Published: 15 December 2022
Online AM: 25 May 2022
Accepted: 13 May 2022
Revised: 05 May 2022
Received: 15 December 2021

SURVEY

Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems

ZHIBO WANG, Wuhan University, Wuhan, Hubei, China

JINGJING MA, Wuhan University, Wuhan, Hubei, China

XUE WANG, Wuhan University, Wuhan, Hubei, China

JIAHUI HU, Zhejiang University, Hangzhou, Zhejiang, China

ZHAN QIN, Zhejiang University, Hangzhou, Zhejiang, China

KUI REN, Zhejiang University, Hangzhou, Zhejiang, China

Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems

ZHIBO WANG, Wuhan University, China and Zhejiang University, China
JINGJING MA and XUE WANG, Wuhan University, China
JIAHUI HU, ZHAN QIN, and KUI REN, Zhejiang University, China

Machine learning (ML) has been universally adopted for automated decisions in a variety of fields, including recognition and classification applications, recommendation systems, natural language processing, and so on. However, in light of high expenses on training data and computing resources, recent years have witnessed a rapid increase in outsourced ML training, either partially or completely, which provides vulnerabilities for adversaries to exploit. A prime threat in training phase is called poisoning attack, where adversaries strive to subvert the behavior of machine learning systems by poisoning training data or other means of interference. Although a growing number of relevant studies have been proposed, the research among poisoning attack is still overly scattered, with each paper focusing on a particular task in a specific domain. In this survey, we summarize and categorize existing attack methods and corresponding defenses, as well as demonstrate compelling application scenarios, thus providing a unified framework to analyze poisoning attacks. Besides, we also discuss the main limitations of current works, along with the corresponding future directions to facilitate further researches. Our ultimate motivation is to provide a comprehensive and self-contained survey of this growing field of research and lay the foundation for a more standardized approach to reproducible studies.

CCS Concepts: • **Theory of computation → Adversarial learning**; • **Security and privacy → Systems security;**

Additional Key Words and Phrases: Poisoning attacks, adversarial machine learning, AI security

ACM Reference format:
Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. 2022. Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems. *ACM Comput. Surv.* 55, 7, Article 134 (December 2022), 36 pages.
https://doi.org/10.1145/3538707

This research was supported in part by National Key R&D Program of China (Grant No. 2020AAA0107705), National Natural Science Foundation of China (Grants No. 62122066, No. U20A20182, No. 61872274, and No. U20A20178), and Key R&D Program of Zhejiang (Grant No. 2022C01018).

Authors' addresses: Z. Wang, School of Cyber Science and Engineering, Wuhan University, Wuhan, China and School of Cyber Science and Technology, Zhejiang University, Hangzhou, China; email: zhibowang@zju.edu.cn; J. Ma and X. Wang, School of Cyber Science and Engineering, Wuhan University, Wuhan, China; emails: {jingjingma, shannonwang}@whu.edu.cn; J. Hu (corresponding author), Z. Qin, and K. Ren, School of Cyber Science and Technology, Zhejiang University, Hangzhou, China; emails: {jiahuihu, qinzhan, kuiren}@zju.edu.cn.

0360-0300/2022/12-ART134 $15.00
https://doi.org/10.1145/3538707

1 INTRODUCTION

In the past few years, **machine learning (ML)**, the de facto approach to **artificial intelligence (AI)**, has earned tremendous success and great popularity in multiple areas, including image classification, **natural language processing (NLP)**, automatic driving, and so on, thus quickly permeating our daily life. However, for resource-constrained learners, data-driven machine learning is not only data-hungry but also computationally intensive. Accordingly, it has been a routine for these learners to use outsourced data as well as third-party platforms, which provide a convenient way for AI developers to rapidly implant intelligence to machines without making clear the logic and theories behind data. Nevertheless, unclosed training style may result in critical security risks. As more applications rely on machine learning for automated decisions in safety-critical systems, such as medical diagnostics [15] and industrial control systems [37], widespread concerns have emerged about potential vulnerabilities introduced by threats in training phase, which particularly refer to poisoning attacks.

The ultimate goal of a poisoner is to subvert the predictions of ML systems by interfering with the training phrase. Poisoning attacks may take place in any circumstances where training phase is not a closure entirely manipulated by the learners. As is indicated in Figure 1, poisoning attacks seek to manufacture error predictions of ML systems by interfering with training phase. The kind of attack that relies on manipulating training data is called data poisoning attack, while the other type that directly exerts on the model is called model poisoning attack. Both outsourced data and third-party platforms provide convenience for poisoners. In lack of training data, some learners download pre-labeled datasets or collect data through a crawler program from anonymous and unverified sources on the web, where adversaries can abundantly deposit poisoned data and merely wait for web crawlers to download them. Through manipulating training data, poisoners are able to conduct data poisoning attacks to indirectly disrupt the final system. In lack of computing resources, some learners update private data to third-party platforms and outsource the training phase. The situation could be worse if a malicious third-party platform, like platforms of **Machine Learning as a Service (MLaaS)** [69] and providers of pre-trained models and codebases, can directly get access to the training process and modify the ML system. For instance, the recent booming of distributed computing such as federated learning architecture [56] makes it possible for adversaries to upload poisoned weights or updates to a central server, and further manipulate the function of the final model [15, 80].

Obviously, it is the training phase that causatively determines the ML model. The safety of the training phase is no less important than the inference phase. Although the security issues of the inference process are well studied and precisely concluded in a wide range of surveys and tutorials [13, 74, 89], the security issues of the training process, especially poisoning attacks, are less mentioned. As far as we know, there exist no thorough surveys specially focusing on poisoning attacks. Biggio et al. [5] and Xu et al. [86] define poisoning attacks as another branch of adversarial attacks besides adversarial examples [79], despite the fact that adversarial examples are aimed at a well-trained and fixed model in inference phase while poisoning attacks take place during the training phase. Goldblum et al. [24] merge poisoning attacks and backdoor attacks into data security issues. However, poisoning attacks are about not only data security but also any other segments (e.g., training algorithms and modeling procedure) in the training phase. Different from adversarial examples and backdoor attacks, poisoning attacks can manipulate the predictions of ML systems without any control of test samples. Therefore, poisoning attacks should not be unified with any other ML threats. An independent survey among poisoning attacks is worthy of attention.

The very beginning of poisoning attacks can be traced back to 2006 when Barreno et al. [2] first revealed that adversaries can maliciously "mis-train" the learning system to confuse the **Intrusion**

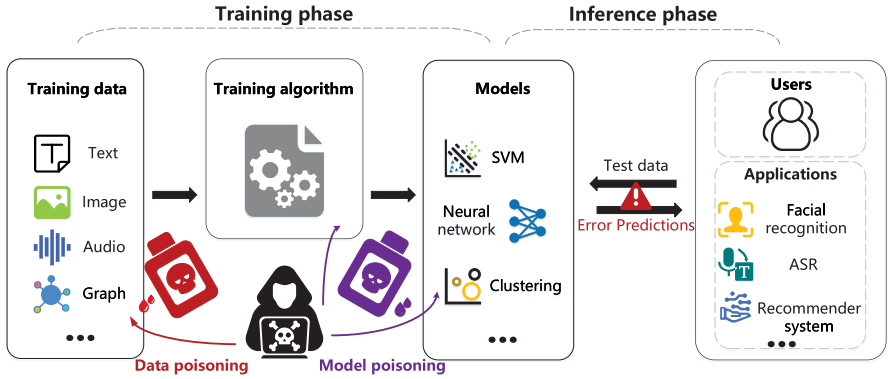


Fig. 1. The framework of poisoning attacks.

Detection Systems (IDS), thus manipulating a learning system to permit a specific attack. In 2012, Biggio et al. [4] formally put forward the concept of poisoning attack. In general, poisoning attacks aim at doing generic or specific damages to the function of the target systems by means of injecting a small fraction of poisoning samples into the training data or directly poisoning the parameters of the model. To be specific, adversaries can leverage poisoning attacks to increase misclassification for a image classifier or subvert the result of a recommendation system.

Recent years have witnessed substantial development in poisoning attacks and corresponding defenses. Victim models have been extended from simple machine learning algorithms (e.g., SVMs [4, 59] and logistic regressions [46, 59]) to deep learning neural networks [30, 75] and advanced paradigms like reinforcement learning [67, 78], federated learning [8, 15], multi-task learning [95], and so on. Apart from the pursue of poor accuracy in basic classification tasks [60], adversaries in various fields have applied poisoning attacks to recommendation system [16], malware detection system [10], industrial control systems [37], automatic driving systems [64], and so on, thus leading to unexpected situations. Faced with new threats continuously emerging, researchers began to take defensive measures [7, 22, 66, 71, 76] from perspectives of both data and models. Although there has been certain works about poisoning attacks and their defends, systematic reviews and standards are still in a lack. It is worthwhile to write a survey that summarizes and organizes recent research results in such a significant domain.

In this survey, we provide a comprehensive overview of the current status of poisoning attacks and propose some insights about future research directions with our understanding of works in the field. Our survey aims to summarize fundamental concepts, popular methods and countermeasures, as well as wide application areas of poisoning attacks, and further provide a comprehensive literature review on the new trend of security research above training phase of machine learning. Through categorizing and systematizing poisoning attacks and corresponding countermeasures, we hope to review how these weaknesses lead to exploitation of machine learning systems. One major mission of our survey is to categorize state-of-the-art researches in the field based on several identified dimensions to provide a richer understanding of the different facets of poisoning attacks. Besides, we provide a unified framework of qualitative analysis and standards for poisoning attacks. It is hoped that our work is able to lay a solid foundation for a more standardized approach for reproducible research works in the field.

In the following, we start by providing background on poisoning attacks, as well as introducing our definition of terms and a unified analysis framework in Section 2. We describe the baseline attack methods against conventional machine learning and their latest advance against deep

learning in Section 3. Subsequently, we introduce corresponding defense algorithms in Section 4. In Section 5, we display the extensive application of poisoning attacks against other learning paradigms in different learning tasks. Finally, we present future directions for poisoning attacks in Section 6 and our conclusion in Section 7.

2 DEFINITIONS AND UNIFIED FRAMEWORK

Before discussing the details about attack methods and corresponding countermeasures, in this section, we briefly describe and explain common technical terms used in poisoning attacks. Then, we provide a unified threat model (framework) to help understand and analyse the contents in the remainder of the article.

2.1 Prerequisites and Definitions

In the following, we will briefly introduce some prerequisites and definitions of necessary terms about typical machine learning systems and poisoning attacks.

2.1.1 Machine Learning. "Machine learning is a discipline focused on two interrelated questions: How can one construct computer systems that automatically improve through experience? and What are the fundamental statistical computational-information-theoretic laws that govern all learning systems, including computers, humans, and organizations" [33]. With the development of this field, machine learning has derived some new branches. We will introduce some main paradigms of machine learning as follows:

- **Supervised Learning.** This is the most widely used machine-learning paradigm that attempts to find a mapping $f : X \rightarrow Y$ based on the input data X and its label Y.
- **Unsupervised Learning.** Unsupervised learning generally refers to learning systems based on unlabeled data under assumptions about structural properties of the data. The blend across supervised learning and unsupervised learning is called semi-supervised learning.
- **Reinforcement Learning. Reinforcement learning (RL)** is a method that learns what to do by interacting with the environment.

Although these learning paradigms are classified in theory, many current researches integrate different categories. In addition, another high-impact branch of machine learning in recent years is **deep learning (DL)**, which relies on deep neural networks. Deep learning can be combined with either supervised learning or unsupervised learning. With the further studies of machine learning, researchers have put forward many new paradigms including federated learning, semi-supervised learning, multi-task learning, and so on. In Section 5, we will briefly introduce these typical paradigms and, respectively, summarize the applications of the poisoning attack in these paradigms, so we do not go into detail here.

2.1.2 Training Phase. ML training is a process where models learns a mapping between inputs and outputs from a set of training data. Figure 2 demonstrates how mainstream training phase works.

Machine learning training phase includes the following four steps:

- **Data Collection.** There is a saying that data determines the upper bound of machine learning while models and algorithms are merely approaching this upper bound. In lack of resources, recent learners usually adopt public dataset or outsourced data from the Internet.
- **Data Preprocessing.** This is a necessary process of review and remedy to prevent data from incompleteness, unfairness, anomaly and irregularity, including data cleaning, enhancement, and transformation. The dataset is normally divided into following parts, including training

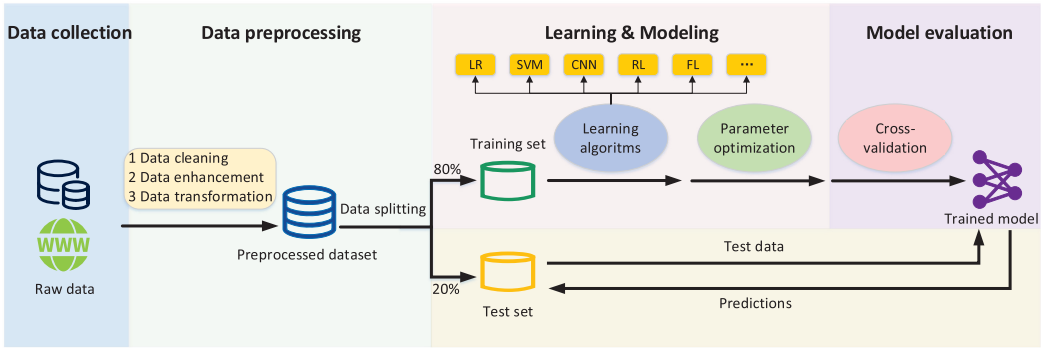


Fig. 2. Machine learning training phase.

data D_{train} and test data D_{test}. To be specific, training data D_{train} refers to data used for training a model, and test data D_{test} refers to data used in inference phase.

- **Learning and Modeling.** At this stage, the corresponding algorithm is selected according to the data that is given and the problem to be solved. Then learners solve the problem by building models based on these data and algorithm.
- **Model Evaluation.** After training, learners need to evaluate the performance of the model through methods like cross-validation with a validation set D_{val}, which is a disjoint dataset from the training set, also serving as an additional equipment for adversaries and defenders in poisoning attacks.

2.1.3 Inference Phase. At this phase, learners apply the trained model to various applications. Users send test data to the trained model and the model reply users with predictions in return.

