Cinà, A.E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B.A., Oprea, A., Biggio, B., Pelillo, M., Roli, F., 2023. Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning, ACM Computing. Survey. 55, 13s, 294:1-294:39.

Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning

ANTONIO EMANUELE CINÀ*, DAIS, Ca' Foscari University of Venice, Italy
KATHRIN GROSSE*, VITA Lab, École Polytechnique Fédérale de Lausanne, Switzerland
AMBRA DEMONTIS†, DIEE, University of Cagliari, Italy
SEBASTIANO VASCON, DAIS, Ca' Foscari University of Venice and European Center for Living Technology, Italy
WERNER ZELLINGER, Software Competence Center Hagenberg GmbH (SCCH), Austria
BERNHARD A. MOSER, Software Competence Center Hagenberg GmbH (SCCH), Austria
ALINA OPREA, Khoury College of Computer Sciences, Northeastern University, MA, USA
BATTISTA BIGGIO, DIEE, University of Cagliari, CINI, and Pluribus One, Italy
MARCELLO PELILLO, DAIS, Ca' Foscari University of Venice, Italy
FABIO ROLI, DIBRIS, University of Genoa, CINI, and Pluribus One, Italy

The success of machine learning is fueled by the increasing availability of computing power and large training datasets. The training data is used to learn new models or update existing ones, assuming that it is sufficiently representative of the data that will be encountered at test time. This assumption is challenged by the threat of poisoning, an attack that manipulates the training data to compromise the model's performance at test time. Although poisoning has been acknowledged as a relevant threat in industry applications, and a variety of different attacks and defenses have been proposed so far, a complete systematization and critical review of the field is still missing. In this survey, we provide a comprehensive systematization of poisoning attacks and defenses in machine learning, reviewing more than 100 papers published in the field in the last 15 years. We start by categorizing the current threat models and attacks, and then organize existing defenses accordingly. While we focus mostly on computer-vision applications, we argue that our systematization also encompasses state-of-the-art attacks and defenses for other data modalities. Finally, we discuss existing resources for research in poisoning, and shed light on the current limitations and open research questions in this research field.

*Equal contribution
†Corresponding Author

Authors' addresses: Antonio Emanuele Cinà, antonioemanuele.cina@unive.it, DAIS, Ca' Foscari University of Venice, Via Torino, 155, Venice, Italy, 30170; Kathrin Grosse, kathrin.grosse@unica.it, VITA Lab, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1015; Ambra Demontis, ambra.demontis@unica.it, DIEE, University of Cagliari, Italy; Sebastiano Vascon, sebastiano.vascon@unive.it, DAIS, Ca' Foscari University of Venice and European Center for Living Technology, Italy; Werner Zellinger, werner.zellinger@scch.at, Software Competence Center Hagenberg GmbH (SCCH), Softwarepark, 21, Hagenberg, Austria, 4232; Bernhard A. Moser, bernhard.moser@scch.at, Software Competence Center Hagenberg GmbH (SCCH), Austria; Alina Oprea, a.oprea@northeastern.edu, Khoury College of Computer Sciences, Northeastern University, 360 Huntington Ave, Boston, MA, USA, 09123; Battista Biggio, battista.biggio@unica.it, DIEE, University of Cagliari, CINI, and Pluribus One, Italy; Marcello Pelillo, pelillo@unive.it, DAIS, Ca' Foscari University of Venice, Italy; Fabio Roli, fabio.roli@unige.it, DIBRIS, University of Genoa, CINI, and Pluribus One, Via All'Opera Pia, 13, Genoa, Italy, 09124.

Manuscript submitted to ACM

CCS Concepts: • **General and reference → Surveys and overviews**; • **Computing methodologies → Neural networks**; **Machine learning**; **Adversarial learning**.

Additional Key Words and Phrases: Poisoning Attacks, Backdoor Attacks, Machine Learning, Computer Vision, Computer Security

ACM Reference Format:

Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A. Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. 2021. Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning. 1, 1 (January 2021), 37 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The unprecedented success of machine learning (ML) in many diverse applications has been inherently dependent on the increasing availability of computing power and large training datasets, under the implicit assumption that such datasets are well representative of the data that will be encountered at test time. However, this assumption may be violated in the presence of *data poisoning* attacks, i.e., if attackers can either compromise the training data, or gain some control over the learning process (e.g., when model training is outsourced to an untrusted third-party service) [43, 70, 126, 134]. Poisoning attacks are staged at training time, and consist of manipulating the training data to degrade the model's performance at test time. Three main categories of data poisoning attacks have been investigated so far [39]. These include indiscriminate, targeted, and backdoor poisoning attacks. In *indiscriminate* poisoning attacks, the attacker manipulates a fraction of the training data to maximize the classification error of the model on the (clean) test samples. In *targeted* poisoning attacks, the attacker manipulates again a subset of the training data, but this time to cause misclassification of a specific set of (clean) test samples. In *backdoor* poisoning attacks, the training data is manipulated by adding poisoning samples containing a specific pattern, referred to as the backdoor trigger, and labeled with an attacker-chosen class label. This typically induces the model to learn a strong correlation between the backdoor trigger and the attacker-chosen class label. Accordingly, at test time, the input samples that embed the trigger are misclassified as samples of the attacker-chosen class.

Although many different attacks can be staged against ML models, a recent survey shows that poisoning is the largest concern for ML deployment in industry [68, 95]. Furthermore, several sources confirm that poisoning is already carried out in practice [68, 119]. For example, Microsoft's chatbot Tay[1] was designed to learn language by interacting with users, but instead learned offensive statements. Chatbots in other languages have shared its fate, including a Chinese[2] and a Korean[3] version. Another attack showed how to poison the auto-complete feature in search engines.[4] Finally, a group of extremists submitted wrongly-labeled images of portable ovens with wheels tagging them as *Jewish baby strollers* to poison Google's image search.[5] Due to their practical relevance, various scientific articles have been published on training-time attacks against ML models. While the vast majority of the poisoning literature focuses on supervised classification models in the computer vision domain, we would like to remark here that data poisoning has been investigated earlier in cybersecurity [126, 134], and more recently also in other application domains, like audio [1, 91] and natural language processing [34, 206], and against different learning methods, such as federated learning [4, 191], unsupervised learning [17, 41], and reinforcement learning [10, 205].

[1] https://www.theguardian.com/technology/2016/mar/26/microsoft-deeply-sorry-for-offensive-tweets-by-ai-chatbot
[2] https://www.khaleejtimes.com/technology/ai-getting-out-of-hand-chinese-chatbots-re-educated-after-rogue-rants
[3] https://www.vice.com/en/article/akd4g5/ai-chatbot-shut-down-after-learning-to-talk-like-a-racist-asshole
[4] http://www.nickdiakopoulos.com/2013/08/06/algorithmic-defamation-the-case-of-the-shameless-autocomplete/
[5] https://www.timebulletin.com/jewish-baby-stroller-image-algorithm/

Within this survey paper, we provide a comprehensive framework for threat modeling of poisoning attacks and categorization of defenses. We identify the main practical scenarios that enable staging such attacks on ML models, and use our framework to properly categorize attacks and defenses. We then review their historical development, also highlighting the main current limitations and the corresponding future challenges. We do believe that our work can serve as a guideline to better understand how and when these attacks can be staged, and how we can defend effectively against them, while also giving a perspective on the future development of trustworthy ML models limiting the impact of malicious users. With respect to existing surveys in the literature on ML security, which either consider a high-level overview of the whole spectrum of attacks on ML [19, 26], or are specific to an application domain [159, 187], our work focuses solely on poisoning attacks and defenses, providing a greater level of detail and a more specific taxonomy. Other survey papers on poisoning attacks do only consider backdoor attacks [59, 88, 103], except for the work by Goldblum et al. [64] and Tian et al. [164]. Our survey is complementary to recent work in [64, 164]; in particular, while in [64, 164] the authors give an overview of poisoning attacks and countermeasures in centralized and federated learning settings, our survey: (i) categorizes poisoning attacks and defenses in the centralized learning setting, based on a more systematic threat modeling; (ii) introduces a unified optimization framework for poisoning attacks, matches the defenses with the corresponding attacks they prevent, and (iii) discusses the historical timeline of poisoning attacks since the early developments in cybersecurity applications of ML, dating back to more than 15 years ago.

