they develop new noisy aggregation approaches to ensemble the teacher models with less noise.

The distributed learning framework methods [74], [118] and the secure aggregation method [113] can protect the privacy of data for multiple users in distributed deep learning framework, but the encryption algorithm or security protocol used in the aggregation of multiple users' data will increase the computational overhead. PATE methods [114], [119] can provide training data protection for most models, but PATE methods need to train an additional student model for users to access the model.

3) TRUSTED PLATFORMS/PROCESSORS-BASED APPROACHES

Another ideal privacy protection solution is to run the machine learning model on a trusted platform. Ohrimenko *et al.* [120] develop machine learning models on trusted processors, the Intel SGX. They propose oblivious machine learning algorithms for SVM, decision trees, matrix factorization, *k*-means clustering, and neural networks. They demonstrate that such trusted processors based machine learning implementations have higher performance than cryptography-based privacy-preserving solutions [120]. Hynes *et al.* [121] propose a privacy-preservation deep learning framework, named Myelin. Myelin combines the differential privacy and trusted hardware enclaves to train the machine learning model. These methods [120], [121] use trusted platforms to prevent attackers from accessing the model, and provide a secure privacy protection mechanism for the training data of the model. Their computational overhead is lower than encryption-based defense methods. However, these methods [120], [121] rely on trusted platforms, which require the support of hardware platforms.

4) SPECIFIC APPLICATION-ORIENTED

Sharma and Chen [122] use image disguising technique for privacy-protection in image based deep learning tasks. They demonstrate that the images undergoing transformations and block-wise permutation can guarantee both the privacy and the availability of the DNN model. This method is simple and effective. However, this method is only suitable for protecting the privacy of image data.

F. DISCUSSION ABOUT DEFENSES

Existing protection methods can be summarized as follows. In the training phase, the defensive works against poisoning attacks or backdoor attacks, can be called *data sanitization* [24], in which the anomalous poisoned data is filtered out first before feeding into the training phase. The anomaly detectors are usually based on training loss, nearest neighbors, and so on [24].

In the test phase, the defense techniques against adversarial examples can be called *smoothing model outputs* [11], i.e., reduce the sensitivity of the model's output to the changes in the input. The defense techniques against sensitive information leakage consists of three major categories, *distributed learning framework*, *traditional cryptographic primitives-based approaches* (e.g., based on differential privacy, homomorphic encryption), and *trusted platform-based approaches*.

V. SECURITY EVALUATIONS

A. DESIGN-FOR-SECURITY

In a typical machine learning system design flow, the designer focuses on the model selection and the performance evaluation, but does not consider the security issues. With the emergence of the above-mentioned security attacks on machine learning systems, it is necessary to perform security evaluations on the machine learning system at the design stage and use latest secure machine learning techniques. This paradigm can be called *design-for-security*, which is a necessary complement to the typical paradigm *design-for-performance*. For example, Biggio *et al.* [41] propose a framework for security evaluation of classifiers. They simulate different level of attacks by increasing the adversary's ability and adversary's knowledge. Similarly, Biggio *et al.* [56] suggest to evaluate the security of classifiers by empirically evaluate the performance degradation under a set of potential attacks. Particularly, they generate training set and test set and simulate attacks for security evaluations.

B. EVALUATION USING STRONG ATTACKS

Carlini and Wagner [50] evaluate ten recent detection methods and show that these defenses can be bypassed by using strong attacks with new loss functions. Therefore, it is suggested to perform security evaluation of machine learning algorithms using strong attacks, which includes the following two aspects. First, evaluate under white-box attacks, e.g., the attacker has perfect knowledge about the model, the data and the defense technique, and has strong ability to manipulate the data or the model. Second, evaluate under high-confidence attacks/maximum-confidence attacks rather than minimally-perturbed attacks only [3]. Carlini and Wagner [49], [50] show that the defense techniques proposed against minimally-perturbed attacks can be bypassed by using high-confidence attacks. The initial works on adversarial examples aim at analyzing the sensitivity of deep learning algorithms to minimal perturbations. However, in order to analyze the security of a deep learning algorithm, it is more reasonable to use the maximum-confidence adversarial attacks which can reflect the security of an algorithm under more powerful attacks [3].