2.1.4 Training Mode.

- **Batch Machine Learning and Online Machine Learning.** In batch learning, the method generates the best predictor by learning on the entire training data set at once. In online learning, data becomes available in a sequential order and is used to update the best predictor for future data at each step. In this case, the learner only has training data that have been shared so far, but has no knowledge about the incoming data.
- **Centralized Learning and Distributed Learning.** Centralized learning refers to a learning mode where the involved training data are centrally gathered to one train model [14]. Distributed learning in parallel leverages distributed data and computing resources to complete large scale computation, e.g., a hot issue of recent study called federated learning.

2.1.5 Poisoning Attack. Poisoning attacks seek to downgrade the predictions of machine learning systems by interfering with the training phase. In a poisoning attack, the attacker is assumed to control a fraction of the training data used by the learning algorithm or have access to other segments (e.g., training algorithms and modeling procedures), with the goal of subverting the entire learning system, or facilitating specific system evasion.

2.2 Threat Model

In this section, we will introduce a unified threat model and explain the definitions of necessary technical terms.

2.2.1 Attack Surface. Attack surface refers to the spot where the poisoning attack happens in the training phase, as is shown in Figure 2.

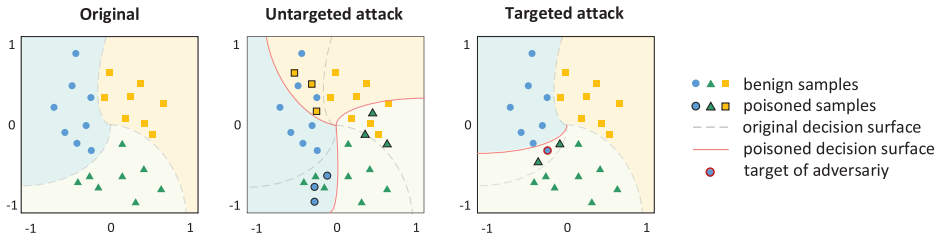


Fig. 3. The adversarial goals of poisoners.

- **Data Poisoning.** The kind of attack where adversaries manipulate the system through controlling a fraction of the training data used by the learning algorithm is called data poisoning attack. Usually, it happens when learners leverage out-sourced training data, such as third-party datasets and unidentified data from the Internet. Data poisoning attacks can be more precisely divided as follows:
  - **Dirty-label** data poisoning. In this kind of attacks, adversaries require the authority to modify the labels and the content of a fraction of the training data.
  - **Clean-label** data poisoning. In this kind of attacks, adversaries only have the authority to modify the content of poisoned data and then insert these data into the training dataset. But all the data are labeled by the victim himself.
- **Model Poisoning.** The other type that directly exerts on the model is called model poisoning attack. Usually, it happens when adversaries have access to learning algorithms or modeling procedure, which is out of control by the learners.

2.2.2 Adversary's Goal. As is shown in Figure 3, from the perspective of adversarial intention, poisoning attacks can be divided into two categories.

- **Targeted attack.** In this setting, the attacker aims to facilitate specific system evasion when victim models are put into use. Targeted attacks refer to the setting where the adversary knows a specific test data $X_t \in D_{test}$ over which the hypothesis will be tested, and expects to manipulate the predictions of the specific set X_t.
- **Untargeted attack.** In this case, the attacker aims to cause a denial of service, by inducing as many error predictions as possible, regardless of the classes in which they occur.

2.2.3 Adversary's Knowledge. Knowledge largely restrains the capacity and strategy of a adversary. Notably, the following settings are quite different from those in adversarial examples. Since poisoning attacks take place in training phase, there have not been a complete victim model for adversaries to query.

- **White-box.** In this setting, adversaries have full knowledge of the target ML model $M = (D_{train}, f, w)$, where f refers to the learning algorithm and w indicates the internal weights of the model. They clearly know the learning task and correspondingly know what kind of dataset and learning algorithm are chosen. They also have direct access to the training data and internal model weights. They keep an eye on the training phase, which is in a transparent white-box.
- **Black-box.** In the contrary of white-box setting, adversaries can not get access to the target system or the training dataset. Nevertheless, the adversaries are not absolutely ignorant of the victim. As any other potential user, the adversaries know the task that the system is designed to accomplish. And then they can infer what kind of algorithm and data the learner may choose. Based on the assumption above, the adversaries can easily collect a surrogate

training dataset $\hat{D}_{train}$ and simulate the original training process with surrogate learning algorithms $\hat{f}$ to train a surrogate victim model $\hat{M} = (\hat{D}_{train}, \hat{f}, \hat{w})$ in place of the original victim model $M = (D_{train}, f, w)$, where $\hat{w}$ indicates the model weights estimated by the adversaries.

- **Gray-box.** This is a complex setting between white-box and black-box settings where adversaries only have partial knowledge about the victims, i.e., information about either training data or the kind of victim model. In this setting, adversaries are also able to train a surrogate victim model $\hat{M} = (\hat{D}_{train}, f, \hat{w})$ or $\hat{M} = (D_{train}, \hat{f}, \hat{w})$ with current knowledge to make up for limited conditions.

2.2.4 Adversary's Capacity. Capacity defines what the attacker/defender can and cannot do to achieve their goal. In poisoning attacks, the adversaries influence the training phase with the following capacities. The first three capabilities usually appear in data poisoning while the last one belongs to model poisoners.

- **Data injection.** Adversaries with this capacity are able to inject poisoned data into the training dataset.
- **Data modification.** Adversaries with this capacity have access to the training set and can directly modify the content of the training data.
- **Label manipulation.** Adversaries with this capacity are able to manipulate the labeling process and label any target training data as they wish.
- **Model tampering.** Adversaries with this capacity can cause damage to the training algorithm and modeling process. They can even tamper with the weights of the model and some necessary documents in the system.

2.2.5 Technical Terms.

- **Benign model.** Benign model refers to the model trained without any pollution.
- **Victim model.** Victim model refers to the model that is or will be infected. The victims of poisoning attacks are not only conventional machine learning algorithms but also something more complicated and structured, like deep neural networks.
- **Poisoned samples.** Poisoned samples (also called adversarial training examples) is the modified training sample used in poisoning attacks during the training process.
- **Source label.** Source label indicates the ground-truth label of a poisoned or an attacked sample.
- **Target label.** Target label is the attacker-specified label. The attacker intends to make all attacked samples be predicted as the target label by the infected model.
- **Attack success rate (ASR)**. In untargeted attacks, attack success rate denotes the proportion of testing samples that are predicted as the wrong label by the infected model in the whole test dataset. In targeted attacks, attack success rate refers to the proportion of the target testing samples that are predicted as the adversary wishes among all the targets.

3 POISONING METHODS

In this section, we first give an overall taxonomy for existing poisoning methods, and then introduce detailed poisoning methods according to this taxonomy.

To provide a richer understanding of the different aspects of poisoning attacks, we collect representative researches and then categorize them based on different attack strategies. As is demonstrated in Figure 4, we build a taxonomy to figure out the following questions:

QUESTION 1: Which surface is attacked in ML training?

Regarding this question, we divide these methods into data poisoning and model poisoning.

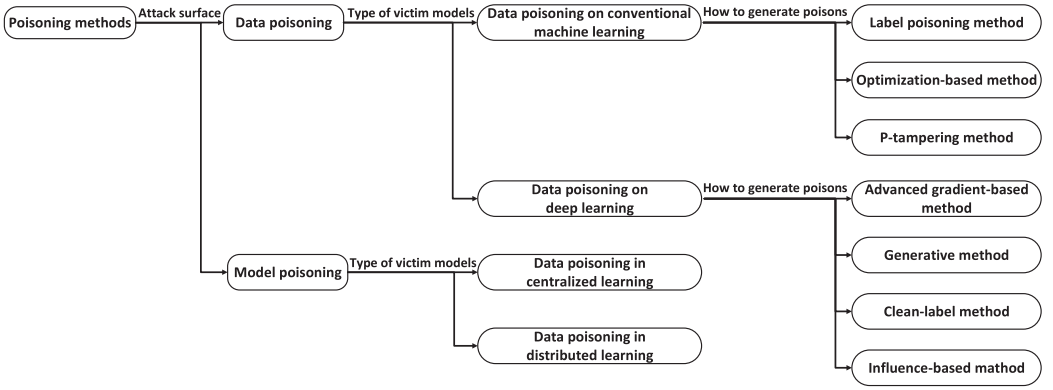


Fig. 4. The taxonomy of poisoning attacks.

Data poisoning attacks require access to training dataset, while model poisoning attacks directly manipulate learning and modeling process.

QUESTION 2: What does the victim model look like?

On account of different complexity, robustness and other inherent characteristics of different models, a certain poisoning methods apply to a certain victim models. For data poisoning, we mainly consider the complexity of victim models. For model poisoning, we establish classification standards depending on whether the victim model learns from a centralized or distributed way.

QUESTION 3: How does the adversary contaminate the victim model?

To answer the third question, we summarize three methods against conventional machine learning and four methods against deep learning for data poisoning. When it comes to model poisoning, there have not been as many researches in this newly emerging method as in data poisoning attacks, so we did not further classify it.

As is shown in Table 1, we classify representative researches using the taxonomy outlined above and identify every attack in both qualitative and a quantitative manner. Qualitatively, we characterize each research from additional dimensions including adversarial goal (i.e., targeted and untargeted) and adversary's capacity (i.e., white-box, gray-box, black-box), which is predefined in Section 2. Quantitatively, we propose following four dimensions for quantitative assessment and each dimension is measured on a categorical scale, ranging from low (*), medium (**), and high (***). The final score is computed by averaging over the attributes described below:

- **Complexity (CPLX) of victim.** This assessment criterion is used to describe the complexity of the victim model. Simple models (e.g., SVMs, linear and logistic regressions, etc.) are graded ⋆ and more complicated algorithms get grades ⋆⋆ (e.g., for simple feed forward or convolutional networks;) or ⋆ ⋆ ⋆ (e.g., for deep neural networks), according to concrete structures. The more complex the victim model is, the more difficult it is to craft effective poisons, thus resulting in the descent of attack success rate as well as efficiency.
- **Poison budgets.** We use poison budgets to describe the proportion of poisons in total training data. The statistical results are based on the lower and upper bounds that are demonstrated in the experiments of each paper. Strategies are rated as low budgets (⋆, i.e., the proportion of poisons is less than 5%), middle budgets (⋆⋆, i.e., the proportion ranges from 5% to 20%), and high budgets (⋆ ⋆ ⋆, i.e., the proportion is more than 20%).
- **Attack strength.** This criterion is used to evaluate the effectiveness of each strategy. In untargeted attacks, attack strength is illustrated by the proportion of testing samples that

Table 1. Catalog of Poisoning Attacks Following the Taxonomy Introduced in Section 3

Attacks	Adversarial goal	Adversary's capacity	Attack strategy								CPLX of victim	Poison budgets	Attack performance	
			Data poisoning for ML			Data poisoning for DL				Model poisoning			Attack strength	Attack CPLX
			S1	S2	S3	S4	S5	S6	S7					
[2]	T	white-box	✓								★	★★	–	★
[62], [1]	T & U	gray-box	✓	✓							★	★★	★ ★ ★	★
[85]	U	white-box	✓	✓							★	★ ★ ★	★★	★
[4]	U	white-box	✓	✓							★	★★	★★	★
[59]	U	white-box	✓	✓							★	★ ★ ★	–	★
[91]	U	white-box	✓	✓							★	★★	★	★
[94]	U	black-box	✓	✓							★	★ ★ ★	★ ★ ★	★
[31]	U	gray-box		✓		✓					★	★ ★ ★	★★	★
[46]	U	black-box	✓	✓							★	★ ★ ★	★★	★
[54]	T	black-box			✓				✓		–	–	–	–
[52]	T & U	black-box			✓				✓		–	–	–	–
[53]	T	black-box			✓				✓		–	–	–	–
[55]	T	black-box			✓				✓		–	–	–	–
[60]	U	gray-box	✓	✓		✓					★★	★★	★	★★
[87]	U	white-box	✓	✓			✓				★★	★ ★ ★	★	★★
[61]	U	gray-box	✓	✓			✓				★★	★ ★ ★	★	★ ★ ★
[18]	T & U	white-box	✓	✓			✓				★★	★ ★ ★	★ ★ ★	★★
[90]	T	white-box	✓	✓			✓				★★	★ ★ ★	★ ★ ★	★★
[34]	T	white-box		✓		✓		✓	✓		★★	★	★	★★
[35]	U	white-box	✓	✓				✓			★	★	★	★★
[16]	T	gray-box		✓				✓			★	★	★ ★ ★	★★
[75]	T	white-box		✓					✓		★ ★ ★	★	★	★★
[96]	T	black-box		✓					✓		★ ★ ★	★	★★	★★
[27]	T	black-box		✓					✓		★ ★ ★	★	★ ★ ★	★★
[30]	T	black-box		✓		✓			✓		★ ★ ★	★★	★ ★ ★	★ ★ ★
[23]	T	gray-box		✓		✓			✓		★ ★ ★	★	★ ★ ★	★★
[73]	T & U	gray-box								✓	★ ★ ★	★ ★ ★	★ ★ ★	★ ★ ★
[3]	T	gray-box		✓						✓	★ ★ ★	★ ★ ★	★ ★ ★	★★
[15]	U	gray-box		✓						✓	★ ★ ★	★ ★ ★	★ ★ ★	★★

S1: Label poisoning method. S2: Optimization-based method. S3: P-tampering method. S4: Advanced gradient-based method. S5: Generative method. S6: Influence-based method. S7: Clean-label method.

are predicted as the wrong label by the infected model in the whole test dataset. In targeted attacks, the strength of each attack is reflected by the attack success rate (ASR), i.e., the proportion of the target testing samples that are predicted as the adversary wishes among all the targets. (★, ★★, or ★ ★ ★, depending on the fooling rate).

- **Attack complexity (CPLX).** Here, we evaluate the complexity of concrete strategies in each research. As is known to us, this character is positively correlated to the complexity of the attack algorithm. Specifically, the complexity of simple algorithms is rated as ★, while the complexity of multiple iterative algorithms is marked ★★ or ★ ★ ★ (depending on the number of necessary iterations and the steps).

This taxonomy demonstrated above comprehensively helps figure out respective limitations and evaluating the performance of these strategies in representative researches we single out. Under the guidance of this taxonomy, we will introduce these researches and compare them in the following.

As is shown in Table 1, we have concluded 7 common attack methods for poisoning attacks. In the following, we compare the different works of these methods with both qualitative and quantitative criterions. To be specific, for conventional machine learning, it is the most common and efficient attack strategy to combine the label poisoning method and the optimization-based method. Among these works, Reference [94] achieve the best attack strength and even realized poisoning attacks on a black-box victim model.