We start our review in Sect. 2, with a detailed discussion on threat modeling for poisoning attacks, and on the underlying assumptions needed to defend against them. This includes defining the learning settings where data poisoning attacks (and defenses) are possible. We further highlight the different attack strategies that give us a scaffold for a detailed overview of data poisoning attacks in Sect. 3. Subsequently, in Sect. 4, we give an overview of the main defense mechanisms proposed to date against poisoning attacks, including training-time and test-time defense strategies. While our survey is mostly focused on poisoning classification models for computer vision, which encompasses most of the work related to poisoning attacks and defenses, in Sect. 5 we discuss related work that has been developed in different contexts. In Sect. 6, we discuss poisoning research resources such as libraries and dataset containing poisoned models. Finally, in Sect. 7 we review the historical development of poisoning attacks and defenses. This overview serves as a basis for discussing ongoing challenges in the field, such as limitations of current threat models, the design of more scalable attacks, and the arms race towards designing more comprehensive and effective defenses. For each of these points, we discuss open questions and related future work.

To summarize, this work provides the following contributions: (i) we propose a unifying framework for threat modeling of poisoning attacks and systematization of defenses; (ii) we categorize around 45 attack approaches in computer vision according to their assumptions and strategies; (iii) we provide a unified formalization for optimizing poisoning attacks via bilevel programming; (iv) we categorize more than 70 defense approaches in computer vision, defining six distinct families of defenses; (v) we take advantage of our framework to match specific attacks with appropriate defenses according to their strategies; (vi) we discuss state-of-the-art libraries and datasets as resources for poisoning research; and (vii) we show the historical development of poisoning research and derive open questions, pressing issues, and challenges within the field of poisoning research. Finally, we also derive a unified formalization for optimizing poisoning attacks via bilevel programming, and investigate in the supplementary material in which other domains poisoning attacks and defenses have been developed.

Table 1. Notation and symbols used in this survey.

Data	Model	Noise
$\mathcal{D}$ Clean samples in training set	$\boldsymbol{\theta}$ Model's parameters	$\boldsymbol{t}$ Test data perturbation
$\mathcal{D}_p$ Poisoning samples in training set	ϕ Model's feature extractor	$\boldsymbol{\delta}$ Training data perturbation
$\mathcal{D}'$ Poisoned training set ($\mathcal{D}' = \mathcal{D} \cup \mathcal{D}_p$)	f Model's classifier	Δ Set of admissible manipulations for $\boldsymbol{\delta}$
	Training	**Attack Strategy**
$\mathcal{V}$ Clean samples in validation dataset	$\mathcal{M}$ Machine learning model	BL Bilevel
$\mathcal{V}_t$ Attacker target samples in validation dataset	$\mathcal{W}$ Learning algorithm	FC Feature Collision
	L Loss function	T^P Patch Trigger
$\mathcal{T}$ Test samples	$\mathcal{L}$ Training loss (regularized)	T^S Semantical Trigger
p Percentage of poisoned data		T^F Functional Trigger

2 MODELING POISONING ATTACKS AND DEFENSES

We discuss here how to categorize poisoning attacks against learning-based systems. We start by introducing the notation and symbols used throughout this paper in Table 1. In the remainder of this section, we define the learning settings under which poisoning attacks have been investigated. We then revisit the framework by Muñoz-González et al. [124] to systematize poisoning attacks according to the attacker's goal, knowledge of the target system, and capability of manipulating the input data. We conclude by characterizing the defender's goal, knowledge, and capability.

2.1 Learning Settings

We define here the three main scenarios under which ML models can be trained, and which can pose serious concerns in relationship to data poisoning attacks. We refer to them below respectively as (i) *training-from-scratch*, (ii) *fine-tuning*, and (iii) *model-training*. In Fig. 1, we conceptually represent these settings, along with the entry points of the attack surface which enable staging a poisoning attack.

Training from Scratch (TS) and Fine Tuning (FT). In the *training-from-scratch* and *fine-tuning* scenarios, the user controls the training process, but collects the training data from external repositories, potentially compromised by attackers. In practice, these are the cases where data gathering and labeling represent time-consuming and expensive tasks that not all organizations and individuals can afford, forcing them to collect data from untrusted external sources. The distinction between the two scenarios hinges on how the collected data are employed during training. In the *training-from-scratch* scenario, the collected data is used to train the model from a random initialization of its weights. In the *fine-tuning* setting, instead, a pretrained model is typically downloaded from an untrusted source, and used to map the input samples on a given representation space induced by a feature mapping function ϕ. Then, a classification function f is fine tuned on top of the given representation ϕ.

Model Training (MT). In the *model-training* (outsourcing) scenario, the user is supposed to have limited computational capacities and outsources the whole training procedure to an untrusted third party, while providing the training dataset. The resulting model can then be provided either as an online service which the user can access via queries, or given directly to the user. In this case, both the feature mapping ϕ and the classification function f are trained by the attacker (i.e., the untrusted party). The user, however, can validate the model's accuracy on a separate validation dataset to ensure that the model meets the desired performance requirements.

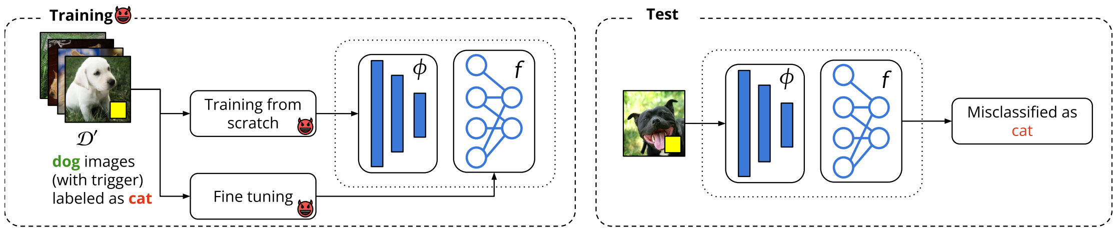


Fig. 1. Training (left) and test (right) pipeline. The victim collects a training dataset $\mathcal{D}'$ from an untrusted source. The training or fine-tuning algorithm uses these data to train a model $\mathcal{M}$, composed of a feature extractor ϕ, and a classification layer f. In the case of fine-tuning, only f is modified, while the feature representation ϕ is left untouched. At test time, some test samples may be manipulated by the attacker to exploit the poisoned model and induce misclassification errors.

2.2 Attack Framework

2.2.1 Attacker's Goal. The goal of a poisoning attack can be defined in terms of the intended security violation, and the attack and error specificity, as detailed below.

Security Violation. It defines the security violation caused by the attack, which can be: (i) an *integrity* violation, if malicious activities evade detection without compromising normal system operation; (ii) an *availability* violation, if normal system functionality is compromised, causing a denial of service for legitimate users; or (iii) a *privacy* violation, if the attacker aims to obtain private information about the system itself, its users, or its data.

Attack Specificity. It determines which samples are subject to the attack. It can be: (i) *sample-specific* (targeted), if a specific set of sample(s) is targeted by the attack, or (ii) *sample-generic* (indiscriminate), if any sample can be affected.

Error Specificity. It determines how the attack influences the model's predictions. It can be: (i) *error-specific*, if the attacker aims to have a sample misclassified as a specific class; or (ii) *error-generic*, if the attacker attempts to have a sample misclassified as any class different from the true class.

2.2.2 Attacker's Knowledge. The attacker may get to know some details about the target system, including information about: (i) the (clean) training data $\mathcal{D}$, (ii) the ML model $\mathcal{M}$ being used, and (iii) the test data $\mathcal{T}$. The first component considers how much knowledge the attacker has on the training data. The second component refers to the ability of the attacker to access the target model, including its internal (trained) parameters $\boldsymbol{\theta}$, but also additional information like hyperparameters, initialization, and the training algorithm. The third component specifies if the attacker knows in advance (or has access to) the samples that should be misclassified at test time. Although not explicitly mentioned in previous work, we have found that the knowledge of test samples is crucial for some attacks to work as expected. Clearly, attacks that are designed to work on specific test instances are not expected to generalize to different test samples (e.g., to other samples belonging to the same class). Depending on the combination of the previously-defined properties, we can define two main attack settings, as detailed below.