C. EVALUATION METRICS

First, it is suggested to use more metrics [50], e.g., not only accuracy, but also the confusion matrix (true positive, false positive, true negative, false negative), precision, recall, ROC (receiver operating characteristic) curve, and AUC (the area under the ROC curve), to report the performance of the learning algorithm, so that the complete performance information can be reflected, and is easy for comparison with other works. Second, the *security evaluation curves* [3] can be used. Biggio and Roli [3] propose to use security evaluation curves to evaluate the security of learning systems. The security evaluation curves characterize the system performance under different attack strength and attackers with different level of knowledge [3], thus can provide comprehensive evaluation of the system performance under attacks, which is also convenient for comparing different defense techniques.

VI. FUTURE DIRECTIONS

Machine learning security is a very active research direction. There have been a lot of works on tit-for-tat attacks and defenses in recent years. We present the following future directions on machine learning security:

1) *Attacks under real physical conditions*. There have been a lot of security attacks against machine learning models, most of which were verified in digital simulation experiments. The effectiveness of these attacks under real-world physical conditions, and the works targeting at real physical world conditions, are active research topics. For example, physical adversarial examples can deceive road sign recognition systems, but these physical adversarial examples are visually obvious and unnatural. Recently, a lot of works aimed at generating natural robust physical adversarial examples. Besides, DNN-based intelligent monitoring systems have been widely deployed. For humans, is it possible to achieve invisibility in front of the object detectors via adversarial examples? Due to the large intra-class differences of humans, and the dynamic movements and different postures of humans, this is a more challenging task than digital adversarial example attacks and the road sign-oriented adversarial example attacks.
2) *Privacy-preserve machine learning techniques.* In recent years, the privacy of machine learning has received increasing attentions. The deployment of deep learning needs to address the issue of privacy protection, including the protection of the model's parameters from the service provider's perspective and the protection of user's privacy data from the user's perspective. To date, the efficiency of cryptographic primitive-based machine learning approaches needs to be improved, which usually introduce high overhead to the training of the model and may degrade the performance of the model. The distributed or integration-based training frameworks still face efficiency and performance problems. It is necessary to study secure and efficient machine learning algorithms, models and frameworks. A collaborative design combining hardware platform, software, and algorithm to protect the privacy of DNN is a promising direction.
3) *Intellectual property (IP) protection of DNN.* The training of deep learning models require massive training data, and a lot of hardware resources to support. The training process usually takes weeks or months. In this sense, machine learning models are valuable commercial intellectual properties of the model providers thus need to be protected. At present, there are only a few watermarking based IP protection works for machine learning models [124]. More effective and secure IP protection methods for DNN are still open problems.
4) *Remote or lightweight machine learning security techniques.* Machine learning will be widely used for platforms in distributed, remote, or IoT scenarios. In these resource-constrained scenarios, many existing security techniques are not applicable. How to provide reliable and effective remote or lightweight machine learning security technique is a promising research direction.
5) *Systematic machine learning security evaluation method.* To date, little work has been done on machine learning security evaluation. Specifically, there is no comprehensive method to evaluate the security and robustness of models and the security and privacy of the model's training data and parameters. There is also no unified method and comprehensive metrics to evaluate the performance of current attacks and defenses. A system evaluation method involving security, robustness, privacy of the machine learning systems, and the corresponding evaluation metrics, need to be studied and established.
6) *What are the underlying reasons behind these attacks and defenses on machine learning?* There are some discussions in the literature, but it still lacks of consensus.

The reasons behind these attacks remain open problems. Besides, the opaqueness of the model makes it currently lack an explanation for the output of the model. However, in some critical applications, such as healthcare and banking, the interpretability of the applied model is required [11].

VII. CONCLUSION

Machine learning based applications are ubiquitous, yet machine learning systems still face a variety of security threats throughout their lifecycles. Machine learning security is an active research topic and remains an open problem. This paper presents a comprehensive survey on machine learning security covers the whole lifecycle of machine learning systems with respect to five major types of attacks and their corresponding countermeasures. A general conclusion is that the threats are real, and new security threats are constantly emerging. For example, studies show that there is a transferability in adversarial examples, which means adversarial examples can generalize well between different machine learning models. It is shown that poisoning examples can also generalize well across different learning models. The transferability can be used to launch attacks in black-box scenarios effectively. Due to the unexplained nature of machine learning models, the essential reasons for these attacks, i.e., is the adversarial example a bug or an intrinsic property of the model, need to be further studied. This paper can hopefully provide comprehensive guidelines for designing secure, robust and private machine learning systems.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Represent.*, Apr. 2014, pp. 1–10.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, Mar. 2015, pp. 1–11.