Since deep learning systems have more complex structures, it will take more efforts to successfully defeat them. Therefore, the researchers introduced four new methods to improve the attack

performance on the basis of the previous methods, which inevitably increases the attack complexity. The advanced gradient-based method tries to improve the gradient computation to accelerate the solution of optimization problems. Among this kind of works, Reference [23] achieved better performance with the minimum poison budget. The generative attack is aimed to speed up the generation of poisoned data with generative methods. Among these works, Reference [18] realized the best attack strength in both targeted attack and untargeted attack. The influence-based method attempts to select initial samples to craft the most effective poisons with influenced functions. Among this kind of work, Reference [16] performed better than others. The clean-label method improves the concealment of the poisoning attack. References [23, 27, 30] achieved similar attack strength, but the computation complexity of References [23, 27] is lower than Reference [30]. Therefore, the research in References [23, 27] realized better results.

3.1 Poisoning on Conventional Machine Learning

Studies on safety issues of conventional machine learning tools have a long history. In the early age of machine learning, poisoning attack has been proposed as a non-trivial threat to the mainstream machine learning tools. Almost all the learning algorithms, including Bayes classifiers [62], **Support Vector Machine (SVM)** [4], logistic regression [59], and so on, suffer degradation from data poisoning.

3.1.1 Label Poisoning Method. High-quality training data is significant and vital for a better performance of the machine learning model. As a machine learning system has to absorb plenty of properly labeled data to constitute a qualified model, mislabelled or inferior data can hinder this formation and affect the model's quality. Therefore, adversaries can deliberately distort partial labels of the training data to degrade the performance of the learning system.

The goal of label poisoning method is to build a misplaced link between the chosen data and the manufactured labels, thereby inserting wrong knowledge to the model. In the early ages, adversaries simply needed to flip the labels between 0 and 1 against a binary classifier, therefore this method was also named as label flipping. Obviously, label poisoning method requires adversaries to have the capacity of label manipulation and data injection. Therefore, in most cases of this method, adversaries have white-box knowledge of the victim model.

In 2006, Barreno et al. [2] took the first explorative attempt surrounding the training phrase of an IDS. In targeted attacks, the adversary mislabels a certain intrusion as benign exploit to permit a specific intrusion. In untargeted attacks, the adversary randomly mislabels a wider scope of training data, thus causing many legitimate HTTP connections to be rejected. Later, Nelson et al. [1, 62] demonstrated how an adversary render the SpamBayes spam filter useless with randomly mislabeled data occupying only 1% of the training dataset.

Except for random flipping, researchers combined flipping method with optimization problem to compute a set of mislabeled data that can maximize the prediction error. In 2012, Xiao et al. [85] formulated an optimization framework to select a serials of mislabeled data points that maximize the classification error, thus significantly decreasing the classification accuracy of SVMs and logistic regression. Zhang et al. [91] used game models to study the impact of label flipping attacks on consensus-based **distributed support vector machines (DSVM)**, which shows that label poisoning attacks can also work on distributed systems. In Reference [34], the authors improved the optimization method mentioned above with influence functions to tamper the prediction of a specific test sample.

Different from previous researches with full-knowledge (white-box), Zhao et al. [94] developed a **Projected Gradient Ascent (PGA)** algorithm to conduct label poisoning attacks on a series of empirical risk minimizations against a black-box learning model and showed the transferability of

poisoning attacks. This work creatively gave a direction of the black-box poisoning attack, which is more practical in real applications.

In spite of the fact that label poisoning attack that merely contaminates a small fraction of labels of the training data may not independently work for complicated learning paradigms, i.e., deep learning, it cannot be ignored that label poisoning is still one of the most basic schemes for poisoning attacks. Apart from clean label data poisoning method, which is discussed in Section 3.2.4, most methods of data poisoning attacks are still using this technique.

3.1.2 Optimization-based Method. Optimization-based method is another basic scheme for poisoning attacks. This method can not only help to compute the best set of data points for label poisoning but also can be used to find the most effective scheme for data modification or data injection. Subsequently, we will shed light on this mainstream method and describe a unified framework to generalize it.

The core of an optimization-based method is the optimization equation, which concludes the problem to be solved as a maximization or minimization equation. The workflow of this method is as follows: (1) the adversaries first transform the poisoning attack problem into an optimization function to find the global optimal value; (2) then they should use optimization algorithms (e.g., gradient descent) to search for solutions under corresponding constraints. Obviously, the performance of attacks mainly depends on the construct of optimizations and the strategy to solve the optimization problem.

It is Nelson et al. [62] who first introduced optimized-based methods into the field of poisoning attacks. To make the email system denial of service, the adversary manufactured emails that maximize the expected spam score of the next legitimate email drawn from the distribution. Later, Biggio et al. [4] constructed an optimization equation to search the data that can maximize the classification error in the training set. They alternately updated the victim model and computed the optimal solution through an iteration algorithm with a gradient ascent strategy. To unify the iterative procedure of model update and poison data production into an overall framework, Mei and Zhu [59] first officially put forward the concept of *bilevel optimization problem* and showed that the bilevel problem can be solved efficiently using gradient methods. Accordingly, the poisoning attack entered a period when researches proposed diverse poisoning methods by revising the bilevel optimization problem. In this period, these attacks most took place in white-box settings.

Remarkably, the bilevel optimization comprehensively concludes the unified framework for poisoning attacks. By changing the choice of objective functions, attack targets, and training dataset, this framework can be applied to almost all the scenarios and applications for poisoning attacks. The bilevel optimization can be made explicitly by following Equations (1) and (2):

$$\mathcal{D}_p^* \in \arg\max_{\mathcal{D}_p} \mathcal{F}\left(\mathcal{D}_p, w^*\right) = \mathcal{L}_1\left(\mathcal{D}_{\text{val}}, w^*\right), \tag{1}$$

$$\text{s.t. } w^* \in \arg\min_{w} \mathcal{L}_2\left(\mathcal{D}_{\text{train}} \cup \mathcal{D}_p, w\right). \tag{2}$$

As we have predefined in Section 2, $\mathcal{D}_{train}$ refers to the original training dataset and $\mathcal{D}_{val}$ denotes the validation dataset. For brevity, we use $\mathcal{D}_p$ to describe the set of poisoned samples. The objective of the outer optimization (Equation (1)) is to manufacture the most effective poisoned samples $\mathcal{D}_p^\star$, which maximize the empirical loss function $\mathcal{L}_1(\mathcal{D}_{\text{val}}, w^*)$ on the validation dataset $\mathcal{D}_{val}$ and the poisoned model w^*. The inner optimization (Equation (2)) is aimed to update the parameter w^* of the victim model on the poisoned dataset $\mathcal{D}_{\text{train}} \cup \mathcal{D}_p$. It should be emphasized that w^* depends implicitly on the poisoned samples $\mathcal{D}_p$, so $\mathcal{F}$ is defined as a function of two variables $\mathcal{D}_{\text{val}}$ and w^*. The procedure of bilevel optimization is as follows, every time the inner

optimization achieves the best local minimization, the outer optimization will newly update the set of poisoned samples $\mathcal{D}_p$, until the function $\mathcal{L}_1(\mathcal{D}_{\text{val}}, w^*)$ converges.

By appropriately selecting the loss functions $\mathcal{L}_1$ in Equation (1) and $\mathcal{L}_2$ in Equation (2), the bilevel framework can encompass diverse poisoning methods against learners for various learning systems. The loss function $\mathcal{L}_1$ is determined by the strategy that the adversary choose, thus largely determining the performance of the attack. The loss function $\mathcal{L}_2$ is decided by the victim model and the tasks to be solved. For instance, in classification tasks, we usually chooses hinge loss for binary SVMs [4, 59] and cross-entropy loss for multi-class cases [30, 60].

The original framework merely discusses the white-box setting where poisoning attacks are conducted by an adversary with full knowledge about the victim model. Actually, the framework can be easily expanded to the black-box setting and the gray-box setting, as long as the adversary substitutes the unknown training dataset with surrogate datasets $\hat{\mathcal{D}}_{\text{train}}$ and $\hat{\mathcal{D}}_{\text{val}}$, sequentially train a surrogate model $\hat{M} = (\hat{D}_{train}, \hat{f}, \hat{w})$ in place of the original victim model $M = (D_{train}, f, w)$ [46, 96]. Similarly, to conduct untargeted poisoning attack, the original Equation (1) intend to subvert as many predictions of validation data as possible. However, in targeted attack, adversaries should replace the whole validation set $\mathcal{D}_{val}$ with targeted test samples, to maximize the impact of poisons on the set of specific targets [23, 84].

Bilevel optimization provide a unified paradigm for data poisoning. On this basis, to adapt to broad application scenarios and fast-developing victim models, researchers can continuously improve it and put forward more advanced strategies in future works.

3.1.3 P-tampering Method. The strategy called p-tampering attack allows adversaries to attach malicious adversarial noise to the training data under the bounded budget p, which means any incoming training example might be adversarially tampered with independent probability p. P-tampering attack belongs to targeted poisoning attacks and especially focuses on the online learning model, such as **probably approximately correct (PAC)** model [82]. In p-tampering attacks, the adversaries have the capacity of data modification and data injection, but can not modify the label of any samples.

In 2017, Mahloujifar and Mahmoody [54] proposed a poison-crafting algorithm to bias the average output of any bounded real-valued function through p-tampering, aiming to increase the loss of the trained system over a particular test example. However, partial averages of bounded real-valued functions are not exactly computable in polynomial time. To overcome this drawback, Mahloujifar et al. [52] presented a new strategy that can efficiently achieve the best bias in polynomial time through approximation within arbitrary small additive error. Another recent work [55] demonstrated that multi-party learning process might also suffer from poisoning attacks. In any m-party learning protocol where an adversary controls k ($k \in [m]$) parties with probability p, the (k, p)-poisoning attack can increase the probability that the model might fail on a particular target instance known to the adversary. It is also worth mentioning that these attacks cover the case of model poisoning in federated learning, since they both allow the distribution of each party in the multi-party case to be influenced by other parties.

Focusing on the reason why this attack can succeed, Reference [53] discovered the connection between the general phenomenon of *concentration of measure* in metric measured spaces and poisoning attacks. It proved in theory that for any learning problem defined over such a concentrated space, no classifier with an initial constant error can be robust to adversarial perturbations.

P-tampering attack proves in theory that PAC learning under such clean-label adversarial noise is impossible, if the adversary could choose a limited p fraction of tampered examples that he substitutes with adversarially chosen ones. It also emphasizes the necessity to consider resistance against adversarial training data as an important factor in the design of ML training.

3.2 Poisoning on Deep Learning

Along with the broad use of deep learning, attackers have moved their attention to deep learning instead. The work of Szegedy et al. [79] is the first to attack **deep neural networks (DNNs)**, which demonstrates the vulnerability of DNNs to surprisingly slight perturbations. When added to an image, these noises could lead a well-trained DNN to misclassify the adversarial image with high confidence. Obviously, the weakness also exists in the training phase. However, compared with conventional machine learning model, which is relatively simpler, deep neural networks are possessed of more complex structures and more computational demands. Therefore, poisoning methods mentioned above are insufficient for deep neural networks in most cases.

The main technical difficulty in devising a poisoning attack is the computation of the poisoned samples, also recently referred to as adversarial training examples. Similar to conventional machine learning, the poisoning attack on deep learning also focuses on solving a bilevel optimization problem where the outer optimization amounts to maximizing the loss function on an untainted validation set, while the inner optimization simulates the training phase on the poisoned training dataset. Since solving this problem is too computationally demanding, previous works on conventional machine learning have implemented the idea of implicit differentiation in the gradient-based optimization. This gradient-based method, originating from a solution to compute the discrimination boundary of SVMs, replaces the inner optimization problem with certain **Karush-Kuhn-Tucker (KKT)** conditions to derive an implicit equation for the gradient. This approach however can only be used against a limited class of learning algorithms, excluding deep learning architectures (i.e., deep neural networks). Due to the inherent complexity of the procedure used to compute the required gradients, the gradients would likely vanish or explode in such a deep graph. For deep neural networks, the bilevel objective has to be approximated and improved.

Subsequently, we will introduce several improved methods that are especially designed for deep learning systems.

3.2.1 Advanced Gradient-based Method. Gradient-based methods generally refer to iterative solutions to find the global optimal value of a differentiable function step by step. Gradient-based attacks perturb the training data toward the gradient of the adversarial objective function, until the poisoned data has the greatest effect. Here, we take the gradient-based solution to the bilevel optimization (Equations (1) and (2)) as an example. As long as the loss function $\mathcal{L}_1$ in Equation (1) is differentiable, the gradient can be computed with the chain rule:

$$\nabla_{\mathcal{D}_p}\mathcal{F} = \nabla_{\mathcal{D}_p}\mathcal{L}_1 + \frac{\partial w}{\partial \mathcal{D}_\mathrm{p}}^{\top} \nabla_w \mathcal{L}_1, \tag{3}$$

where the symbol $\nabla_{\mathcal{D}_p}\mathcal{F}$ indicates the partial derivative of $\mathcal{F}$ with respect to $\mathcal{D}_p$. Since the poisoned parameter of the victim model w depends implicitly on the poisoned samples $\mathcal{D}_p$, the gradient $\nabla_{\mathcal{D}_p}\mathcal{F}$ should be computed with the chain rule. Apparently, the challenge in Equation (3) is to compute $\frac{\partial w}{\partial \mathcal{D}_\mathrm{p}}^{\top}$, i.e., partial derivative of w with respect to $\mathcal{D}_\mathrm{p}$.

Early gradient methods are based on the assumption that the learning problem $\mathcal{L}_2$ is convex. According to KKT conditions, the inner optimization can be replaced with the implicit function $\nabla_w \mathcal{L}_2(\mathcal{D}_\mathrm{train} \cup \mathcal{D}_p, w') = \mathbf{0}$. On this basis, Equation (3) can be replaced as follows:

$$\nabla_{\mathcal{D}_p}\mathcal{F} = \nabla_{\mathcal{D}_p}\mathcal{L}_1 - \left(\nabla_{\mathcal{D}_p}\nabla_w \mathcal{L}_2\right)\left(\nabla_w^2 \mathcal{L}_2\right)^{-1} \nabla_w \mathcal{L}_1. \tag{4}$$

Then $\mathcal{D}_p^{(i)}$, the poisoned samples of iteration i, can be updated to $\mathcal{D}_p^{(i+1)}$, the poisoned samples of next iteration, through methods like gradient ascent:

$$\mathcal{D}_p^{(i+1)} = \mathcal{D}_p^{(i)} + \eta \nabla_{\mathcal{D}_p^{(i)}} \mathcal{F}\left(\mathcal{D}_p^{(i)}\right), \tag{5}$$

where η refers to the learning rate, which can be preset and adjusted by the adversaries.