White-Box Attacks. The attacker has complete knowledge about the targeted system. Although not always representative of practical cases, this setting allows us to perform a worst-case analysis, and it is particularly helpful for evaluating defenses.

Black-Box Attacks. Black-box attacks can be subdivided into two main categories: black-box *transfer* attacks, and black-box *query* attacks. Although generally referred to as a black-box attack, *black-box transfer attacks* assume that the

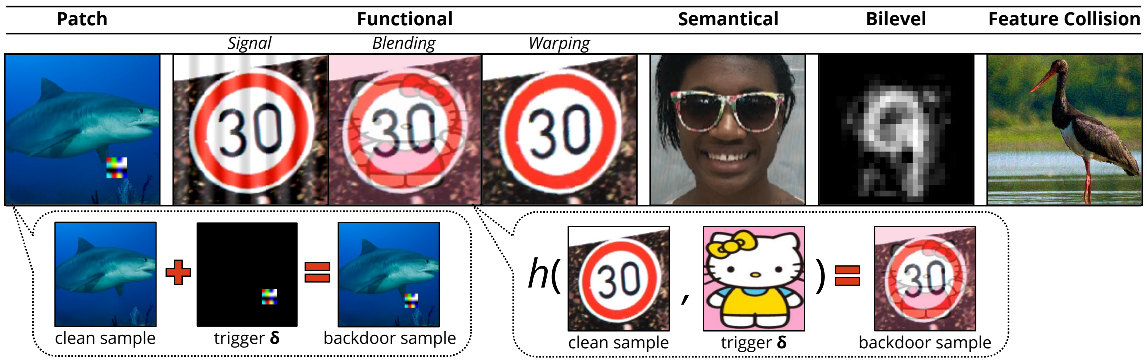


Fig. 2. Visual examples of data perturbation noise (δ) categories. The first five figures show some examples of patch, functional, and semantical triggers. For functional triggers we consider signal [8], blending [33], and warping [129] transformations. The remaining two depict poisoning samples crafted with a bilevel attack with visible noise, and a clean-label feature collision attack with imperceptible noise. The second row shows the backdoor image generation process with patch and functional blending triggers. For the latter, a h manipulation function blends the original image and the backdoor trigger according to a certain ratio.

attacker has partial knowledge of the training data and/or the target model. In particular, the attacker is assumed to be able to collect a surrogate dataset and use it to train a surrogate model approximating the target. Then, white-box attacks can be computed against the surrogate model, and subsequently *transferred* against the target model. Under some mild conditions, such attacks have been shown to transfer successfully to the target model with high probability [45]. It is also worth remarking that *black-box query attacks* can also be staged against a target model, by only sending input queries to the model and observing its predictions to iteratively refine the attack, without exploiting any additional knowledge [31, 132, 167]. However, to date, most of the poisoning attacks staged against learning algorithms in black-box settings exploit surrogate models and attack transferability.

2.2.3 Attacker's Capability. The attacker's capability is defined in terms of how the attacker can influence the *learning setting*, and on the *data perturbation* that can be applied to training and/or test samples.

Influence on Learning Setting. The three learning settings described in Sect. 2.1 open the door towards different data poisoning attacks. In both *training-from-scratch* (TS) and *fine-tuning* (FT) scenarios, the attacker alters a subset of the training dataset collected and used by the victim to train or fine-tune the machine learning model. Conversely, in the *model-training* (MT) scenario, as firstly hypothesized by Gu et al. [70], the attacker acts as a malicious third-party trainer, or as a man-in-the-middle, controlling the training process. The attacker tampers with the training procedure and returns to the victim user a model that behaves according to their goal. The advantage for the attacker is the victim will never be aware of the training dataset actually used. However, to keep their attack stealthy, the attacker must ensure that the provided model retains high prediction accuracy, making sure to pass the validation phase without suspicion from the victim user. The attacker's knowledge, discussed in Sect. 2.2.2, is defined depending on the setting under consideration. In the *model-training* and *training-from-scratch* settings, $\mathcal{D}'$ and $\mathcal{M}$ refer to the training data and algorithm used for training the model from random initialization of its weights. Conversely, in the *fine-tuning* setting, $\mathcal{D}'$ and $\mathcal{M}$ refer to the fine-tuning dataset and learning algorithm, respectively.

Data Perturbation. Staging a poisoning attack requires the attacker to manipulate a given fraction (p) of the training data. In some cases, i.e., in backdoor attacks, the attacker is also required to manipulate the test samples that are under their control, by adding an appropriate trigger to activate the previously-implanted backdoor at test time. More specifically, poisoning attacks can alter a fraction of the training labels and/or apply a (different) perturbation to each of the training (poisoning) samples. If the attack only modifies the training labels, but it does not perturb any training sample, it is often referred to as a *label-flip* poisoning attack. Conversely, if the training labels are not modified (e.g., if they are validated or assigned by human experts or automated labeling procedures), the attacker can stage a so-called *clean-label* poisoning attack. Such attacks only slightly modify the poisoning samples, using imperceptible perturbations that preserve the original semantics of the input samples along with their class labels [148]. We define the strategies used to manipulate training and test data in poisoning attacks in the next section.

2.2.4 Attack Strategy. The attack strategy defines how the attacker manipulates data to stage the desired poisoning attack. Both indiscriminate and targeted poisoning attacks only alter the training data, while backdoor attacks also require embedding the trigger within the test samples to be misclassified. We revise the corresponding data manipulation strategies in the following.

Training Data Perturbation ($\boldsymbol{\delta}$). Two main categories of perturbation have been used to mount poisoning attacks. The former includes perturbations which are found by solving an optimization problem, either formalized as a *bilevel* (BL) programming problem, or as a *feature-collision* (FC) problem. The latter involves the manipulation of training samples in targeted and backdoor poisoning attacks such that they collide with the target samples in the given representation space, to induce misclassification of such target samples in an attacker-chosen class. When it comes to backdoor attacks, three main types of triggers exist, which can be applied to training samples to implant the backdoor during learning: *patch triggers* (T^P), which consist of replacing a small subset of contiguous input features with a patch pattern in the input sample; *functional triggers* (T^F), which are embedded into the input sample via a blending function; and *semantical triggers* (T^S), which perturb the given input while preserving its semantics (e.g., modifying face images by adding sunglasses, or altering the face expression, but preserving the user identity). The choice of this strategy plays a fundamental role since it influences the computational effort, effectiveness, and stealthiness of the attack. More concretely, the trigger strategies are less computationally demanding, as they do not require optimizing the perturbation, but the attack may be less effective and easier to detect. Conversely, an optimized approach can enhance the effectiveness and stealthiness of the attack, at the cost of being more computationally demanding. In Fig. 2 we give some examples of patch, functional, and semantical triggers, one example of a poisoning attack optimized with bilevel programming, and one example of a *clean-label* feature-collision attack.

Test Data Perturbation ($\boldsymbol{t}$). During operation, i.e., at test time, the attacker can submit malicious samples to exploit potential vulnerabilities that were previously implanted during model training, via a backdoor attack. In particular, as we will see in Sect. 3.3, backdoor attacks are activated when a specific trigger $\boldsymbol{t}$ is present in the test samples. Normally, the test-time trigger is required to exactly match the trigger implanted during training, thus including patch, functional, and semantical triggers.

2.3 Defense Framework

In this section, we introduce the main strategies that can be used to counter poisoning attacks, based on different assumptions made on the defender's goal, knowledge and capability.

2.3.1 Defender's Goal. The defender's goal is to preserve the integrity, availability, and privacy of their ML model, i.e., to mitigate any kind of security violation that might be caused by an attack. The defender thus adopts appropriate countermeasures to alleviate the effect of possible attacks, without significantly affecting the behavior of the model for legitimate users.