[3] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.

[4] N. N. Dalvi, P. M. Domingos, Mausam, S. K. Sanghai, and D. Verma, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, Aug. 2004, pp. 99–108.

[5] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *Proc. USENIX Workshop Large-Scale Exploit. Emerg. Threat.*, Apr. 2008, pp. 1–9.

[6] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, Nov. 2010.

[7] N. Akhtar and A. S. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, Jul. 2018.

[8] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[9] M. S. Riazi and F. Koushanfar, "Privacy-preserving deep learning and inference," in *Proc. Int. Conf. Comput.-Aided Design ICCAD*, Nov. 2018, pp. 1–4.

[10] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.

[11] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and privacy in machine learning," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Apr. 2018, pp. 399–414.

[12] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*. [Online]. Available: http://arxiv.org/abs/1708.06733

[13] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Jul. 2018, pp. 268–282.

[14] B. I. P. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. Lau, S. Rao, N. Taft, and J. D. Tygar, "ANTIDOTE: Understanding and defending against poisoning of anomaly detectors," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, Nov. 2009, pp. 1–14.

[15] P. Li, Q. Liu, W. Zhao, D. Wang, and S. Wang, "Chronic poisoning against machine learning based IDSs using edge pattern detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[16] B. Biggio, G. Fumera, F. Roli, and L. Didaci, "Poisoning adaptive biometric systems," in *Proc. Int. Workshops Stat. Tech. Pattern Recognit. Struct. Synt. Pattern Recognit.*, Nov. 2012, pp. 417–425.

[17] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *Proc. Int. Conf. Biometrics (ICB)*, Jun. 2013, pp. 1–7.

[18] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Conf. Mach. Learn.*, Jun. 2012, pp. 1467–1474.

[19] B. Biggio, I. Pillai, S. R. Bulò, D. Ariu, M. Pelillo, and F. Roli, "Is data clustering in adversarial settings secure?" in *Proc. ACM Workshop Artif. Intell. Secur. AISec*, Nov. 2013, pp. 87–98.

[20] B. Biggio, K. Rieck, D. Ariu, C. Wressnegger, I. Corona, G. Giacinto, and F. Roli, "Poisoning behavioral malware clustering," in *Proc. Workshop Artif. Intell. Secur. Workshop AISec*, Nov. 2014, pp. 27–36.

[21] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning," in *Proc. 32nd Int. Conf. Mach. Learn.*, Jul. 2015, pp. 1689–1698.

[22] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. Annu Conf. Neural Inf. Proc. Syst.*, Dec. 2016, pp. 1885–1893.

[23] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 19–35.

[24] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," 2018, *arXiv:1811.00741*. [Online]. Available: http://arxiv.org/abs/1811.00741

[25] Y. Wang and K. Chaudhuri, "Data poisoning attacks against online learning," 2018, *arXiv:1808.08994*. [Online]. Available: http://arxiv.org/abs/1808.08994

[26] X. Zhang, X. Zhu, and L. Lessard, "Online data poisoning attack," 2019, *arXiv:1903.01666*. [Online]. Available: http://arxiv.org/abs/1903.01666

[27] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," 2017, *arXiv:1703.01340*. [Online]. Available: http://arxiv.org/abs/1703.01340

[28] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur. AISec*, 2017, pp. 27–38.

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

[30] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, Dec. 2018, pp. 381–392.

[31] C. Miao, Q. Li, H. Xiao, W. Jiang, M. Huai, and L. Su, "Towards data poisoning attacks in crowd sensing systems," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. - Mobihoc*, Jun. 2018, pp. 111–120.

[32] Y. Ji, X. Zhang, and T. Wang, "Backdoor attacks against learning systems," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2017, pp. 1–9.