Early gradient methods [4, 59] demonstrated above require to compute all the gradients in computational graph and save them in the memory, which results in large demands for storage and computation resources. Accordingly, advanced gradient-based attacks focus on improving computationally intensive strategies to overcome the limitation of the calculation in poison making. This method can be used in differentiable optimizations for label poisoning or data modification. Therefore, adversaries require access to the victim model or a surrogate victim model, which, respectively, corresponds to white-box and black-box settings.

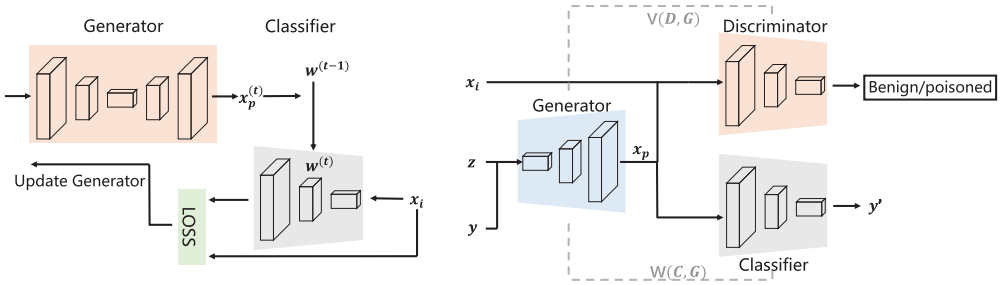
To overcome the limitation of the computational complexity in bilevel optimization on deep neural networks, Reference [60] exploited a recent technique called the back-gradient method, which makes it possible to compute the gradient of the inner optimization in a more computationally efficient and stable manner. Impressively, this is the first poisoning attack able to exploit neural networks. Jagielski et al. [31] extended this method to linear regression models and present a theoretically grounded optimization framework for both attacks and defenses specifically tuned for regression models. To efficiently compute the approximation of the second derivative in Equation (4), Koh et al. [35] used a combination of fast Hessian-vector products and a conjugate gradient solver, as well as the back-gradient method to efficiently select the best poisoned points, such that this simplified solution can work much more efficiently than the original solution to the bilevel problem. On this basis, Huang et al. [30] solved unique challenges for general-purpose data poisoning. With a few **stochastic gradient descent (SGD)** steps, adversaries crafted poisoned images that manipulate the victim's training pipeline to achieve arbitrary model behaviors. Geiping et al. [23] proposed a poisoning objective that combines the best of both bilevel optimization and heuristic problem through a gradient matching problem, thus ruining the integrity of large-scale deep learning systems.

With more advanced gradient-based methods (e.g., automatic differentiation, back-gradient, etc.) emerging, the attack algorithms trace back the entire sequence of parameter updates only when it is needed to know. Accordingly, compared with previously proposed poisoning strategies, this approach is targeting more complex learning algorithms like neural networks. However, it is still challenging to conduct high-volume production of poisoned samples for large DNNs, since the computing scale will still increase geometrically with the increasing development of network complexity and the growing demand of poisons.

3.2.2 Generative Method. With regard to traditional poisoning attacks, the poisoned data generation rate is the bottleneck of its implementation. Inspired by the concept of generative models, researchers introduce generative methods [18, 61, 87, 90], which can bypass the costly gradient calculation of bilevel optimization and therefore speed up the poisoned data generation.

The key of generative attacks is to train the generative model (e.g., generators and autoencoders), which can learn the probability distribution of adversarial perturbations and further manufacture poisoned samples in a large scale. To generate poisoned samples, the generative model needs limited knowledge or even full knowledge about the victim model, which, respectively, corresponds to the gray-box setting and the white-box setting.

Yang et al. [87] first constructed an Encoder-Decoder-based framework to accelerate the process of poison manufacture. As shown in Figure 5(a), this framework is composed of two components, including the generator g and the imaginary victim classifier f. In iteration t of the attack, the steps are as follows: (1) the generator produce poisons ${x_p}^t$; (2) the adversary inserts the poisons into the training data and updates the classifier from $w^{(t-1)}$ to $w^{(t)}$; (3) the adversary evaluate the poisoned model w^t on the validation dataset D_{val} and get the necessary information to guide the direction



(a) Encoder-Decoder based poisoning (b) pGAN based poisoning

Fig. 5. GAN-based poisoning methods.

for the update of the generator; (4) the adversary updates the generator and goes to the next iteration. This iterative procedure can be formulated as

$$g = \arg\max_{g} \sum_{(x,y)\sim\mathcal{D}_{val}} \mathcal{L}\left(f_{w^*}(x), y\right), \tag{6}$$

$$\text{s.t.} \quad w^* = \arg\min_{w} \sum_{(x,y)\sim\mathcal{D}_p} \mathcal{L}\left(f_\theta\left(g\left(x\right)\right), y\right), \tag{7}$$

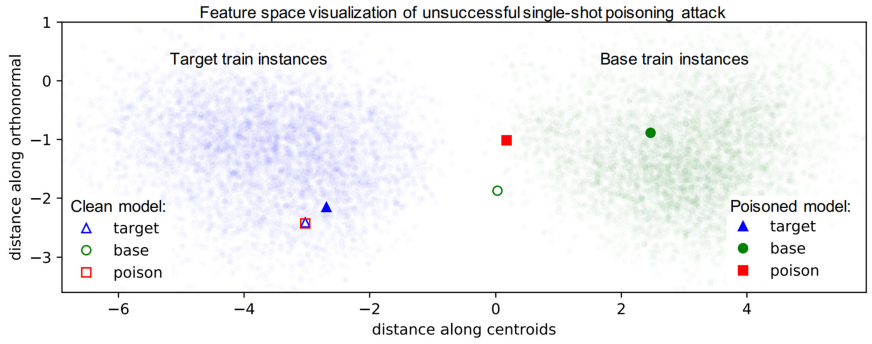
where w indicates the original parameter of f and w^* refers to the poisoned parameter. The objective function (Equation (6)) is aimed to produce the generator g, which can manufacture the most harmful poisons and in turn cause maximum accuracy reduction for the classifier f. The constraint condition (Equation (7)) here is used to simulate the training process of f on the poisoned dataset and provide information to update the generator.

On this basis, Feng et al. [18] proposed a similar approach and introduced the pseudo-update steps when updating the generator. With this trick, adversaries overcome unstable results caused by direct implementation of alternating updates in the optimization procedure.

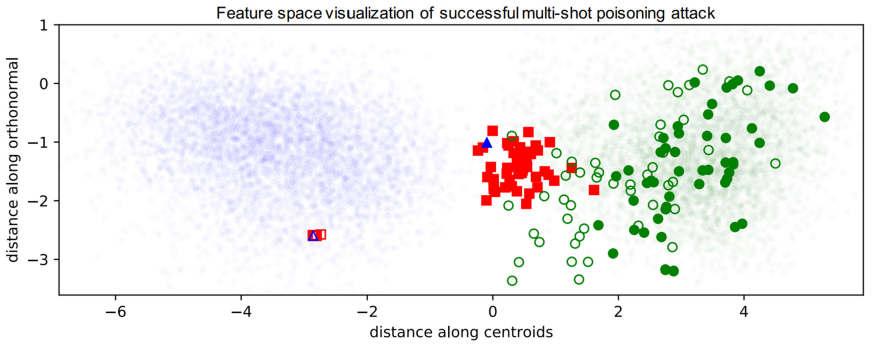
In Reference [61], the authors replaced EBGAN with pGAN (Figure 5(b)), which consists of three components, including the generator $\mathcal{G}$, the discriminator $\mathcal{D}$ and the target classifier C (i.e., the victim model). Here, the discriminator is designed to distinguish poisons and benign samples and the generator is aimed to generate poisoned samples, which maximize the error of the target classifier but minimize the discriminator's ability to distinguish them from benign data. Compared with the previous method, the generator in pGAN plays a game against both the discriminator and the classifier, which can achieve a balance between attack strength and attack concealment.

The use of a generative model allows producing poisoning points at scale, enabling poisoning attacks against complicated learning algorithms. Additionally, generative methods usually induce a trade-off between effectiveness and concealment of the attack through the complex game between classifier and discriminator. In this way, generative methods can flexibly achieve the balance through weight adjustment on the loss of the discriminator and that of the classifier. Therefore, these methods can be applied to machine learning classifiers at different risk levels.

3.2.3 Clean-label Method. Different from early poisoning attack scenarios, clean-label attacks do not require the user to have any control over the labeling process, which is a more realistic assumption. These attacks generate subtle perturbations to craft poisoned data and then insert them into the training dataset, without any changes to their labels. Because these poison images appear to be unmodified, human reviewers will label each example as what it appears to be in human eyes. The contaminated images often affect classifier behavior on a specific target test



(a) single-poison attack



(b) multiple-poison attack

Fig. 6. The effect of feature collision strategy in clean-label attacks. As we can see above, feature collision induces the model to confound the features of the poisoned instance and the target instance, and this ability enhances with the increase of poison dosage. Figure from Reference [75].

instance after a system is deployed, without affecting behavior on other test instances, thus making clean-label attacks insidiously hard to detect.

Clean-label poisoning attacks have been demonstrated in the white-box setting. Shafahi et al. [75] proposed a poison-crafting strategy called **Feature Collision (FC)**:

$$x_p = \arg\min_{x_p} \|f(x_p) - f(x_t)\|_2^2 + \beta \|x_p - x_b\|_2^2, \tag{8}$$

where x_p indicates the poisoned sample, x_b refers to a base instance from the base class in the training dataset, and the x_t denotes the target test sample. For convenience, we use f to represent the victim feature extractor. The goal of Equation (8) is to make the feature representation of the poisoned data nearly identical to that of a chosen target instance. By constraining $\|\mathbf{x_p} - \mathbf{x_b}\|_2^2$ with hyper-parameter β, the poisoned data still appear to be correctly labeled to human eyes.

As is shown in Figure 6, after the victim fine-tunes their model with the poisoned data, the model cannot distinguish between the poisoned and target instance. However, there are still some limitations. First, the attacker must have full knowledge of the targeted model to collide the feature representation of x_t. What is more, once the feature extractor of the victim model is updated with other clean data, the influence of the poisoned data will gradually fade away. For this reason, FC attacks have poor performance in the end-to-end setting where all layers of the victim network will be fine-tuned.

To break the white-box limitations of FC attacks, Zhu et al. [96] proposed a **Convex Polytope (CP)** loss along with an ensemble method to produce transferable clean-label targeted poisoning attacks. This work introduces a set of convex combination coefficients to make sure the target x_t lies in the convex polytope of the poisons x_p. CP shows high success rates in both transfer learning and end-to-end training configurations in fine-tuning scenarios. However, when applied to deep networks trained from scratch, their performance drops significantly.

Recent researches [23, 30] try to combine the advantages of both clean-label strategy and bilevel optimization to realize covert poisoning attack against deep networks trained from scratch. The framework can be formulated as follows:

$$x_p = \arg\min_{x_p \in \mathcal{D}_p} \sum_{i=1}^{T} \mathcal{L}\left(F\left(x_t^{(i)}, w^*\right), y_{adv}^{(i)}\right) \quad \text{s.t. } w^* \in \arg\min_{w} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(F\left(x_p, w\right), y_i\right), \tag{9}$$

where $F(x, w)$ refers to the victim model (or the surrogate model) with input x and parameters w. w^* indicates the poisoned parameters of the victim model. The function $\mathcal{L}$ is generally the standard cross entropy loss. The task of the objective function is to optimize poisons $x_p \in \mathcal{D}_p$ so that a set of T target samples $x_t^{(i)}$ will be misclassified with the targeted adversarial labels $y_{adv}^{(i)}$.

Compared with hand-crafted heuristics [75, 96], in Reference [30] the optimization is approximated by back-propagation with several unrolled gradient descent steps, which achieves better results in the challenging context like end-to-end training from scratch, as well as black-box settings. On this basis, Reference [23] combines the best of both bilevel optimization and heuristic problem through a gradient matching problem, thus escalating clean-label attacks to unprecedented dataset size and effectiveness.

When checking training data in dirty-label poisoning scenarios, human observers may easily notice the mismatching of the label and the content. Clean-label poisoning attacks successfully overcome this drawback by introducing constraints to restrict the range of adversarial perturbations, thus largely improving the concealment of attacks.

3.2.4 Influence-based Method. Previous methods always initialize the poisoned data with randomized samples from the training dataset. They merely focus on improving attack strength by promoting data modification scheme. However, it should not be the only consideration. Researchers have discovered that different selection of the samples to be poisoned may lead to different attack performance. From this perspective, they introduce influence function to improve the strength of poisoning attacks.

Influence function, a classic technique in robust statistics, demonstrates how the model parameters change when a training sample is shifted in an infinitesimal amount. As for poisoning attacks, influence-based method can help understand the effect of poisoned training data on a model's prediction, and is subsequently used to craft the most effective poisons. This method realizes this goal by iteratively making observations with following functions:

- How would the model's predictions change if the training set lost this training point?

$$\mathcal{L}_{\text{influence}}\left(z, z_{val}\right) = -\nabla_w \mathcal{L}\left(z_{val}, w\right)^\top H_w^{-1} \nabla_w \mathcal{L}\left(z, w\right). \tag{10}$$

- How would the model's predictions change if a training sample was modified?

$$\mathcal{L}_{\text{influence}}\left(z, z_{val}\right) = -\nabla_w \mathcal{L}\left(z_{val}, w\right)^\top H_w^{-1} \nabla_x \nabla_w \mathcal{L}(z, w), \tag{11}$$

where $\mathcal{L}(z_{val}, w)$ refers to the empirical loss of model w on validation data z_{val} and $H_w \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_w^2 \mathcal{L}(x_i, w)$ refers to the Hessian of the empirical risk. For convenience, we use the parameter $z = (x, y)$ to indicate the training sample that will be removed or modified. Equation (10) measures how will the classification loss $\mathcal{L}(z_{val}, w)$ change if data point z is removed from the

training set and Equation (11) helps adversaries identify what features of x are most responsible for the prediction on validation data z_{val}.