2.3.2 Defender's Knowledge and Capability. The defender's knowledge and capability determine in which learning setting a defense can be applied. We identify four aspects that influence how the defender can operate to protect the model: (i) having access to the (poisoned) training data $\mathcal{D}'$, and to (ii) a separate, clean validation set $\mathcal{V}$, and (iii) having control on the training procedure $\mathcal{W}$, and on (iv) the model's parameters $\boldsymbol{\theta}$. We will see in more detail how these assumptions are matched to each defense in Sect. 4.

2.3.3 Defense Strategy. The defense strategy defines how the defender operates to protect the system from malicious attacks before deployment (i.e., at training time), and after the model's deployment (i.e., at test time). We identify six distinct categories of defenses:

(1) *training data sanitization*, which aims to remove potentially-harmful training points before training the model;
(2) *robust training*, which alters the training procedure to limit the influence of malicious points;
(3) *model inspection*, which returns for a given model whether it has been compromised (e.g., by a backdoor attack);
(4) *model sanitization*, which cleans the model to remove potential backdoors or targeted poisoning attempts;
(5) *trigger reconstruction*, which recovers the trigger embedded in a backdoored network; and
(6) *test data sanitization*, which filters potentially-triggered samples presented at test time.

These defenses essentially work by either (i) cleaning the data or (ii) modifying the model. In the former case, the defender aims to sanitize training or test data. *Training data sanitization* and *test data sanitization* as thus two strategies adopted respectively at training and at test time to mitigate the influence of data poisoning attacks. Alternatively, the defender can act directly on the model, by (i) identifying possible internal vulnerabilities and removing/fixing components that lead to anomalous behavior/classifications, or by (ii) changing the training procedure to make the model less susceptible to training data manipulations. The first approach is employed in *model inspection*, *trigger reconstruction* and *model sanitization* defensive mechanisms. The second approach, instead, includes algorithms that operate at the training level to implement *robust training* mechanisms.

2.4 Poisoning Attacks and Defenses

We provide in Fig. 3 a preliminary, high-level categorization of attacks and defenses according to our framework (while leaving a more complete categorization of each work to Tables 2-3, respectively for attacks and defenses). This simplified taxonomy categorizes attacks and defenses based on whether they are applied at training time (and in which learning setting) or at test time; whether the attack aims to violate integrity or availability;[6] and whether the defense aims to sanitize data or modify the learning algorithm/model. As one may note, indiscriminate and targeted poisoning only manipulate data at training time to violate availability and integrity, respectively, and they are typically staged in the *training-from-scratch* (TS) or *fine-tuning* (FT) learning settings. Backdoor attacks, in addition, require manipulating the test data to embed the trigger and cause the desired misclassifications, with the goal of violating integrity. Such attacks can be ideally staged in any of the considered learning settings. For defenses, data sanitization strategies can be applied either at training time or at test time, while defenses that modify the learning algorithm or aim to sanitize the model can be applied clearly only at training time (i.e., before model deployment). To conclude, while being simplified, we do believe that this conceptual overview of attacks and defenses provides a comprehensive understanding of the main

[6]To our knowledge, no poisoning attack violating a model's privacy has been considered so far, so we omit the privacy dimension from this representation.

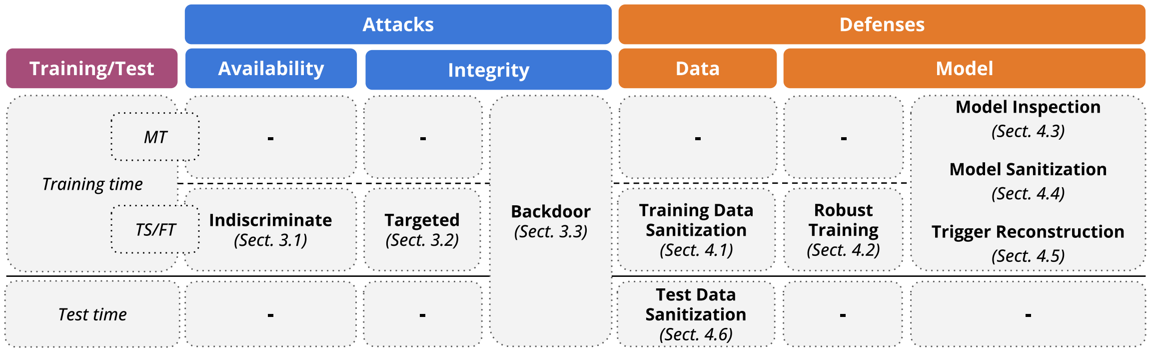
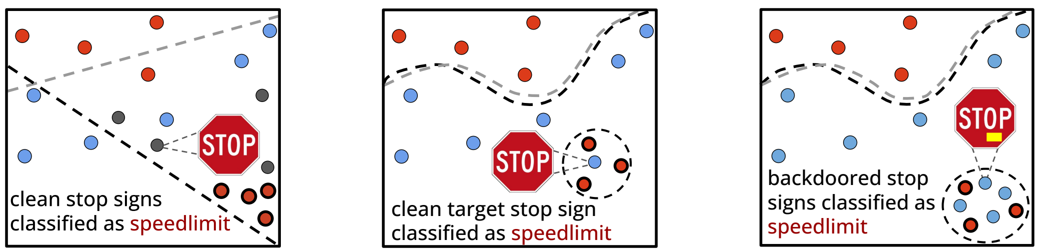


Fig. 3. Conceptual overview of poisoning attacks and defenses according to our framework. Attacks are categorized based on whether they compromise system integrity or availability. Defenses are categorized based on whether they sanitize data or modify the learning algorithm/model. Training-time (test-time) defenses are applied before (after) model deployment. Training-time interventions are also divided according to whether *model-training* (MT) is outsourced, or *training-from-scratch* (TS) / *fine-tuning* (FT) is performed.



(a) Indiscriminate attack. (b) Targeted attack. (c) Backdoor attack.

Fig. 4. Conceptual representation of the impact of indiscriminate, targeted, and backdoor poisoning on the learned decision function. We depict the feature representations of the *speed limit* sign (red dots) and *stop signs* (blue dots). The poisoning samples (solid black border) change the original decision boundary (dashed gray) to a poisoned variant (dashed black).

assumptions behind each poisoning attack and defense strategy. Accordingly, we are now ready to delve into a more detailed description of attacks and defenses in Sects. 3 and 4.

3 ATTACKS

We now take advantage of the previous framework to give an overview of the existing attacks according to the corresponding violation and strategy. A compact summary of all attacks from the vision domain is given in Table 2.

3.1 Indiscriminate (Availability) Poisoning Attacks

Indiscriminate poisoning attacks represent the first class of poisoning attacks against ML algorithms. The attacker aims to subvert the system functionalities, compromising its availability for legitimate users by poisoning the training data. More concretely, the attacker's goal is to cause misclassification on clean validation samples by injecting new malicious samples or perturbing existing ones in the training dataset. In Fig. 4a we consider the case where an attacker poisons a linear street-sign classifier to have stop signs misclassified as speed limits. The adversary injects poisoning samples to rotate the classifier's decision boundary, thus compromising the victim's model performance. In the following, we present the strategies developed in existing works, and we categorize them in Table 2. Although they could also operate

Table 2. Taxonomy of existing poisoning attacks, according to the attack framework defined in Sect. 2. The presence of the ✓ indicates that the corresponding properties is satisfied by the attack. For the attacker's knowledge we use: ○ when the attacker has knowledge of the corresponding component; ◐ if the attacker uses a surrogate to mount the attack; ● if the attacker does not require that knowledge. In the attacker's capabilities we use MT, TS and FT as acronyms for *model-training*, *training-from-scratch*, and *fine-tuning* learning settings. ◔, ◑, ⬤ represent the amount of poisoning: small (≤ 10%), medium (≤ 30%), or high percentage of the training set. The columns δ and t define the training and test strategies: optimized bilevel – BL, feature collision – FC and trigger – T.