[33] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*. [Online]. Available: http://arxiv.org/abs/1712.05526

[34] C. Liao, H. Zhong, A. C. Squicciarini, S. Zhu, and D. J. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," 2018, *arXiv:1808.10307*. [Online]. Available: http://arxiv.org/abs/1808.10307

[35] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in CNNs by training set corruption without label poisoning," 2019, *arXiv:1902.11237*. [Online]. Available: http://arxiv.org/abs/1902.11237
[36] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," 2018, *arXiv:1807.00459*. [Online]. Available: http://arxiv.org/abs/1807.00459
[37] M. Zou, Y. Shi, C. Wang, F. Li, W. Song, and Y. Wang, "PoTrojan: Powerful neural-level trojan designs in deep learning models," 2018, *arXiv:1802.03043*. [Online]. Available: http://arxiv.org/abs/1802.03043
[38] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Feb. 2018, pp. 1–15.
[39] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 45–48.
[40] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Aug. 2005, pp. 641–647.
[41] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Sep. 2013, pp. 387–402.
[42] N. Rndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 197–211.
[43] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! A case study on Android malware detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 4, pp. 711–724, Jul. 2019.
[44] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proc. Netw. Distrib. Syst. Symp.*, Feb. 2016, pp. 1–15.
[45] H. Dang, Y. Huang, and E. C. Chang, "Evading classifiers by morphing in the dark," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 119–133.
[46] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, Feb. 2017, pp. 1–14.
[47] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.
[48] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[49] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.
[50] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur. (AISec)*, 2017, pp. 3–14.
[51] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 86–94.
[52] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. D. McDaniel, "Adversarial examples for malware detection," in *Proc. 22nd Eur. Symp. Res. Comput. Secur.*, Sep. 2017, pp. 62–79.
[53] 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 *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, 2016, pp. 1528–1540.
[54] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.
[55] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 284–293.
[56] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 984–996, Apr. 2014.
[57] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.
[58] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, 2017, pp. 506–519.
[59] J. Lu, H. Sibai, E. Fabry, and D. A. Forsyth, "NO need to worry about adversarial examples in object detection in autonomous vehicles," 2017, *arXiv:1707.03501*. [Online]. Available: http://arxiv.org/abs/1707.03501
[60] N. Papernot *et al.*, "Technical report on the CleverHans v2.1.0 adversarial examples library," 2018, *arXiv:1610.00768*. [Online]. Available: http://arxiv.org/abs/1610.00768
[61] D. Goodman, X. Hao, Y. Wang, Y. Wu, J. Xiong, and H. Zhang, "Advbox: A toolbox to generate adversarial examples that fool neural networks," 2020 *arXiv:2001.05574*. [Online]. Available: http://arxiv.org/abs/2001.05574
[62] A. P. Norton and Y. Qi, "Adversarial-playground: A visualization suite showing how adversarial examples fool deep learning," in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2017, pp. 1–4.
[63] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 601–618.
[64] Y. Shi, Y. Sagduyu, and A. Grushin, "How to steal a machine learning classifier with deep learning," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2017, pp. 1–5.
[65] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," 2018, *arXiv:1811.02054*. [Online]. Available: http://arxiv.org/abs/1811.02054
[66] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 36–52.
[67] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt, "Model reconstruction from model explanations," in *Proc. Conf. Fairness, Accountability, Transparency FAT*, Jan. 2019, pp. 1–9.
[68] 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 *Proc. 23rd USENIX Secur. Symp.*, Aug. 2014, pp. 17–32.
[69] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, 2015, pp. 1322–1333.
[70] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.
[71] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, 2017, pp. 603–618.
[72] S. Hidano, T. Murakami, S. Katsumata, S. Kiyomoto, and G. Hanaoka, "Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 115–126.
[73] B. Liu, C. Wu, H. Li, Y. Chen, Q. Wu, M. Barnell, and Q. Qiu, "Cloning your mind: Security challenges in cognitive system designs and their solutions," in *Proc. 52nd Annu. Design Autom. Conf. - DAC*, Jun. 2015, pp. 1–5.
[74] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.
[75] S. Chen, C. Cornelius, J. Martin, and D. H. Chau, "ShapeShifter: Robust physical adversarial attack on faster R-CNN object detector," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Sep. 2018, pp. 52–68.
[76] S. Thys, W. V. Ranst, and T. Goedeme, "Fooling automated surveillance cameras: Adversarial patches to attack person detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–7.
[77] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *Proc.10th Int. Conf. Mult. Classif. Syst.*, Jun. 2011, pp. 350–359.
[78] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *Comput. Secur.*, vol. 73, pp. 326–344, Mar. 2018.
[79] R. Zhang and Q. Zhu, "A game-theoretic defense against data poisoning attacks in distributed support vector machines," in *Proc. IEEE 56th Annu. Conf. Decis. Control (CDC)*, Dec. 2017, pp. 4582–4587.

[80] C. Liu, B. Li, Y. Vorobeychik, and A. Oprea, "Robust linear regression against training data poisoning," in *Proc. 10th ACM Workshop Artif. Intell. Secur. AISec*, Nov. 2017, pp. 91–102.