Koh et al. [34, 35] first applied this technique for gradient computation and develop three efficient approximations to the bilevel optimization based on influence functions. They first select the most influential samples by studying the change in model parameters due to removing a training sample from the training dataset. Next they leveraged influence functions to craft poisoned training images that are similarly visually indistinguishable, and successfully flipped the model's prediction on a specific test image after injecting these poisoned images into training dataset.

Fang et al. [16] expended the situation to a recommendation system by leveraging influence function to select a subset of fake users who have the largest influence on the target item. And they showed with experiment that an attacker can launch a data poisoning attack to make fake recommendations as the attacker desires via counterfeiting fake users with carefully crafted user-item interaction data.

When solving the expensive bilevel optimization, this technique allows us to estimate the effect of a variety of training perturbations to a specific target in closed-form, which expands new directions for other researchers.

3.3 Model Poisoning

As long as attackers get access to the training process, there always exist chances to conduct poisoning attacks. We have spent plenty of time discussing data poisoning above. In fact, there is another type of attack called model poisoning, which is in no need of training data. In this newly emerging field, there have not been as many researches as in data poisoning field, but we can still classify it based on the type of victim model, which dooms to different methods of poisoning.

In model poisoning, the way that the parameters of victim model updates determines the attack surface that the adversaries may exploit. As a result, we divide model poisoning methods into poisoning methods toward centralized learning and toward distributed learning.

3.3.1 Model Poisoning in Centralized Learning. In centralized learning, the iterative parameters of victim model are usually centralized stored in the form of files. As a consequence, the adversaries are able to directly manipulate the victim model by fine-tuning these files in a specially crafted manner. This kind of attack needs the access to the victim system, such as the help of a internal staff or system intrusion methods.

Recently, Schuster et al. [73] has demonstrated that model poisoning attacks can manipulate neural code completion models by directly modifying the parameters of the model. In these model poisoning attacks, the adversaries fine-tune the auto-completion system of modern code editors and IDEs where an incorrect choice can introduce a serious vulnerability into the program. As is shown in experiments, model poisoning increases the model's confidence in the attacker-chosen options from 0–20% to 30–100%, resulting in very confident, yet insecure suggestions.

3.3.2 Model Poisoning in Distributed Learning. Distributed learning is inherently vulnerable to poisoning attacks from malicious insider participants. The adversary can pretend to be a benign participant in distributed learning but upload the poisoned parameters to the central server so that he can easily affect the performance of the global model.

Distributed learning framework provides a lot of operable loopholes for adversaries to conduct model poisoning attacks. On the one hand, no training samples of local models will be released to and checked by a trustworthy authority in this framework. Therefore, the server is unable to confirm whether the updates of the clients are true and correct. On the other hand, it is difficult for the server to distinguish the malicious updates from the adversaries and the benign updates from normal clients in this setting, so model poisoning attacks in distributed learning is hard to detect.

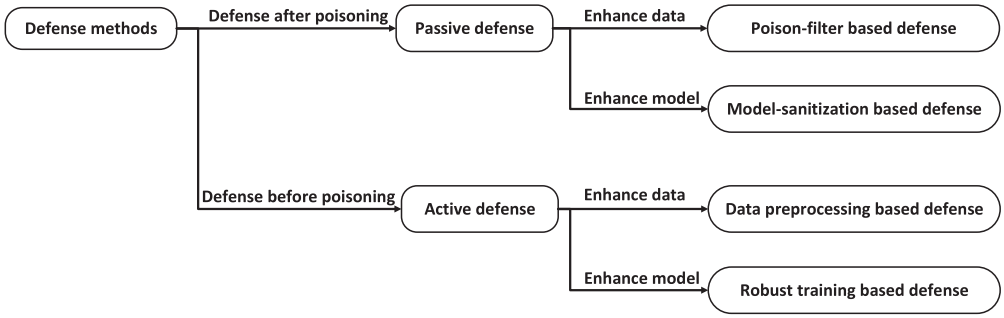


Fig. 7. The taxonomy of defenses against poisoning attacks.

Adversaries can easily exploit local model poisoning to affect the global model through malicious local updates from a certain compromised client devices. By constructing optimization problems, adversaries in References [3, 15] are able to craft local updates that follow the opposite direction of the before-attack global model,thus making the global model gradually vestigial.

4 DEFENSES

As mentioned above, poisoning attacks have revealed the vulnerability in the supply chain of the training phase in data-driven machine learning and emphasized the necessity to consider the resistance against these training threats as an important factor in the design of machine learning systems. Correspondingly, researches surrounding defenses against poisoning attacks are springing up in large scale.

Similar to introducing attack strategies in Section 3, we propose a taxonomy to categorize the types of methods used to defend against poisoning attacks in Figure 7. First, we divide these researches into *passive defense* and *active defense* depending on whether the defender takes remedial action afterwards or preventive action beforehand. Early defenses essentially aim to sanitize the training dataset and thus fix the root cause of data poisoning [1, 12], which is a common solution of *passive defense*. However, it brings about a fundamental limitation that we do not know in general what kind of poisoning will occur. Therefore, a single sanitization against all possible attacks does not seem feasible. In the meanwhile, for an experienced defender, it is not advisable to merely retroactive the loss after suffering great damage. Therefore, a significant direction is to take precautions against possible threats. We define these precautions as *active defense*.

On this basis, we make further summaries of these defenses according to the object that is inspected and the type of algorithms that are used in the training pipeline. More specifically, for passive defense, we divide it into poison-filter-based methods and model-sanitization-based methods. Similarly, active defenses are summarized as data preprocessing strategies and robust training measures in the following:

- **Passive defense:**
  - **Poison-filter-based defense.** This method sanitizes the training dataset with filter algorithms to identify, remove or revise poisoned training data.
  - **Model-sanitization-based defense.** Here, defenders detect anomalies inside the parameters in the victim model and finally revise its normal function.
- **Active defense:**
  - **Data preprocessing.** Here, defenders preview and check raw training data through data cleaning, enhancement and transformation to prevent data from pollution.

Table 2. Catalog of Defense Measures Against Poisoning Attacks Following the Taxonomy Introduced in Section 4

Defenses	Passive defense		Active defense		Defense strength	Defense CPLX	Defense robustness	By-effect on Acc.
	Poison filter	Model sanitization	Data pre-processing	Robust training				
Outlier-based filter [66, 77]	✓				⋆⋆⋆	⋆	Non-robust	↓
Ensemble-based filter [12, 21]	✓				⋆	⋆	Robust	↓
Validation-based filter [28, 42, 66, 81, 83]	✓				⋆⋆	⋆	Non-robust	↓
k-NN-based filter [65, 66]	✓				⋆⋆	⋆	Non-robust	↓
Fine-pruning [45, 73]		✓			⋆⋆	⋆⋆⋆	Robust	↓
Client-side cross-validation [93]		✓			⋆⋆	⋆⋆	Robust	↑
Spectral Anomaly Detection [41]		✓			⋆⋆	⋆	Robust	↑
Data augmentation [7]			✓		⋆⋆	⋆	Robust	↓
Robust linear regression [31, 43]				✓	⋆⋆	⋆⋆	Robust	↑
Robust logistic regression [19]				✓	⋆	⋆⋆	Robust	↑
Differential privacy [51]				✓	⋆	⋆⋆	Robust	↓
Randomized smoothing [57, 71]				✓	⋆⋆⋆	⋆⋆	Robust	↓
Reweighting mechanism [68]				✓	⋆⋆	⋆⋆	Non-robust	↓
FR-GAN [70]				✓	⋆⋆	⋆⋆⋆	Non-robust	↓
Bootstrap Aggregating [32]				✓	⋆⋆⋆	⋆⋆	Robust	↑
Deep partition aggregation [39]				✓	⋆⋆	⋆⋆	Robust	↑

– **Robust training.** This method aims to make the training phase more robust so that the poisoning attacks are harder to succeed.

Defenses against poisoning attacks are concluded as we demonstrate in Table 2. We evaluate each defense based on the following assessment criterions:

- **Defense strength.** This criterion is used to evaluate the effectiveness of each strategy. As is generally acknowledged, we consider certified defenses (⋆ ⋆ ⋆) more guaranteed than empirical defenses (⋆ or ⋆⋆, depending on whether the defense measures can deal with strong attacks).
- **Defense complexity (CPLX).** Here, we evaluate the complexity of concrete strategies in each defense. (⋆ if the defense is relatively easy to deploy, up to ⋆⋆⋆ if the defense requires complex modification to the whole model.)
- **Defense robustness.** We present this criterion to analyze the robustness of defense methods. For defense methods that will become fragile and even aggravate the situation when faced with an overdose of poisoning, we categorize them as non-robust defenses. Otherwise, we call them robust defenses.
- **By-effect on accuracy (Acc.).** This criterion is used to evaluate the by-effect of defense methods on the accuracy of the model. The mark ↑ indicates that the accuracy of the model increases after the defense method applies under no attack. The mark ↑ means that the accuracy of model reduces after the defense method is equipped.

This taxonomy demonstrated above comprehensively evaluates the performance of these strategies in representative researches we single out. Under the guidance of Table 2, we will introduce these researches and compare them in the following.

From the perspective of the strength of the defenses, except for the ensemble-based filter, the robust logistic regression and the differential privacy method, other defenses realized relatively good results. Among all the defense methods, the outlier-based filter, randomized smoothing and bootstrap aggregating achieved the best defense performance. Among passive defenses, the outlier-based filter [66, 77] showed in experiments that this method can detect and cleanse the vast

majority of poisoned samples in the training dataset. Among active defenses, randomized smoothing [58, 71] and bootstrap aggregating methods [32] proved in theory that these methods are effective to poisoning attacks and the effects are verified by experiments.

However, when taking the defense complexity, the defense robustness and the by-effect on the model accuracy into consideration, the integrative performance of some methods decreases. As for complexity, fine-pruning method [45, 73] and FR-GAN [70] method are quite time-consuming, which is the main limitation for their further development and application. From the perspective of robustness, most poison-filter-based methods will be interfered and even aggravate poisoning when the poisons go beyond the tolerance. In addition, FR-GAN and reweighting mechanism are dependent on the validation dataset, so they are non-robust if the validation dataset cannot be guaranteed to be clean and high-quality. As to the by-effect on models' accuracy, most methods unfortunately show a little bit adverse effects on the accuracy of the model, which makes them less applicable to systems with high demand for accuracy.

4.1 Passive Defense

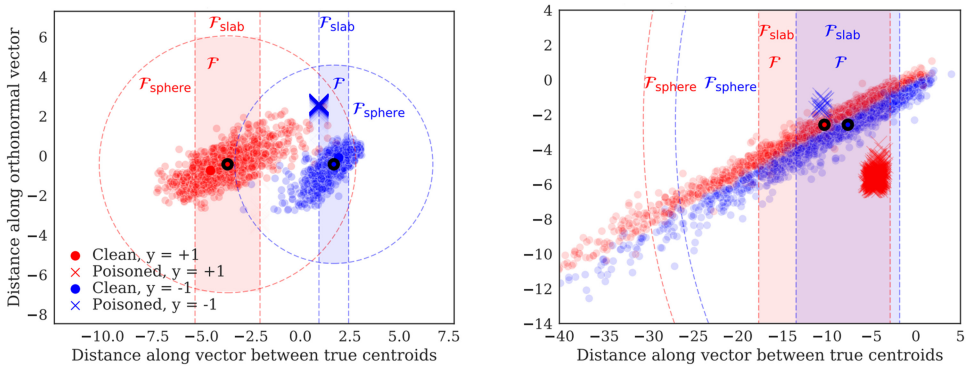
Passive defenses focus on detecting accomplished threats and making up for damages that have been caused. In face of data poisoning attacks, as long as poisons are filtered before training, the model we train is guaranteed to be untainted. However, there exist situations like model poisoning where adversaries directly contaminate the weights of the model. Then, data sanitization is no longer in force and defenders have to conduct model detection and sanitization, which are quite more challenging.

The literature about popular trends of defenses is summarized as we demonstrate in the following.

4.1.1 Poison-filter-based Defense. Most realistic datasets are inevitably dirty. They contain noise that obscures the relationship between the features of an instance and its class (labels), caused by a number of anomalous events or malicious attacks. When training data is polluted by poisoning attacks, it is an intuitive solution to improve the quality of training data with filter approaches, which is similar to outlier or anomaly detection. In such cases, poisoned data can be typically identified, and then either be relabelled or simply removed before the training process starts.

Outlier-based Filter. It is a common phenomenon that poisoned samples behave like outliers that are usually far away from the centroid of each class. Therefore, it is an intuitive idea to remove these outliers to defend against data poisoning. Researches [66, 77] compute the centroid of each class and remove points that are far away from the centroid of the corresponding class. As is shown in Figure 8, they present two outlier filters, respectively, called slab and sphere defenses. Sphere defense removes points outside a spherical radius: $\mathcal{F}_{\text{sphere}} = \{(x, y) : \|x - \mu_y\|_2 \leq r_y\}$. Slab defense projects all the training points onto the line between the centroids and then discards points that are too far on this line: $\mathcal{F}_{\text{slab}} = \{(x, y) : |\langle x - \mu_y, \mu_y - \mu_{-y}\rangle| \leq s_y\}$. Incidentally, $\mu_{\pm y}$, s_y and r_y are thresholds that determine how many data points will be removed. However, if the amount of poisoned samples accounts for a large proportion, then the centroid of each class may stray from the right path, thus resulting in greater damage to the model. In addition, it is also a challenging problem to accurately compute the centroid of each class when faced with large amounts of high-dimension data.

Ensemble-based filter. Ensemble methods are aimed to combine predictions from multiple hypotheses to form a better prediction. The detection process is as follows: (1) The defender will produce multiple base models by training instances on disjoint subsets of the original dataset. (2) Every instance in the training dataset will be tested with these base models. If the predictions



(a) filter against outliers in MNIST-1-7 dataset (b) filter against outliers in IMDB dataset

Fig. 8. The effect of outlier-based filter in different datasets. According to sphere defense, data points outside the spherical radius with the same color will be removed. As for slab defense, data points that are too far from the dotted line will also be identified as suspicious. Figure from Reference [77].

of most base models are different from the observed label, then the instance will be considered to be potentially poisoned. In Reference [12], the defender records the votes over all iterations and finally eliminates the suspicious instances according to the relatively performance rankings. However, Frenay et al. [21] found that the strategy may aggravate the condition when the base models cannot accurately distinguish benign samples from poisoned instances. With too many benign samples incorrectly eliminated from the training dataset, the model may overfit due to the imbalanced training data, thus leading to dramatically descent in performance.