Section		Attacks	Goal		Knowledge			Capability			Strategy		Model
			Sample Specific	Error Specific	$\mathcal{D}$	$\mathcal{M}$	$\mathcal{T}$	Setting	Clean Label	p	δ	t	DNN
Indiscriminate	3.1.1	Biggio et al. [15], Xiao et al. [190],			○	○		TS		◑	LF	-	
		Xiao et al. [189], Paudice et al. [133]			○	○		TS		◑			
	3.1.2	Biggio et al. [16], Xiao et al. [188] Frederickson et al. [58]			○	○		TS		◑			
		BetaPoison [42], Ma et al. [116]		✓	○	◐		TS		◑	BL	-	
		Demontis et al. [45], Solans et al. [153]			◐	◐		TS		◑			
		Muñoz-González et al. [124], Yang et al. [194]			◐	◐		TS		◑			✓
	3.1.3	Mei and Zhu [121]		✓	○	○		TS	✓	⬤			
		Feng et al. [53]		✓	◐	◐		TS	✓	⬤	BL	-	✓
		Fowl et al. [55]		✓	◐	◐		TS	✓	⬤			✓
Targeted	3.2.1	Koh and Liang [92]	✓	✓	○	○	✓	FT		◔			
		Muñoz-González et al. [124]	✓	✓	◐	◐		TS		◔	BL	-	✓
		Jagielski et al. [84]		✓	◐	◐		TS/FT		◔			✓
	3.2.2	PoisonFrog [148]	✓	✓	○	○	✓	FT	✓	◔			✓
		Guo and Liu [72], StingRay[158] ConvexPolytope [212], BullseyePolytope [2]	✓	✓	◐	◐	✓	FT	✓	◔	FC	-	✓
	3.2.3	Geiping et al. [62], MetaPoison [80]	✓	✓	◐	◐	✓	TS	✓	◔	BL	-	✓
Backdoor	3.3.1	BadNet [70], LatentBackdoor [196]	✓	✓	○	○		MT		◔			✓
		BaN [143]	✓	✓	○	○		MT		◑	T^P	T^P	✓
		TrojanNN [110]	✓	✓	●	○		MT		◔			✓
	3.3.1	WaNET [129], Li et al. [99], DFST [36]	✓	✓	○	○		MT		◑			✓
		Refool [111]	✓	✓	○	○		TS	✓	◑	T^F	T^F	✓
		SIG [8]	✓	✓	●	●		TS	✓	◑			✓
		Chen et al. [33], Zhong et al. [211]	✓	✓	●	●		TS/FT		◔			✓
	3.3.1	FaceHack [145]	✓	✓	○	○		MT		◑			✓
		Chen et al. [33]	✓	✓	●	●		TS/FT		◔	T^S	T^S	✓
		Wenger et al. [179]	✓	✓	○	●		FT		◑			✓
	3.3.2	Nguyen and Tran [128], LIRA [48]	✓	✓	○	○		MT		◑			✓
		Li et al. [99]	✓	✓	●	○		MT		◔	BL	T^F	✓
		Li et al. [101]	✓	✓	○	◐		TS		◑			✓
		Zhong et al. [211]	✓	✓	◐	◐		TS/FT		◔			✓
	3.3.3	HiddenTrigger [142]	✓	✓	○	○		FT	✓	◑	FC	T^P	✓
		Turner et al. [169]	✓	✓	◐	◐		TS	✓	◔			✓
	3.3.4	Souri et al. [156]	✓	✓	◐	◐		TS	✓	◔	BL	T^P	✓

on the *fine-tuning* (FT) scenario, existing works have been proposed only in the *training-from-scratch* (TS) setting. By contrast, their application in the *model-training* (MT) scenario would not be feasible, as the model, with reduced accuracy due to the attack, would not pass the user validation phase. Indiscriminate attacks, to be adaptable in the latter scenario, must compromise the availability of the system but not in terms of increasing the classification error. This has been recently done by Cinà et al. [38], who proposed a so-called *sponge* poisoning attack aimed to increase the model's prediction latency.

3.1.1 Label-Flip Poisoning. The most straightforward strategy to stage poisoning attacks against ML is label-flip, originally proposed by Biggio et al. [15]. The adversary does not perturb the feature values, but they mislabel a subset of samples in the training dataset, compromising the performance accuracy of ML models such as Support Vector Machines (SVMs). Beyond that, Xiao et al. [190] showed that random flips could have far-from-optimal performance, which nevertheless would require solving an NP-hard optimization problem. Due to its intractability, heuristic strategies have been proposed by Xiao et al. [190], and later by Xiao et al. [189], to efficiently approximate the optimal formulation.

3.1.2 Bilevel Poisoning. In this case, the attacker manipulates both the training samples and their labels. The pioneering work in this direction was proposed by Biggio et al. [16], where a gradient-based indiscriminate poisoning attack is exploited against SVMs. They exploited *implicit differentiation* to derive the gradient required to optimize the poisoning samples by their iterative algorithm. Until convergence, the poisoning samples are iteratively updated following the implicit gradient, directing towards maximization of the model's validation error. Mathematically speaking, this idea corresponds to treating the poisoning task as a bilevel optimization problem:

$$\max_{\boldsymbol{\delta} \in \Delta} \quad L(\mathcal{V}, \mathcal{M}, \boldsymbol{\theta}^\star)\,, \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{\theta}^\star \in \arg\min_{\boldsymbol{\theta}} \; \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{\boldsymbol{\delta}}, \mathcal{M}, \boldsymbol{\theta})\,. \tag{2}$$

with Δ being the set of admissible manipulation of the training samples that preserve the constraints imposed by the attackers (e.g., ℓ_p, or box-constraints)[7]. We define with $\mathcal{D}_p = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ the training data controlled by the attacker, before any perturbation is applied, being y_i the pristine label of sample $\boldsymbol{x}_i$ and n the number of samples in $\mathcal{D}_p$. We then denote with $\mathcal{D}_p^{\boldsymbol{\delta}}$ the corresponding poisoning dataset manipulated according to the perturbation parameter $\boldsymbol{\delta}$. The attacker optimizes the perturbation $\boldsymbol{\delta}$ (applied to the poisoning samples $\mathcal{D}_p$) to increase the error/loss L of the target model $\mathcal{M}$ on the clean validation samples $\mathcal{V}$. Our formulation in Eqs. (1)-(2) encompass both dirty or clean-label attacks according to the nature of $\mathcal{D}_p^{\boldsymbol{\delta}}$. For example, we can define $\mathcal{D}_p^{\boldsymbol{\delta}} = \{(\boldsymbol{x}_i + \boldsymbol{\delta}_i, y_i')\}_{i=1}^n$ [8], being y_i' the poisoning label chosen by the attacker, with $y_i' = y_i$ for a clean-label attack and $y_i' \neq y_i$ for a dirty-label attack. Solving this bilevel optimization is challenging, since the inner and the outer problems in Eqs. (1)-(2) have conflicting objectives. More concretely, the inner objective is a regularized empirical risk minimization, while the outer one is empirical risk maximization, both considering data from the same distribution. A similar approach was later generalized in Xiao et al. [188] and Frederickson et al. [58] to target feature selection algorithms (i.e., LASSO, ridge regression, and elastic net). Subsequent work tried to analyze the robustness of ML models when the attacker has limited knowledge about the training dataset or the victim's classifier. In this scenario, the most investigated methodology is given by the *transferability* of the attack [45, 116, 153]. The attacker crafts the poisoning samples using surrogate datasets and/or models, and then transfers the attack to another target model. This approach has proven effective for corrupting logistic

[7] For example, the attacker can constraint the perturbation magnitude of $\boldsymbol{\delta}$ imposing $\|\boldsymbol{\delta}\|_p \leq \epsilon$ with $\Delta = \{\boldsymbol{\delta} \in \mathbb{R}^{n \times d} \mid \|\boldsymbol{\delta}\|_p \leq \epsilon\}$.
[8] In this example we used $\boldsymbol{\delta}$ as additive noise. To be more generic we can define a manipulation function h parametrized by $\boldsymbol{\delta}$ and the sample $\boldsymbol{x}$ to perturb. See example in Fig. 2 for functional blending trigger.

classifiers [45], algorithmic fairness [153], and differentially-private learners [116]. More details about the transferability of poisoning attacks are reported in Sect. 3.5.