[81] J. Steinhardt, P. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 3517–3529.

[82] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, "Mitigating poisoning attacks on machine learning models: A data provenance based approach," in *Proc. 10th ACM Workshop Artif. Intell. Secur. AISec*, 2017, pp. 103–110.

[83] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl. (ACSAC)*, 2016, pp. 508–519.

[84] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Proc. AAAI Workshop Artif. Intell. Saf.*, Jan. 2019, pp. 66–73.

[85] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. 21st Int. Symp. Res. Attacks Intrusions Def.*, Sep. 2018, pp. 273–294.

[86] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[87] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against trojan attacks on deep neural networks," in *Proc. 35th Annu. Comput. Secur. Appl. Conf. ACSAC*, 2019, pp. 113–125.

[88] B. Gia Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," 2019, *arXiv:1908.03369*. [Online]. Available: http://arxiv.org/abs/1908.03369

[89] A. Kołcz and C. H. Teo, "Feature weighting for improved classifier robustness," in *Proc. 6th Conf. Email Anti-Spam*, Jul. 2009, pp. 1–8.

[90] A. Globerson and S. T. Roweis, "Nightmare at test time: Robust learning by feature deletion," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, Jun. 2006, pp. 353–360.

[91] B. Biggio, G. Fumera, and F. Roli, "Multiple classifier systems for robust classifier design in adversarial environments," *Int. J. Mach. Learn. Cybern.*, vol. 1, nos. 1–4, pp. 27–41, Dec. 2010.

[92] N. Šrndić and P. Laskov, "Detection of malicious PDF files based on hierarchical document structure," in *Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp.*, Feb. 2013, pp. 1–16.

[93] B. Li and Y. Vorobeychik, "Evasion-robust classification on binary domains," *ACM Trans. Knowl. Discovery from Data*, vol. 12, no. 4, pp. 1–32, Jul. 2018.

[94] L. Chen, Y. Ye, and T. Bourlai, "Adversarial machine learning in malware detection: Arms race between evasion attack and defense," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Sep. 2017, pp. 99–106.

[95] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 766–777, Mar. 2016.

[96] L. Chen, S. Hou, and Y. Ye, "SecureDroid: Enhancing security of machine learning-based detection against adversarial Android malware attacks," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf. ACSAC*, 2017, pp. 362–372.

[97] L. Tong, B. Li, C. Hajaj, C. Xiao, N. Zhang, and Y. Vorobeychik, "Improving robustness of ML classifiers against realizable evasion attacks using conserved features," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 285–302.

[98] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2018, pp. 1–5.

[99] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. 5th Int. Conf. Learn. Represent.*, Apr. 2017, pp. 1–17.

[100] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent.*, May 2018, pp. 1–10.

[101] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent.*, May 2015, pp. 1–9.

[102] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.

[103] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Proc. 32nd AAAI Conf. Artif. Intel.*, Feb. 2018, pp. 1660–1669.

[104] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," in *Proc. 6th Int. Conf. Learn. Represent.*, May 2018, pp. 1–12.

[105] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *Proc. 6th Int. Conf. Learn. Represent.*, May 2018, pp. 17–32.

[106] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Feb. 2018, pp. 1–15.

[107] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, 2017, pp. 135–147.

[108] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Feb. 2017, pp. 1–12.

[109] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against neural network model stealing attacks using deceptive perturbations," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 43–49.

[110] L. Hanzlik, Y. Zhang, K. Grosse, A. Salem, M. Augustin, M. Backes, and M. Fritz, "MLCapsule: Guarded offline deployment of machine learning as a service," 2018, *arXiv:1808.00590*. [Online]. Available: http://arxiv.org/abs/1808.00590

[111] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: Protecting against DNN model stealing attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Jun. 2019, pp. 512–527.

[112] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," 2019, *arXiv:1907.05587*. [Online]. Available: http://arxiv.org/abs/1907.05587

[113] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," 2016, *arXiv:1611.04482*. [Online]. Available: http://arxiv.org/abs/1611.04482

[114] N. Papernot, M. Abadi, U. Erlingsson, I. J. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *Proc. 5th Int. Conf. Learn. Represent.*, Apr. 2017, pp. 1–16.

[115] M. Abadi, A. Chu, I. J. Goodfellow, H. B. Mcmahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, Oct. 2016, pp. 308–318.