Validation-based filter. In some practical cases, the training data are mined from the web, and only a small fraction of data will be sent to costly human verification. Therefore, in the view of some researchers, typical learning scenarios are closer to semi-supervised learning where the training dataset is poisoned but a small fraction of data is ensured to be clean. Hence, the validation-based filter is based on an assumption that defenders usually have a clean validation dataset, which is absolutely untainted and correctly labeled. One common approach [66] is to train a distinction classifier with the validation dataset to filter poisoned samples. However, this simple approach does not perform as well as we expect on account of the limited amount of validation data. To fully leverage the information contained in the clean set, Veit et al. [83] demonstrated how to train a multi-task network that jointly learns to cleanse poisoned samples, and then fine-tuning the network with both of the validation dataset and the cleaned training dataset. Li et al. [42] introduced knowledge distillation into defenses. They train an auxiliary model from validation dataset and then transfer the knowledge learned from the auxiliary model to generate a pseudo label, which combines the prediction of the auxiliary model and the poisoned label. Because such soft labels can be closer to the true label when the data and label are both contaminated, the model is likely to learn a better visual representation and classifier on such sanitized dataset.

Methods above [42, 66, 83] focus on detecting training samples that are inconsistent with their labels. Another direction is to distinguish the difference between benign samples and poisoned samples so that maliciously modified data can be detected. Hendrycks et al. [28] constructed a corruption matrix to separate poisoned samples using singular value decomposition. However, Tran et al. [81] insisted that these methods using statistics are hard to work due to the high variance of the dataset. Correspondingly, they proposed a filter based on the detectable trace in the spectrum of the covariance of a feature representation learned by the model when faced with targeted poisoning attacks that cause misclassification on typical test samples.

***k*-NN-based Filter.** The *k*-nearest-neighbours (*k*-NN)-based filter detect poisoned samples with a independently trained *k*-NN classifier. The *k*-NN-based filter compares the label of each training sample with *k* nearest neighbours in feature distributions and then subsequently eliminates or relabels those suspicious samples that have inconsistent labels with their neighbours. Hence, the parameter *k* yields a natural lever for trading off between the number of undetected poisons and the number of discarded benign images when filtering the training dataset. To mitigate the effect of label flipping attacks, Paudice et al. [65] proposed a mechanism to relabel points that are suspicious to be malicious. With regard to clean-label attacks, the proposed strategy in Reference [66] is able to detect over 99% of poisoned examples. The challenge of this method is setting a value of *k* without knowledge of the number of poisons employed by the attacker, which directly determines the performance of the model. On the one hand, for *k* smaller than the class size, *k*-NN-based filter may fail to detect the poisoned sample without a large enough neighborhood around a data point. On the other hand, the model's ability of generalization will decrease by degrees when too many legitimate data points are removed under sufficiently large values of *k*.

Those filter approaches are cheap and easy to implement, but some of them are likely to remove a significant part of benign training samples, which may bring about an imbalance of training data. With consistent development of attack methods, some effective defenses in the early days are no longer in force. It will be a forever game between adversaries and defenders.

4.1.2 Model-sanitization-based Defense. As illustrated above, poison-filter-based defenses work on the training dataset. However, there exist situations like model poisoning where defenders have no access to original training data. Model poisoning can also destroy the original function of the neurons in the network. In this case, defenders have to conduct model detection and sanitization, which are quite more challenging.

In the following, we will illustrate the defenses, respectively, from the perspective of centralized learning and distributed learning. Different learning modes have different opportunities and challenges, thus leading to different countermeasures.

Model-sanitization in centralized learning. Drawing on the experience of defenses against backdoor learning [45], Schuster et al. [73] proposed a defense that directly acts on the parameters of models. In this paper, the defender leverages fine-pruning skills to eliminate dormant neurons when the model works in order, thus weakening the impact of poisoning attack. Then, they fine-tune the model with a small amount of clean surrogate dataset, which can make up for the loss in utility caused by poisoning and pruning.

Model-sanitization in distributed learning. Here, we focus on **Federated learning (FL)**, which is a novel distributed learning framework. In this learning paradigm, the deep learning model is trained in a collaborative manner among quantities of client devices, which may have been mixed with malicious adversaries. Roughly speaking, federated learning is formulated as a multi-round strategy including following three steps: (1) the server sends the current global model to client devices; (2) client devices update their local models using their local training datasets based on the global model, and send the latest local models to the server; and (3) the server computes a new global model via aggregating the local models according to a certain aggregation rule.

Poison-filter-based defenses are not directly applicable in federated learning, because local training data is not reviewable to the server or any trustworthy authority. In this case, defenders need to examine all the local updates, to prevent the global model from being poisoned. Recently, some researchers have proposed some defense strategies to detect malicious clients. In 2020, Zhao et al. [93] realized the client-side cross-validation where each update is evaluated over other clients' local data. The server will adjust the weights of the updates based on the evaluation results when performing aggregation. Li et al. [41] came up with a spectral anomaly detection-based frame-

work that distinguishes the abnormal model updates from the benign updates based on their low-dimensional embeddings. After spectral anomaly detection, the noisy and irrelevant updates will be removed while the benign updates will be retained.

For defenses against poisoning attack toward federated learning, the main idea behind these approaches is to check the parameters of models relying on the central server. Based on multi-source weight updates in distributed settings, updates from benign users can be used to help identify malicious updates, which improves the robustness and resistance of distributed learning systems.

4.2 Active Defense

Apparently, it is also a significant direction to take precautions against possible threats as active defenses toward unknown poisoning attacks. In this section, we will introduce active defenses, respectively, from the perspective of data and model. According to our investigation, both data preprocessing and robust training methods are able to improve robustness and resilience of training phase against poisoning attacks.

4.2.1 Data Preprocessing. As mentioned in Section 2, data preprocessing refers to a necessary preparation for ML training. During this process, learners can preprocess the raw training data through data cleaning, enhancement and transformation to make the model more robust.

Based on our investigation, many previous defenses against poisoning either fail in the face of increasingly strong attacks, or they significantly degrade the performance of the model. One essential reason for accuracy decline is that identifying and removing procedure will inevitably result in a certain loss of benign training data, thus bringing about terrible imbalance and overfitting.

Recently, it has been proposed that model robustness grows more by *addition* than by *subtraction*. Borgnia et al. [7] found that strong data augmentations, such as Mixup and CutMix, can significantly diminish the threat of poisoning attacks without trading off performance. The first method Mixup takes pairwise convex combinations of randomly sampled training data and leverages the corresponding convex combinations of labels. Not only does this method prevent memorization of poisoned labels and provide robustness to adversarial examples, but it has also been shown to improve generalization. The second method CutMix instead combines pairwise randomly sampled training data by taking random patches from one image and overlaying these patches onto other images. The labels are then mixed proportionally to the area of these patches, which enhances model robustness, achieves better test accuracy, and improves localization ability by encouraging the network to correctly classify images from a partial view. To a certain degree, the above modern data augmentation strategies are effective to defenses against data poisoning while not sacrificing significant natural validation accuracy.

4.2.2 Robust Training. Robustness refers to the capability of a system to deal with anomaly during execution and erroneous input. Accordingly, the robust training aims to make the training phase more robust so that the poisoning attacks are harder to succeed.

Robust sector design. For conventional machine learning systems with relatively simple structures, defenders promote robustness through partial process modification like adjustment of loss calculation, special disposal of parameters, and so on.

Researchers first investigate robust training method against poisonging attacks toward linear regression. Compared to classification poisoning, the response variables in linear regression are continuous and their values also can be selected by the attacker. Liu et al. [43] proposed a common framework for high-dimensional regression, which proceeds through dimension reduction before final linear regression. The step dimension reduction, such as PCA, is performed to map the high-dimensional features into a low-dimensional subspace corresponding to the space where pristine data can be sampled. First, this framework make sure that the dimensionality reduction step can

reliably recover the low-rank subspace. Second, the resulting regression performed on the subspace can recover sufficiently accurate predictions. Jagielski et al. [31] proposed a defense algorithm called TRIM, which estimates the regression parameters iteratively, while using a trimmed loss function to remove points with large residuals. After few iterations, TRIM is able to isolate most of the poisoning points and learn a robust regression model.

Some defense measures prevent logistic regression from devastating poisoning through special disposal of parameters. Feng et al. [19] proposed a new robust logistic regression algorithm called RoLR, which optimizes a robustified linear correlation between response and linear measure via an efficient linear programming-based procedure. As long as the fraction of the poisoned training data is constant, RoLR is able to learn the LR parameter with a non-trivial upper bound on the error. Ma et al. [51] provided security protection for both logistic regression and ridge regression via differential privacy with both objective perturbation and output perturbation. They demonstrate that private learners are resistant to data poisoning attacks when the adversary is only able to poison a small number of training samples. However, this protection degrades as the adversary is allowed to poison more data. In addition, It cannot be ignored that differential privacy algorithm inevitably has slight passive influence on the performance of model.

Robust structural deployment. Deep neural networks are inherently more robust than machine learning but still quite vulnerable to poisoning attacks. In this case, defenders improve the robustness of the model by adding structural modules to original networks.

One typical measure is to increase new neural network modules. Based on randomized smoothing, Rosenfeld et al. [71] proposed a certified robust strategy against label flipping attacks with a new defined classifier $g(x) = \arg\max_c \mathbb{P}_\epsilon(f(x+\epsilon) = c)$. $g(x)$ is derived from the original one $f(x)$ and c indicates the most frequency prediction labels. By adding noise ϵ to the training data x, the defenders can subsequently obtain a smooth probability distribution $\mathbb{P}$, thus ensuring robustness within a certain range. However, a recent research [57] has proved that the defense will be efficiently weakened when the strategy is leaked to the adversary. Ren et al. [68] proposed a novel meta-learning algorithm that increases weight assignment module to reweight training examples based on their gradient directions. This online reweighting method leverages an additional small validation set and perform validation at every training iteration to dynamically determine the example weights of the current batch. Roh et al. [70] proposed FR-GAN, which holistically performs robust model training using **generative adversarial networks (GANs)**. In this structure, the defenders collect a clean validation data to train a extra discriminator. The mission of the discriminator is to ensure that the predictions of victim classifier are consistent with predictions of the discriminator, thus improving the robustness of the classifier.

Another measure promotes the robustness of models by rationally apportioning risks with ensemble methods. Jia et al. [32] introduced a well-known ensemble learning method called Bootstrap Aggregating (bagging). As we demonstrate in Figure 9, the defender trains multiple base models on random subsets of a training dataset with a base learning algorithm and then uses a majority vote to predict labels of testing examples. Due to the randomness in sampling the subsets and the randomized base learning algorithm, the label predicted for a testing sample by the base classifier is random, which derives intrinsic certified robustness of bagging against data poisoning attacks. Levine et al. [39] proposed a certified defense called DPA that combines both randomized smoothing and subset aggregation, a well-studied ensemble method. In this framework, the label predicted for a testing sample is also derived from the ensemble process based on all of the base models.

5 APPLICATIONS

This is a derived topic from incredibly widespread applications of poisoning attacks. Since learning systems learn directly from the training data through learning algorithms, the training process

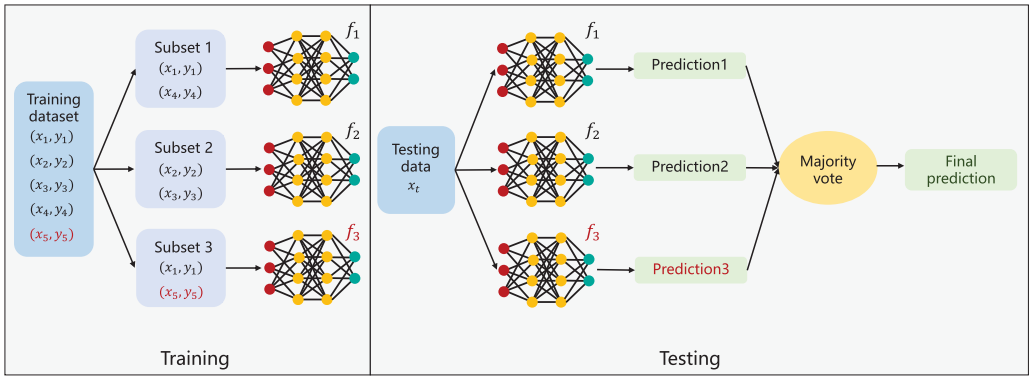


Fig. 9. The framework of bootstrap aggregating (bagging) defense.

causatively determines the performance of the learning systems. Therefore, as long as adversaries have access to the training process of any kind of machine learning systems, many applications of ML are vulnerable to poisoning attacks in nature. In this section, we discuss the most compelling applications in machine learning where poisoning attacks have demonstrated considerable success. Targeted on different applications, we also conclude their practical challenges and current solutions to provide directions for future researches.

Poisoning attacks can be used to manipulate learning systems in a range of application areas that we survey below.

5.1 Poisoning on Other Learning Paradigms

From the perspective of applications, poisoning attacks have also been applied in more expansive domains of machine learning, including the latest learning paradigms like federated learning, reinforcement learning, semi-supervised learning, and so on. We investigate poisoning researches toward these paradigms and summarize different challenges and corresponding countermeasures in their respective fields.

5.1.1 Federated Learning. FL is a novel distributed learning framework, where the deep learning model is trained in a collaborative manner among quantities of client devices. Roughly speaking, federated learning is formulated as a multi-round strategy including following three steps: (1) the server sends the current global model to client devices; (2) client devices update their local models using their local training datasets based on the global model, and send the latest local models to the server; (3) and the server computes a new global model via aggregating the local models according to a certain aggregation rule. Federated learning allows data to remain in the client devices and the shares between the server, and participants are only model parameters or updates, thus segregating the server from direct access to the private training data.

However, we notice that the federated learning architecture is vulnerable to poisoning attacks from malicious insider participants. The adversary can pretend to be a benign participant in federated learning but upload the poisoned parameters to the server so that he can easily affect the performance of the global model. The federated learning framework provides excellent chances for both data poisoning and model poisoning, since in this setting it is not permitted for the server to check any local training data of the clients, thus making poisoning attacks difficult to detect.

Researches [8, 80, 90] have evaluated the effect of data poisoning attack on federated learning. To poison the global model, the adversaries impersonate several benign clients but secretly craft malicious updates by injecting poisoned data into the local dataset. After the unguarded server

aggregates all the local updates (e.g., takes the average of all updates), the global model will inevitably be damaged. Notably, maybe more than one attacker colludes with each other and injects malicious training samples into local models of their own, which may result in a more harrowing catastrophe in federated learning [55].