Differently from previous work, Cinà et al. [42] observed that a simple heuristic strategy, together with a variable reduction technique, can reach noticeable results against linear classifiers, with increased computational efficiency. More concretely, the authors showed how previous gradient-based approaches can be affected by several factors (e.g., loss landscape) that degrade their performance in terms of computation time and attack efficiency.

Although effective, the aforementioned poisoning attacks have been designed to fool models with a relatively small number of parameters. More recently, Muñoz-González et al. [124] showed that devising poisoning attacks against larger models, such as convolutional neural networks, can be computationally and memory demanding. To this end, Muñoz-González et al. [124] pioneered the idea to adapt hyperparameter optimization methods, which aims to solve bilevel programming problems more efficiently, in the context of poisoning attacks. The authors indeed proposed a back-gradient descent technique to optimize poisoning samples, drastically reducing the attack complexity. The underlying idea is to back-propagate the gradient of the objective function to the poisoning samples while learning the poisoned model. However, they assume the objective function is sufficiently smooth to trace the gradient backward correctly. Accordingly with the results in [124], Yang et al. [194] showed that computing the analytical or estimated gradient of the validation loss in Eq. (1) with respect to the poisoning samples can be as well computational and query expensive. Another way explored in Yang et al. [194] was to train a generative model from which the poisoning samples are generated, thus increasing the generation rate.

3.1.3 Bilevel Poisoning (Clean-Label). Previous work examined in Sect. 3.1.2 assumes that the attacker has access to a small percentage of the training data and can alter both features and labels. Similar attacks have been staged by assuming that the attacker can control a much larger fraction of the training set, while only slightly manipulating each poisoning sample to preserve its class label, i.e., performing a clean-label attack. This idea was introduced by Mei and Zhu [121], who considered manipulating the whole training set to arbitrarily define the importance of individual features on the predictions of convex learners. More recently, DeepConfuse [53] and Fowl et al. [55] proposed novel techniques to mount clean-label poisoning attacks against DNNs. In [53], the attacker trains a generative model, similarly to [194], to craft clean-label poisoning samples which can compromise the victim's model. Inspired by recent developments proposed in [62], Fowl et al. [55] used a gradient alignment optimization technique to alter the training data imperceptibly, but diminishing the model's performance. Even though Feng et al. [53] and Fowl et al. [55] can target DNNs, the attacker is assumed to perturb a high fraction of samples in the training set. We do believe that this is a very demanding setting for poisoning attacks. In fact, such attacks are often possible because ML is trained on data collected in the wild (e.g., labeled through tools such as a mechanical Turk) or crowdsourced from multiple users; thus, it would be challenging for attackers in many applications to realistically control a substantial fraction of these training data. In conclusion, the quest for scalable, effective, and practical indiscriminate poisoning attacks on DNNs is still open. Accordingly, it remains also unclear whether DNNs can be significantly subverted by such attacks in practical settings.

3.2 Targeted (Integrity) Poisoning Attacks

In contrast to indiscriminate poisoning, targeted poisoning attacks preserve the availability, functionality and behavior of the system for legitimate users, while causing misclassification of some specific target samples. Similarly to indiscriminate poisoning, targeted poisoning attacks manipulate the training data but they do not require modifying the test data.

An example of a targeted attack is given in Fig. 4b, where the classifier's decision function for clean samples is not significantly changed after poisoning, preserving the model's accuracy. However, the model isolated the target stop sign (grey) to be misclassified as a speed-limit sign. The system can still correctly classify the majority of clean samples, but outputs wrong predictions for the target stop sign.

In the following sections, we describe the targeted poisoning attacks categorized in Table 2. Notably, such attacks have been investigated both in the *training-from-scratch* (TS) and *fine-tuning* (FT) settings, defined in Sect. 2.1.

3.2.1 *Bilevel Poisoning.*

In Sect. 3.1.2, we reviewed the work in Muñoz-González et al. [124]. In addition to indiscriminate poisoning, the authors also formulated targeted poisoning attacks as:

$$\min_{\boldsymbol{\delta} \in \Delta} \quad L(\mathcal{V}, \mathcal{M}, \boldsymbol{\theta}^\star) + L(\mathcal{V}_t, \mathcal{M}, \boldsymbol{\theta}^\star)\,, \tag{3}$$

$$\text{s.t.} \quad \boldsymbol{\theta}^\star \in \arg\min_{\boldsymbol{\theta}} \; \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{\boldsymbol{\delta}}, \mathcal{M}, \boldsymbol{\theta})\,. \tag{4}$$

Within this formulation, the attacker optimizes the perturbation $\boldsymbol{\delta}$ on the poisoning samples $\mathcal{D}_p$ to have a set of target (validation) samples $\mathcal{V}_t$ misclassified, while preserving the accuracy on the clean (validation) samples in $\mathcal{V}$. It is worth remarking here that the attack is optimized on a set of validation samples, and then evaluated on a separate set of test samples. The underlying rationale is that the attacker can not typically control the specific realization of the target instances at test time (e.g., if images are acquired from a camera sensor, the environmental and acquisition conditions can not be controlled), and the attack is thus expected to generalize correctly to that case.

A similar attack was introduced by Koh and Liang [92], to show the equivalence between gradient-based (bilevel) poisoning attacks and influence functions, i.e., functions defined in the area of robust statistics that identify the most relevant training points influencing specific predictions. Notably, these authors were the first to consider the *fine-tuning* (FT) scenario in their experiments, training the classification function f (i.e., an SVM with the RBF kernel) on top of a feature representation ϕ extracted from an internal layer of a DNN. Although these two bilevel optimization strategies have been proven effective, they remain too computationally demanding to be applied to DNNs.

Jagielski et al. [84] showed how to generalize targeted poisoning attacks to an entire subpopulation in the data distribution, while reducing the computational cost. To create subpopulations, the attacker selects data samples by matching their features or clustering them in feature space. The poisoning attack can be performed either by label flipping, or linearizing the influence function to approximate the poisoning gradients, thus reducing the computational cost of the attack. Muñoz-González et al. [124] and Jagielski et al. [84] define a more ambitious goal for the attack compared to Koh and Liang [92], as their attacks aim to generalize to all samples coming from the target distribution or the given subpopulation. Specifically, the attack by Koh and Liang [92] is tailored for misleading the model only for some specific test samples, which means considering the test set $\mathcal{T}$ rather than a validation set $\mathcal{V}_t$ in Eq. (3). However, the cost of the attack by Muñoz-González et al. [124] is quite high, due to need of solving a bilevel problem, while the attack by Jagielski et al. [84] is faster, but it does not achieve the same success rate on all subpopulations.