[116] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[117] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[118] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[119] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson, "Scalable private learning with PATE," in *Proc. 6th Int. Conf. Learn. Represent.*, May 2018, pp. 1–34.

[120] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 619–636.

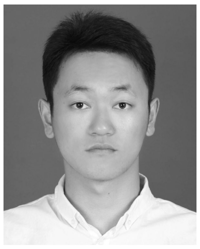
[121] N. Hynes, R. Cheng, and D. Song, "Efficient deep learning on multi-source private data," 2018, *arXiv:1807.06689*. [Online]. Available: http://arxiv.org/abs/1807.06689

[122] S. Sharma and K. Chen, "Disguised-Nets: Image disguising for privacy-preserving outsourced deep learning," 2019, *arXiv:1902.01878*. [Online]. Available: http://arxiv.org/abs/1902.01878

[123] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1895–1912.

[124] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *Proc. 27th USENIX Secur. Symp.*, Aug. 2018, pp. 1615–1631.

MINGFU XUE (Member, IEEE) received the Ph.D. degree in information and communication engineering from Southeast University, Nanjing, China, in 2014. From July 2011 to July 2012, he was a Research Intern with Nanyang Technological University, Singapore. He is currently an Assistant Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing. From 2014 to 2019, he has presided ten research projects or fundings. He has published around 30 articles in security related journals and international conferences. His research interests include hardware security, hardware Trojan detection, artificial intelligence security, and secure and private machine learning systems. He is a member of ACM, IEICE, CCF, and CAAI, a Committee Member of the Chinese Artificial Intelligence and Security Professional Committee, an Executive Committee Member of the ACM Nanjing Chapter, and a Committee Member of the Computer Network and Distributed Computing Specialized Committee of Jiangsu Province. He has also been a Technical Program Committee Member for over ten international conferences. He won the Best Paper Award in ICCCS2015. He is the Programme Chair of the third Chinese Symposium on Hardware Security.



CHENGXIANG YUAN received the B.S. degree in digital media technology from Yancheng Teachers University, in 2017. He is currently pursuing the master's degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include artificial intelligence security, and secure and private machine learning systems.

HEYI WU received the master's degree in information engineering from Southeast University, Nanjing, China, in 2013. He joined the Suzhou State Taxation Administration (Suzhou SAT) as a Network Security Engineer, in 2013. He is currently the CTO of the Nanjing Upsec Network Security Technology Research Institute Company, Ltd., Nanjing. He is also a Lecturer of two security organizations (FreeBuf & Pediy). His research interests include network security, AI security, and blockchain security. He is a member of the Artificial Intelligence and Security Professional Committee of Chinese Association for Artificial Intelligence (CAAI).

YUSHU ZHANG (Member, IEEE) received the Ph.D. degree from the College of Computer Science, Chongqing University, Chongqing, China, in December 2014. He held various research positions at the City University of Hong Kong, Southwest University, University of Macau, and Deakin University. He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He has published over 100 refereed journal articles and conference papers in these areas. His research interests include multimedia security, artificial intelligence, cloud computing security, big data security, the IoT security, and blockchain. He is an Editor of the *Signal Processing*.

WEIQIANG LIU (Senior Member, IEEE) received the B.Sc. degree in information engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2006, and the Ph.D. degree in electronic engineering from Queen's University Belfast (QUB), Belfast, U.K., in 2012. He was a Research Fellow with the Institute of Electronics, Communications and Information Technology (ECIT), QUB, from August 2012 to November 2013. In December 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently an Associate Professor. He has published one research book by Artech House and over 90 leading journal articles and conference papers. One of his articles is the Feature Paper of the IEEE TRANSACTIONS ON COMPUTERS (TC) issue in December 2017. His research interests include approximate computing, computer arithmetic, hardware security and VLSI design for digital signal processing, and cryptography. He has been a Technical Program Committee Member for several international conferences, including ARITH, DATE, ASAP, ISCAS, ASP-DAC, ICONI, SiPS, ISVLSI, and NANOARCH, and a Steering Committee Member of the IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS (TMSCS). His articles have received the Best Paper Award at the IEEE International Symposium on Circuits and Systems (ISCAS 2011) and ACM GLSVLSI 2015. He serves as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: Regular Papers, and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, and the Guest Editors of Proceedings of the IEEE and IEEE TETC (two special issues). He is also the Program Co-Chair of IEEE ARITH 2020.