In nature, methods above eventually influence the local updates through poisoning training data. However, it is a more intuitive idea for adversaries to directly poison the local updates without waste of time for modifying training data. Through constructing optimization problems, adversaries in References [3, 15] craft local updates that follow the opposite direction of the before-attack global model, thus making the global model gradually vestigial.

Existing defenses against poisoning attacks in centralized learning framework essentially aim to sanitize the training dataset. However, these defenses are not directly applicable in federated settings, because local training data is not reviewable to the server or any trustworthy authority. To generate a accurate global model, one idea is to detect malicious clients by distinguishing poisoned updates from benign updates, and another idea is to conduct robust aggregation rules to resist interference from unknown adversaries.

To detect malicious clients, Zhao et al. [93] realized the client-side cross-validation where each update is evaluated over other clients' local data. The server will adjust the weights of the updates based on the evaluation results when performing aggregation. Li et al. came up with a [41] spectral anomaly detection-based framework that distinguishes the abnormal model updates from the benign updates based on their low-dimensional embeddings. After spectral anomaly detection, the noisy and irrelevant updates will be removed while the benign updates will be retained.

Robust aggregation rules seem to be more widely studied. In 2017, Blanchard et al. [6] introduced Krum, an aggregation rule that satisfies resilience property and can rule out the local updates inconsistent with direction of the other updates. Krum computes a closest subset to all updated vectors, aiming at selecting the vectors that minimize the sum of the squared distances to every other vector. Inspired by geometric median, researches [11, 88] presented two Byzantine-robust **gradient descent (GD)** algorithms. In this strategy, the server iteratively updates the global model with gradient subtly restricted by coordinate-wise median and coordinate-wise trimmed mean. El Mhamdi et al. [25] proposed Bulyan, which computes trimmed mean on the basis of multiple Krum calculations. In the latest works, Konstantinov et al. [36] automatically assigned weights to updates of different sources by minimizing any weighted version of the empirical loss. In this way, the malicious clients will be assigned with increasingly insignificant weights, so that no further damage will be caused by the poisoned updates. On this basis, Fu et al. [22] combined repeated median regression with the residual-based reweighting scheme in iteratively reweighted least squares, thus providing both theoretical and experimental guarantees for the security of distributed learning systems. It is worth mentioning that robust aggregations can merely mitigate the process of poisoning attack but are unable to ultimately root out this threat.

Federated learning attaches importance to user privacy but compromises with security issues, thus leading to a dilemma, which will remain a heated concern in the future study. As a mainstream threat, poisoning attack against federated learning have arisen increasing attention.

5.1.2 Reinforcement Learning. RL is a method that learns what to do by interacting with our environment and mapping situations to actions, to maximize a numerical reward signal. Different from conventional machine learning, the reinforcement learner is not told which actions to take, but instead must discover which actions yield the most reward through trial-and-error search. The training data in RL is the trajectories that the learner rolls out from the environment, usually described with tuples like $(state, action, reward)$.

Focused on goal-directed learning from interaction, reinforcement learning is likely to suffer great losses from slightly poisoned rewards. In each iteration, the adversary eavesdrops on the interaction between the learner and the environment. According to certain poisoning strategies, the adversary manufactures poisons and send it to the learner. Recently, there has been poisoning attacks for bandit algorithms in both offline [49] and online [44] settings. The former setting assumes that the bandit algorithm updates periodically and that the attacker can manipulate the rewards before the updates in order to hijack the behavior of the bandit algorithm. In the latter setting, the bandit algorithm updates instantly for each time step while the attacker stealthily monitors the decision of the bandit algorithm at at each time and poisons the reward signal returned from the bandit environment.

Later researches [50, 67] extended poisoning attacks to more complicated RL settings where the final objective of RL agents is to find a policy that maximizes average reward, based on Markov decision processes. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and all subsequent rewards. In Reference [67], the attacker conducted poisoning attacks in two settings, including an offline setting where the agent is doing planning in the poisoned environment and an online setting where the agent is learning a policy using a regret-minimization framework with poisoned feedback. Sun et al. [78] built an online poisoning framework against deep RL via the **Vulnerability-aware Adversarial Critic Poison (VA2C-P)** algorithm working without any prior knowledge of the environment. By simulating the poisoning task as a reward-minimizing poisoning problem, this poisoning algorithm successfully prevents agents from learning a good policy or induces the agents to converge to a target policy. Similarly, Zhang et al. [92] formulated the optimal online poisoning problem as a stochastic optimal control problem, and provide a systematic solution using tools from model predictive control and deep RL.

Given the prominent applications of reinforcement learning in robotics, games, recommendation systems and so on, an intentionally and adversarially planted bad policy, like poisoning attacks, could be devastating. The study of poisoning attacks against reinforcement learning is still in its infancy. There are several promising directions for future work where attackers are able to expand the attack models, like simultaneously attacking rewards and transitions, or broaden the goals to specific target. In addition, there are few researches concerned with defenses against the training phase threat, which should be attached more significance.

5.1.3 Semi-supervised Learning. Labeling is time-consuming and labor-intensive. Therefore, unlabeled data is easy to obtain, while labeled data is usually difficult to collect. However, the current learning algorithms rely heavily on a strong assumption that all clients have a wealth of ground truth labeled data, which may not be always feasible in the real life. In this case, **Semi-Supervised Learning (SSL)** is more suitable for real-world applications, thus becoming a newly heated direction in the field of deep learning. SSL aims to improve the result of learning with a limited amount of labeled data together with a vast amount of unlabelled data.

Several works pay attention to poisoning tasks on the SSL paradigm. Focusing on **graph-based semi-supervised learning (G-SSL)**, Liu et al. [47] simulated the problem as a constrained quadratic minimization problem. Depending on whether it is a regression or classification task, attackers either take a continuous or discrete optimization method. To yield the highest reduction in prediction accuracy, [20] defined an influence function to capture the influence of labeled inputs onto unlabelled ones, based on which attackers can choose the most impressive samples as poisons. Taking **Semi-supervised Federated Learning (SSFL)** into consideration, Reference [48] found that the SSFL system is more susceptible to poisoning attacks than the common FL systems. In the scenarios of SSFL, the client's dataset is unlabeled, and the clients are required to use

pseudo-labeling to label unlabeled data. This creates an attack surface for malicious clients to arbitrarily generate wrong labels or modify the labels of the dataset. If the pseudo-labels given to the training samples are wrong or inappropriate, then these samples will become poisoned samples to compromise the modeling training. Mehra et al. [58] realized a novel data poisoning attack that, respectively, uses clean-label and dirty-label points to demonstrate the dramatic failure of **unsupervised domain adaptation (UDA)** methods at target generalization on benchmark datasets.

Depending on the known labels to infer the unknown labels, SSL algorithms are more sensitive to data quality than supervised machine learning, thus making it important to study the potential threats related to the training phrase, more specifically, poisoning attacks. Nevertheless, the literature about robust semi-supervised learning is still scarce at present. In consequence, it is still a promising and significant direction for future work.

5.1.4 Multi-task Learning. **Multi-task learning (MTL)** is a machine learning paradigm that improves the performance of each task by exploiting useful information contained in multiple related tasks. However, the relatedness of tasks can also be exploited by attackers to launch data poisoning attacks.

A recent work [95] focused on **multi-task relationship learning (MTRL)** models, a popular subclass of MTL models where the relationships of different tasks are learned and quantified directly from training data. Concluded as a bi-level optimization, the proposed PATOM algorithm maximizes the poisoned data in the direction of increasing the empirical loss of target tasks through a stochastic gradient ascent. The research demonstrates with experiments that the attacker can indirectly influence the target tasks by contaminating training data of other tasks even though he cannot inject any data into the target task.

According to our investigation, optimization of poisoning attacks on MTL models can be much more challenging than that on STL models, because tasks in MTL are related to each other and an attack on one task might potentially influence other tasks. Simultaneously, this also provides convenience for the attacker to indirectly influence the inaccessible target tasks by attacking some accessible tasks, which cannot be addressed by existing methods on poisoning STL models.

5.2 Poisoning Toward Different Tasks

Early works of poisoning attacks focus on **computer vision (CV)** tasks as mentioned above, but nowadays applications extend much further than that.

5.2.1 Computer Vision Tasks. CV is a main direction in applications of machine learning, which makes it a heated issue for adversaries. Though recent machine learning algorithms have achieved remarkable accuracy in common CV tasks, such as image classification, image location, object detection, semantic segmentation, and so on, they are still vulnerable faced with threats like poisoning attacks. In the early researches of poisoning attacks, most researchers conduct experiments based on CV datasets and classification tasks. However, few works extend the application to other CV tasks where further study is needed.

5.2.2 Text Tasks. Apart from computer vision, poisoning attacks also exist in applications of text domain, i.e., NLP. Previous works of adversarial examples have demonstrated the inherent vulnerability of learning tasks in text domain, even in state-of-the-art strategies. First, there are not enough high-quality manually labeled datasets in text domain, like ImageNet in CV tasks. Large-scale natural-language corpora are drawn from public sources that are weakly examined can be edited by any adversary. Therefore, the adversary's modifications can survive until they are used for training. Second, in NLP tasks, pre-trained embedding is more commonly used than that in CV tasks, which provides chances for model poisoning. In case of any crisis of model poisoning and

data poisoning, not only should we carefully check over pre-trained model, but we should also pay attention to fine-tuning process.

Difference between the properties of text tasks and other tasks determines necessary changes on poisoning strategy. The input in CV tasks is continuous RGB value, while in NLP problems, the input is a discrete word sequence in the form of one-hot vector. If the perturbation is directly carried out on raw text, then the poison may be meaningless. Because the set of high-dimensional one-hot vectors does not admit infinitesimal perturbation, researchers [9, 38, 72] defined the perturbation on continuous word embedding instead of discrete word inputs. Kurita et al. [38] validated the potential threat of weight poisoning from untrusted pre-trained models on three text classification tasks: sentiment classification, toxicity detection and spam detection. [9, 72] conducted data poisoning via modifying locations in the embedding space.

5.2.3 Recommendation System. Recommendation system is an imperative component of many web services to help users locate items they are interested in. However, several studies [16, 17, 29, 40, 44, 49, 50, 67] have demonstrated that recommendation systems are also vulnerable to poisoning attacks, which can induce the system to make recommendations as the adversary desires. As recommendation systems are driven by user-item interaction data, adversaries can manipulate a recommendation system via injecting fake users with fake user-item interaction data to the system.

Taking special utilities into consideration, most attacks against recommendation systems are targeted at specific users or items. Researchers have explored data poisoning attacks against stochastic multi-armed bandits [44] and contextual bandits [49], which belong to the family of online learning algorithms with limited feedback. As a simplified paradigms of reinforcement learning, such algorithms are widely used in real-world applications such as news article recommendation. Through modifying rewards of targeted trials before the updates, an adversary can easily hijack the behavior of the bandit algorithm. Another common solution for recommendation system is called the factorization-based collaborative filtering algorithm. According to a large amount of user-item interaction data, this algorithm learns to model users' latent preferences for different items. Researchers formulate optimization problems for selecting filler items and assigning rating scores for the fake users, with an objective to maximize the number of normal users to whom the target item is recommended [16, 29, 40]. Similarly, graph-based recommendation systems is also proved vulnerable to these optimization-based poisoning attacks [17].

5.2.4 Well-intentioned Poisoning. Depending on the way it is used, poisoning attack can also play a role in the right way, such as copyright protection and privacy protection.

Orekondy et al. [63] applied poisoning attacks to an active defense against highly accurate model stealing attacks. The goal of model stealing attacker is to train a replica model with the predictions from the target model. By introducing bounded perturbations to poison the predictions, the defense amplify the attacker's error rate up to 85 times with minimal impact on the utility for benign users.

Poisoning attack can also serve as a scheme of privacy protection in data release and sharing situations. Due to, respectively, limited training data, different entities addressing the same vision task can share task-specific image data to enlarge each other's training data volume. However, these images may involve sensitive contents, which result in privacy hazards. Guo et al. [26] proposed a private **Deep Poisoning Module (DPM)**, which deliberately poisons convolutional image features to prevent image reconstructions, while ensuring that the altered image data is functionally equivalent to the non-poisoned data for the specific vision task.

6 FUTURE WORK

We conclude that there are clearly opportunities for future research in the following fields.

6.1 Explainable Poisoning

The following questions have puzzled researchers from the very beginning to the very present during the development of poisoning attacks. When focusing on a certain poison, what kind of poisoned feature will interfere the model most? When it comes to a large group of poisons, what kind of poison will twist the decision space of the model to the most extent? As we indicate in Section 3, existing mainstream strategies craft poisons by combining label poisoning with adversarial perturbation via solving a certain optimization problem. Due to the fact that the heavy computational expense of solutions to the bilevel optimization greatly block up the application of poisoning attack on complicated learning paradigms like deep learning, an essential problem to be solved is to explore the mechanism behind. Since no universally accepted conjecture on the existence of how poisoned samples effect machine learning models, some more fundamental approaches are required, such as topology, statistics, and other learning theories, which may accelerate and facilitate the existence of some more threatening attacks and more powerful defenses.

6.2 Transferable Poisoning

Current poisoning methods mainly focus on white-box attacks, which require adequate knowledge toward the target victim. On the one hand, it would be empirically impossible for adversaries to get access to such information in reality without the assistance of any internal spy, which is difficult to implement in time before every trial. On the other hand, poisons generated by complicated computation are specifically effective for the certain model but behave poorly on others. As long as the target model is updated or replaced, the fruits of previous work will turn into waste and the same hard work has to start all over again. Therefore, inspired by the concept of transferability, it seems that there is a lot of room for poisoning attack to improve from this perspective. Our ultimate goal is to conduct effective attacks in black-box and variable settings.

6.3 Accelerated Poisoning

The biggest obstacle in the application of untargeted poisoning attacks, which need more poisons to wreak yet more damage. Though gradient-based solutions to bi-level optimization acquired certain effect on traditional machine learning, similar but advanced approaches encounter setbacks in deep learning algorithms. Faced with rapidly increasing computational complexity, current methods are sufficient to deal with industrial-scale problems.

6.4 Attacks Toward Other Tasks

Though adopted in such broad applications, the potential of poisoning attack has not been fully realized. As the rapid replacement of machine learning algorithm, poisoning methods always need to be upgraded in the face of brand-new and unique challenges.