3.2.2 *Feature Collision (Clean-Label).*

This category of attacks is based on a heuristic strategy named *feature collision*, suited to the so-called *fine-tuning* (FT) scenario, which avoids the need of solving a complex bilevel problem to optimize poisoning attacks. In particular, PoisonFrog [148] was the first work proposing this idea, which can be formalized as:

$$\min_{\boldsymbol{\delta} \in \Delta} \quad \|\phi(\boldsymbol{x} + \boldsymbol{\delta}) - \phi(\boldsymbol{z})\|_2^2\,. \tag{5}$$

This attack amounts to creating a poisoning sample $\boldsymbol{x} + \boldsymbol{\delta}$ that collides with the target test sample $\boldsymbol{z} \in \mathcal{T}$ in the feature space, so that the fine-tuned model predicts $\boldsymbol{z}$ according to the poisoning label associated with $\boldsymbol{x}$. To this end, the adversary leverages the feature extractor ϕ to minimize the distance of the poisoning sample with the target in the feature space. Moreover, the authors observed that, due to the complexity and nonlinear behavior of ϕ, even poisoning samples coming from different distributions can be slightly perturbed in the input space to match the feature representation of the target sample z. To make the poisoning sample look realistic in input space and implement a clean-label attack, the adversarial perturbation $\delta \in \Delta$ is bounded by the attacker in its ℓ_p norm [148] (e.g., $\|\delta\|_2 \leq \epsilon$). Such box constraint can also be implemented as a soft constraint, as originally done by Shafahi et al. [148].[9] Similarly, Guo and Liu [72] adopted *feature collision* to stage the attack, but they extended the attack's objective function to further increase the poisoning effectiveness. Nevertheless, although this strategy turns out to be effective, it assumes that the feature extractor is fixed and that it is not updated during the fine-tuning process. Moreover, StringRay [158], ConvexPolytope [212], and BullseyePolytope [2] observed that when reducing the attacker's knowledge the poisoning effectiveness decreases. These works showed that *feature collision* is not practical if the attacker does not know exactly the details of the feature extractor, as the embedding of poisoning samples may not be consistent across different feature extractors. To mitigate these difficulties, ConvexPolytope [212] and BullseyePolytope [2] optimize the poisoning samples against *ensemble models*, constructing a convex polytope around the target samples to enhance the effectiveness of the attack. The underlying idea is that constructing poisoning samples against ensemble models may improve the attack transferability. The authors further optimize the poisoning samples by establishing a strong connection among all the layers and the embeddings of the poisoning samples, partially overcoming the assumption that the feature extractor ϕ remains fixed.

All these approaches have the property of creating clean-label samples, as first proposed in Shafahi et al. [148], to stay undetected even when the class labels of training points are validated by humans. This is possible as these attacks are staged against deep models, since for these models, small (adversarial) perturbations of samples in the input space correspond to large changes in their feature representations.

3.2.3 Bilevel Poisoning (Clean-Label). Although *feature collision* attacks are effective, they may not result in the optimal accuracy, and they do not minimize the number of poisoned points to change the model's prediction on a single test point. Moreover, they assume that the training process is not significantly changing the feature embedding. Indeed, when the whole model is trained from scratch, these strategies may not work properly as poisoning samples can be embedded differently. Recent developments, including MetaPoison [80] and the work by Geiping et al. [62], tackle the targeted poisoning attack in the *training-from-scratch* (TS) scenario, while ensuring the clean-label property. These approaches are derived from the bilevel formulation in Eqs. (3)-(4), but they exploit distinct and more scalable approaches to target DNNs, and optimize the attack directly against the test samples $\mathcal{T}$ as done in [92]. More concretely, MetaPoison [80] uses a meta-learning algorithm, as done by Muñoz-González et al. [124], to decrease the computational complexity of the attack. They further enhance the transferability of their attack by optimizing the poisoning samples against an ensemble of neural networks, trained with different hyperparameter configurations and algorithms (e.g., weight initialization, number of epochs). Geiping et al. [62] craft poisoning samples to maximize the alignment between the inner loss and the outer loss in Eqs. (3)-(4). The authors observed that matching the gradient direction of malicious examples is an effective strategy for attacking DNNs trained from scratch, even on large training datasets. Although modern *feature collision* or optimized strategies are emerging with notable results for targeted attacks, their performance, especially in black-box settings, still demands further investigation.

[9] The original formulation of feature collision in [148] adopts the ℓ_p constraint as soft constraint up-weighted by a Lagrangian penalty term β, which is basically equivalent to our hard-constraint formulation for appropriate choices of β and ϵ.

3.3 Backdoor (Integrity) Poisoning Attacks

Backdoor poisoning attacks aim to cause an integrity violation. In particular, for any test sample containing a specific pattern, i.e., the so-called *backdoor trigger*, they aim to induce a misclassification, without affecting the classification of clean test samples. The backdoor trigger is clearly known only to the attacker, making it challenging for the defender to evaluate whether a given model provided to them has been backdoored during training or not. In Fig. 4c we consider the case where the attacker provides a backdoored street-sign detector that has good accuracy for classifying street signs in most circumstances. However, the classifier has successfully learned the backdoor data distribution, and will output speed-limit predictions for any stop-sign containing the backdoor trigger. In the following sections, we describe backdoor attacks following the categorization given in Table 2. Notably, such attacks have been initially staged in the *model-training* (MT) setting, assuming that the user outsources the training process to an untrusted third-party service, but they have been then extended also to the *training-from-scratch* (TS) and *fine-tuning* (FT) scenarios.

3.3.1 Trigger Poisoning. Earlier work in backdoor attacks considered three main families of backdoor triggers, i.e., *patch*, *functional*, and *semantical* triggers, as discussed below.

Patch. The first threat vector of attack for backdoor poisoning has been investigated in BadNets [70]. The authors considered the case where the user outsources the training process of a DNN to a third-party service, which maliciously alters the training dataset to implant a backdoor in the model. To this end, the attacker picks a random subset of the training data, blends the backdoor trigger into them, and changes their corresponding labels according to an attacker-chosen class. A similar idea has been investigated further in LatentBackdoor [196] and TrojanNN [110], where the backdoor trigger is designed to maximize the response of selected internal neurons, thus reducing the training data needed to plant the trigger. Additionally, LatentBackdoor [196] designed the trigger to survive even if the last layers are fine-tuned with novel clean data, while TrojanNN [110] does not need access to the training data as a reverse-engineering procedure is applied to create a surrogate dataset. All these attacks assume that the trigger is always placed in the same position, limiting their application against specific defense strategies [5, 28, 155]. To overcome this issue, BaN [143] introduced different backdoor attacks where the trigger can be attached in various locations of the input image. The underlying idea was to force the model to learn the backdoor trigger and make it location invariant.

Functional. The patch strategy is based on the idea that poisoning samples repeatedly present a fixed pattern as a trigger, which may however be detected upon human validation of training samples (in the TS and FT scenarios, at least). In contrast, a functional trigger represents a stealthier strategy as the corresponding trigger perturbation is slightly spaced throughout the image or changes according to the input. Some works assume to slightly perturb the entire image so that those small variations are not detectable by humans, but evident enough to mislead the model. In WaNET [129] warping functions are used to generate invisible backdoor triggers (see Fig. 2). Moreover, the authors enforced the model to distinguish the backdoor warping functions among other pristine ones. In Li et al. [99] *steganography* algorithms are used to hide the trigger into the training data. Specifically, the attacker replaces the least significant bits to contain the binary string representing the trigger. In DFST [36] style transfer generative models are exploited to generate and blend the trigger. However, the aforementioned poisoning approaches assume that the attacker can change the labeling process and that no human inspection is done on the training data. This assumption is then relaxed by Barni et al. [8] and Liu et al. [111], where clean-label backdoor poisoning attacks are considered; in particular, Liu et al. [111] used natural reflection effects as trigger to backdoor the system, while Barni et al. [8] used an invisible sinusoidal signal as backdoor trigger (see Fig. 2). More practical scenarios, where the attacker is assumed to have limited knowledge, have been investigated by Chen et al. [33] and Zhong et al. [211]. In these two works, the authors used the idea of blending fixed patterns to backdoor the model. In the former approach, Chen et al. [33] assume that the attacker blends image patterns into the training data and tunes the blend ratio to create almost invisible triggers, while impacting the backdoor's effectiveness. In the latter, Zhong et al. [211] assume that an invisible grid pattern is generated to increase the pixel's intensity, and its effectiveness is tested in the TS and FT settings.

Semantical. The semantical strategy incorporates the idea that backdoor triggers should be feasible and stealthy. For example, Sarkar et al. [145] used facial expressions or image filters (e.g., old-age, smile) as backdoor triggers against real-world facial recognition systems. At training time, the backdoor trigger is injected into the training data to cause the model to associate a smile filter with the authorization of a user. At test time, the attacker can use the same filter to mislead classification. Similarly, Chen et al. [33] and Wenger et al. [179] tried to poison face-recognition systems by blending physically-implementable objects (e.g., sunglasses, earrings) as triggers.