6.5 Comprehensive Defenses

Normal users usually have no idea about which surface of the ML system will be attacked and which attack method the adversaries may use. Therefore, it is difficult to exactly select a suitable defense method with the best effect at once. In addition, new type of poisoning, e.g., clean-label poisoning, and new scenario of poisoning, e.g., federated learning, invalidate previous approaches and claim customized defenses. Until now, no defense (including the ones using formal verification) dares to declare that it is effective against all attacks. Therefore, this issue will remain active and span several future research directions.

6.6 Negative Effects of Defenses

To prevent the ML system from being poisoned, some defense methods may filter suspicious data in the training dataset. Since the defenders have no knowledge about what does the poisoned data look like, some benign samples may be filtered by mistake, which will change the distribution of the training data, especially in a long-tailed distribution. These defenses may significantly reduce the accuracy of the ML system and may bring about fairness problems of artificial intelligence.

6.7 Low-cost Defenses

Existing methods to prevent poisoning attacks are usually time-consuming and resource-consuming. Poison-filter-based defenses need extra data to train poison filters and inspect every data in the training set, which requires high expenditure of time. Robust training-based defenses also need additional calculations to improve the robustness of the model. Therefore, how to reduce the cost of defenses is an issue worth exploring.

6.8 Judge the Attacks and the Defenses in the Physical World

Despite the fact that researches among poisoning attacks and defenses has last for more than a decade, their true impact on safety and security in the physical world have not been evaluated at this moment. This will be a direction of practical significance.

6.9 Qualitative Analysis and Benchmark

Existing works of poisoning attacks and defenses apply different configurations and experimental criterion, which makes it difficult to compare the merits and disadvantages of different works. A unified framework of qualitative analysis is urgently needed to achieve a better assessment of different approaches and encourage the appearance of true progress in this field.

7 CONCLUSION

Poisoning attacks, as a typical threat in the training phrase of ML, is a critical and booming research area. In this article, we provide a comprehensive overview of state-of-the-art studies based on several identified dimensions. In particular, we focus on explaining the hypotheses on the existence of poisoning attacks and position the phenomenon in the security and robustness context. We categorize and characterize existing attacks and defenses proposed in the literature and present the wide applications of poisoning attacks in different ML systems. The potential research directions are illustrated at the end. We hope that this article can remind researchers of the inherent vulnerabilities in the training phase of ML and arouse attention to poisoning attacks. The fierce cat-and-mouse game of wit and evasion between attacks and defenses seems never come to rest. As far as we can see, it would be an important step toward trustworthy ML.

REFERENCES

[1] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. Doug Tygar. 2010. The security of machine learning. *Mach. Learn.* 81, 2 (2010), 121–148.

[2] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. Doug Tygar. 2006. Can machine learning be secure? In *Proceedings of the ACM Symposium on Information, Computer and Communications Security*. 16–25.

[3] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *Proceedings of the International Conference on Machine Learning*. PMLR, 634–643.

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*. 1467–1474.

[5] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recogn.* 84 (2018), 317–331.

[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 118–128.

[7] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. 2021. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'21)*. IEEE, 3855–3859.

[8] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding distributed poisoning attack in federated learning. In *Proceedings of the IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS'19)*. IEEE, 233–239.

[9] Alvin Chan, Yi Tay, Yew-Soon Ong, and Aston Zhang. 2020. Poison attacks against text datasets with conditional adversarially regularized autoencoder. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: Findings*. 4175–4189.

[10] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. 2018. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Comput. Secur.* 73 (2018), 326–344.

[11] Yudong Chen, Lili Su, and Jiaming Xu. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–25.

[12] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. 2008. Casting out demons: Sanitizing training data for anomaly sensors. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'08)*. IEEE, 81–95.

[13] Jia Ding and Zhiwu Xu. 2020. Adversarial attacks on deep learning models of computer vision: A survey. In *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 396–408.

[14] Georgios Drainakis, Konstantinos V. Katsaros, Panagiotis Pantazopoulos, Vasilis Sourlas, and Angelos Amditis. 2020. Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis. In *Proceedings of the IEEE 19th International Symposium on Network Computing and Applications (NCA'20)*. IEEE, 1–8.

[15] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security'20)*. 1605–1622.

[16] Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. 2020. Influence function-based data poisoning attacks to top-n recommender systems. In *Proceedings of the Web Conference 2020*. 3019–3025.

[17] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 381–392.

[18] Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. 2019. Learning to confuse: Generating training time adversarial data with auto-encoder. *Adv. Neural Info. Process. Syst.* 32 (2019), 11994–12004.

[19] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. 2014. Robust logistic regression and classification. *Adv. Neural Info. Process. Syst.* 27 (2014), 253–261.

[20] Adriano Franci, Maxime Cordy, Martin Gubri, Mike Papadakis, and Yves Le Traon. 2020. Effective and efficient data poisoning in semi-supervised learning. Retrieved from https://arXiv:2012.07381.

[21] Benoît Frénay and Michel Verleysen. 2013. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 5 (2013), 845–869.

[22] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. 2019. Attack-resistant federated learning with residual-based reweighting. Retrieved from https://arXiv:1912.11464.

[23] Jonas Geiping, Liam H. Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. 2020. Witches' Brew: Industrial scale data poisoning via gradient matching. In *Proceedings of the International Conference on Learning Representations*.

[24] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2022. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Trans. Pattern Anal. Mach. Intell.* 99, 1 (2022), 1–18.

[25] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3521–3530.

[26] Hao Guo, Brian Dolhansky, Eric Hsin, Phong Dinh, Cristian Canton Ferrer, and Song Wang. 2021. Deep poisoning: Towards robust image data sharing against visual disclosure. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 686–696.

[27] Junfeng Guo and Cong Liu. 2020. Practical poisoning attacks on neural networks. In *Proceedings of the European Conference on Computer Vision*. Springer, 142–158.

[28] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. 2018. Using trusted data to train deep networks on labels corrupted by severe noise. *Adv. Neural Info. Process. Syst.* 31 (2018), 10456–10465.

[29] Rui Hu, Yuanxiong Guo, Miao Pan, and Yanmin Gong. 2019. Targeted poisoning attacks on social recommender systems. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'19)*. IEEE, 1–6.
[30] W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. 2020. MetaPoison: Practical general-purpose clean-label data poisoning. *Adv. Neural Info. Process. Syst.* 33 (2020), 12080–12091.
[31] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'18)*. IEEE, 19–35.
[32] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. 2021. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7961–7969.
[33] Michael I. Jordan and Tom M. Mitchell. 2015. Machine learning: Trends, perspectives, and prospects. *Science* 349, 6245 (2015), 255–260.
[34] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1885–1894.
[35] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. 2018. Stronger data poisoning attacks break data sanitization defenses. Retrieved from https://arXiv:1811.00741.
[36] Nikola Konstantinov and Christoph Lampert. 2019. Robust learning from untrusted sources. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3488–3498.
[37] Moshe Kravchik, Battista Biggio, and Asaf Shabtai. 2021. Poisoning attacks on cyber attack detectors for industrial control systems. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 116–125.
[38] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2793–2806.
[39] Alexander Levine and Soheil Feizi. 2020. Deep partition aggregation: Provable defenses against general poisoning attacks. In *Proceedings of the International Conference on Learning Representations*.
[40] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Adv. Neural Info. Process. Syst.* 29 (2016), 1885–1893.
[41] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. 2020. Learning to detect malicious clients for robust federated learning. Retrieved from https://arXiv:2002.00211.
[42] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. 2017. Learning from noisy labels with distillation. In *Proceedings of the IEEE International Conference on Computer Vision*. 1910–1918.
[43] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. 2017. Robust linear regression against training data poisoning. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 91–102.
[44] Fang Liu and Ness Shroff. 2019. Data poisoning attacks on stochastic bandits. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4042–4050.
[45] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 273–294.
[46] Sijia Liu, Songtao Lu, Xiangyi Chen, Yao Feng, Kaidi Xu, Abdullah Al-Dujaili, Mingyi Hong, and Una-May O¡¯Reilly. 2020. Min-max optimization without gradients: Convergence and applications to black-box evasion and poisoning attacks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 6282–6293.
[47] Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. 2019. A unified framework for data poisoning attack to graph-based semi-supervised learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 9780–9790.
[48] Yi Liu, Xingliang Yuan, Ruihui Zhao, Yifeng Zheng, and Yefeng Zheng. 2020. RC-SSFL: Towards robust and communication-efficient semi-supervised federated learning system. Retrieved from https://arXiv:2012.04432.
[49] Yuzhe Ma, Kwang-Sung Jun, Lihong Li, and Xiaojin Zhu. 2018. Data poisoning attacks in contextual bandits. In *Proceedings of the International Conference on Decision and Game Theory for Security*. Springer, 186–204.
[50] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Xiaojin Zhu. 2019. Policy poisoning in batch reinforcement learning and control. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 14570–14580.
[51] Yuzhe Ma, Xiaojin Zhu Zhu, and Justin Hsu. 2019. Data poisoning against differentially private learners: Attacks and defenses. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 4732–4738.
[52] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. 2018. Learning under p-tampering attacks. In *Algorithmic Learning Theory*. PMLR, 572–596.
[53] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. 2019. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4536–4543.
[54] Saeed Mahloujifar and Mohammad Mahmoody. 2017. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. In *Proceedings of the Theory of Cryptography Conference*. Springer, 245–279.

[55] Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. 2019. Data poisoning attacks in multi-party learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4274–4283.
[56] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.
[57] Akshay Mehra, Bhavya Kailkhura, Pin-Yu Chen, and Jihun Hamm. 2021. How robust are randomized smoothing-based defenses to data poisoning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13244–13253.
[58] Akshay Mehra, Bhavya Kailkhura, Pin-Yu Chen, and Jihun Hamm. 2021. Understanding the limits of unsupervised domain adaptation via data poisoning. Retrieved from https://arXiv:2107.03919.
[59] Shike Mei and Xiaojin Zhu. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. 2871–2877.
[60] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 27–38.
[61] Luis Muñoz-González, Bjarne Pfitzner, Matteo Russo, Javier Carnerero-Cano, and Emil C. Lupu. 2019. Poisoning attacks with generative adversarial nets. Retrieved from https://arXiv:1906.07773.
[62] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. Doug Tygar, and Kai Xia. 2008. Exploiting machine learning to subvert your spam filter. In *Proceedings of First USENIX Workshop on Large Scale Exploits and Emergent Threats (LEET'08)*. 1–9.
[63] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2020. Prediction poisoning: Towards defenses against DNN model stealing attacks. In *Proceedings of the 8th International Conference on Learning Representations*.
[64] Naman Patel, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrami. 2020. Bait and switch: Online training data poisoning of autonomous driving systems. Retrieved from https://arXiv:2011.04065.
[65] Andrea Paudice, Luis Muñoz-González, and Emil C. Lupu. 2018. Label sanitization against label flipping poisoning attacks. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 5–15.
[66] Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. 2020. Deep k-NN defense against clean-label data poisoning attacks. In *Proceedings of the European Conference on Computer Vision*. Springer, 55–70.
[67] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2020. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 7974–7984.
[68] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4334–4343.
[69] Mauro Ribeiro, Katarina Grolinger, and Miriam A. M. Capretz. 2015. Mlaas: Machine learning as a service. In *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA'15)*. IEEE, 896–902.
[70] Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. 2020. Fr-train: A mutual information-based approach to fair and robust training. In *Proceedings of the International Conference on Machine Learning*. PMLR, 8147–8157.
[71] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. 2020. Certified robustness to label-flipping attacks via randomized smoothing. In *Proceedings of the International Conference on Machine Learning*. PMLR, 8230–8241.
[72] Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. 2020. Humpty dumpty: Controlling word meanings via corpus poisoning. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'20)*. IEEE, 1295–1313.
[73] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. 2021. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security'21)*. 1559–1575.
[74] Alex Serban, Erik Poll, and Joost Visser. 2020. Adversarial examples on object recognition: A comprehensive survey. *ACM Comput. Surveys* 53, 3 (2020), 1–38.
[75] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 6106–6116.
[76] Shiqi Shen, Shruti Tople, and Prateek Saxena. 2016. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 508–519.
[77] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 3520–3532.
[78] Yanchao Sun, Da Huo, and Furong Huang. 2020. Vulnerability-aware poisoning mechanism for online RL with unknown dynamics. In *Proceedings of the International Conference on Learning Representations*.

[79] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*.

[80] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 480–501.

[81] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 8011–8021.

[82] Leslie G. Valiant. 1984. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.

[83] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. 2017. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 839–847.

[84] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning? In *Proceedings of the International Conference on Machine Learning*. PMLR, 1689–1698.

[85] Han Xiao, Huang Xiao, and Claudia Eckert. 2012. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*. IOS Press, 870–875.

[86] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K. Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *Int. J. Autom. Comput.* 17, 2 (2020), 151–178.

[87] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. 2017. Generative poisoning attack method against neural networks. Retrieved from https://arXiv:1703.01340.

[88] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5650–5659.

[89] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 9 (2019), 2805–2824.

[90] Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. 2019. Poisoning attack in federated learning using generative adversarial nets. In *Proceedings of the 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)'19*. IEEE, 374–380.

[91] Rui Zhang and Quanyan Zhu. 2017. A game-theoretic analysis of label flipping attacks on distributed support vector machines. In *Proceedings of the 51st Annual Conference on Information Sciences and Systems (CISS'17)*. IEEE, 1–6.

[92] Xuezhou Zhang, Xiaojin Zhu, and Laurent Lessard. 2020. Online data poisoning attacks. In *Learning for Dynamics and Control*. PMLR, 201–210.

[93] Lingchen Zhao, Shengshan Hu, Qian Wang, Jianlin Jiang, Shen Chao, Xiangyang Luo, and Pengfei Hu. 2020. Shielding collaborative learning: Mitigating poisoning attacks through client-side detection. *IEEE Trans. Depend. Secure Comput.* 18, 5 (2020), 2029–2041.

[94] Mengchen Zhao, Bo An, Wei Gao, and Teng Zhang. 2017. Efficient label contamination attacks against black-box learning models. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 3945–3951.

[95] Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. 2018. Data poisoning attacks on multi-task relationship learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2628–2635.

[96] Chen Zhu, W. Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2019. Transferable clean-label poisoning attacks on deep neural nets. In *Proceedings of the International Conference on Machine Learning*. PMLR, 7614–7623.

Received 15 December 2021; revised 5 May 2022; accepted 13 May 2022