3.3.2 *Bilevel Poisoning.*

Trigger-based strategies assume that the attacker uses a predefined perturbation to mount the attack. However, an alternative strategy for the attacker is to learn the trigger/perturbation itself to enhance the backdoor effectiveness. To this end, even backdoor poisoning can be formalized as a bilevel optimization problem:

$$\min_{\boldsymbol{\delta} \in \Delta} \quad L(\mathcal{V}, \mathcal{M}, \boldsymbol{\theta}^\star) + L(\mathcal{V}_t^{\boldsymbol{t}}, \mathcal{M}, \boldsymbol{\theta}^\star)\,, \tag{6}$$

$$\text{s.t.} \quad \boldsymbol{\theta}^\star \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{\boldsymbol{\delta}}, \mathcal{M}, \boldsymbol{\theta})\,. \tag{7}$$

Here, the attacker optimizes the training perturbation $\boldsymbol{\delta}$ for poisoning samples in $\mathcal{D}_p$ to mislead the model's prediction for validation samples $\mathcal{V}_t$ containing the backdoor trigger $\boldsymbol{t}$. In contrast to indiscriminate and targeted attacks (in Sect. 3.1 and Sect. 3.2), the attacker injects the backdoor trigger in the validation samples $\boldsymbol{t}$ to cause misclassifications. Additionally, as for targeted poisoning, the error on $\mathcal{V}$ is minimized to preserve the system's functionality.

One way to address this bilevel formulation is to craft optimal poisoning samples using generative models [48, 101, 128], as also done in [194] for indiscriminate poisoning. Nguyen and Tran [128] trained the generative model with a loss that enforces the *diversity* and *noninterchangeable* of the trigger, while LIRA [48]'s generator is trained to enforce effectiveness and invisibility of the triggers. Conversely, Li et al. [101] used a generative neural network steganography technique to embed a backdoor string into poisoning samples. Another way is to perturb training samples with adversarial noise, as done by Li et al. [99] and Zhong et al. [211]. More concretely, in the former approach, the trigger maximizes the response of specific internal neurons, and a regularization term is introduced in the objective function to make the backdoor trigger invisible. In the latter work, the attacker looks for the minimum universal perturbation that pushes any input towards the decision boundary of a target class. The attacker can use this invisible perturbation trigger on any image, inducing the model to misclassify the target class.

3.3.3 *Feature Collision (Clean-Label).*

The backdoor trigger visibility influences the stealthiness of the attack. A backdoor trigger that is too obvious can be easily spotted when the dataset is inspected [142]. However, Hidden Trigger [142] introduced the idea of using the *feature collision* strategy, seen in Sect. 3.2.2 and formulated in Eq. (5), to hide the trigger into natural target samples. Specifically, the attacker first injects a random patch trigger into the training set, and then each poisoning sample is masked via *feature collision*. The resulting poisoning images are visually indistinguishable from the target, and have a consistent label (i.e., they are clean-label), while the test samples with the patch trigger will collide with the poisoning samples in feature space, ensuring that the attack works as expected.

Although the work in [142] implements an effective and stealthy clean-label attack, it is applicable only in the feature extractor ϕ is not updated. Such a limitation is mitigated by Turner et al. [169] who exploit a surrogate latent space, rather than ϕ, to interpolate the backdoor samples, hiding the training-time trigger. Moreover, the attacker can tune the trigger visibility at test time to enhance the attack's effectiveness.

3.3.4 Bilevel Poisoning (Clean-Label). Inspired by recent success of the gradient-alignment technique in [62] for targeted poisoning, Souri et al. [156] exploited the same bilevel-descending strategy to stage clean-label backdoor poisoning attacks in the *training-from-scratch* scenario. Similarly to Saha et al. [142] the training and the test data perturbations are different, enhancing the stealthiness of the attack and making it stronger against existing defenses.

3.4 Current Limitations

Although data poisoning has been widely studied in recent years, we argue here that two main challenges are still hindering a thorough development of poisoning attacks.

3.4.1 Unrealistic Threat Models. The first challenge we formulate here questions some of the threat models considered in previous work. The reason is that such threat models are not well representative of what may happen in many real-world scenarios for the attackers. They are valuable because they allow system designers to test the system's robustness under worst-case scenarios, but their practicability and effectiveness against realistic production systems are unknown. To give an accurate estimate of how poisoning attacks are effective against ML production systems, we should consider assumptions that are less favorable to the attacker. For example, Fowl et al. [55] and Feng et al. [53] assume that the attacker controls almost the entire training dataset to effectively mount an indiscriminate poisoning attack against DNNs. While this may happen in certain hypothesized situations, it is also not quite surprising that a poisoning attack works if the attacker controls a large fraction of the training set. We believe that poisoning attacks that assume that only a small fraction of the training points can be controlled by the attacker are more realistic and, therefore, viable against real production systems. We refer the reader to a similar discussion in the context of federated learning poisoning in [150].

Another limitation of threat models considered for poisoning attacks is that, in some cases, exact knowledge of the *test* samples is implicitly assumed. For example, [148] and [62] optimize a targeted poisoning attack to induce misclassification of few specific *test* samples. In particular, the attack is both optimized and tested using the same *test* samples, differently from work which optimizes the poisoning samples using validation data, and then tests the attack impact on a separate test set [16, 124]. This evaluation setting clearly enables the attack to reach higher success rates, but at the same time, there is no guarantee that the attack will generalize even to minor variations of the considered test samples, questioning its applicability outside of settings in which the attacker has exact knowledge of the test inputs. For instance, the attack may not work as expected in physical domains, where images are acquired by a camera under varying illumination and environmental conditions. In such cases, it is indeed clear that the attacker can not know beforehand the specific realization of the test sample, as they do not control the acquisition conditions. On a similar note, only a few studies on backdoor poisoning have considered real-world scenarios where external factors (such as lighting, camera orientation, etc.) can alter the trigger. Indeed, as done in [148] and [62], most papers consider digital applications where the implanted trigger is nearly unaltered.

In conclusion, although some recent works seem to have improved the effectiveness of poisoning attacks, their assumptions are often not representative of the actual production system or the attacker's settings, limiting their applicability only in the proposed context.

3.4.2 Computational Complexity of Poisoning Attacks. The second challenge we discuss here is related to the solution of the bilevel programming problem used to optimize poisoning attacks. The problem, as analyzed by Muñoz-González et al. [124], is that solving the bilevel formulation with a gradient-based approach requires computing and inverting the

Hessian matrix associated to the equilibrium conditions of the inner learning problem, which scales cubically in time and quadratically in space with respect to the number of model's parameters. Even if one may exploit rank-one updates to the Hessian matrix, and Hessian-vector products coupled with conjugate descent to speed up the computation of required gradients, the approach remains too computationally demanding to attack modern deep models, where the number of parameters is on the order of millions. Nevertheless, it is also true that that solving the bilevel problem is expected to improve the effectiveness of the attack and its stealthiness against defenses. For example, the bilevel strategy approach is the only one at the state of the art which allows mounting an effective attack in the *training-from-scratch* (TS) setting. Other heuristic approaches, e.g., *feature collision*, have been shown to be totally ineffective if the feature extractor ϕ is updated during training [62]. For backdoor poisoning, the recent developments in the literature show that bilevel-inspired attacks are more effective and can better counter existing defenses [48, 128, 156]. Thus tackling the complexity of the bilevel poisoning problem remains a relevant open challenge to ensure a fairer and scalable evaluation of modern deep models against such attacks.

3.5 Transferability of Poisoning Attacks

Transferability is a characteristic of attacks to be effective even against classifiers the attacker does not have full knowledge about. The term transferability was first investigated for adversarial examples in [66, 131, 132]. In case of limited knowledge (i.e., black-box attacks), the attacker can use surrogate learners or training data to craft the attack and transfer it to mislead the unknown target model. Nevertheless, the first to introduce the idea of surrogates for data poisoning attacks were Nelson et al. [126] and Biggio et al. [16]. The authors claimed that if the attacker does not have exact knowledge about the training data, they could sample a surrogate dataset from the same distribution and transfer the attack to the target learner. In subsequent work, Muñoz-González et al. [124] and Demontis et al. [45] analyzed the transferability of poisoning attacks using also surrogate learners, showing that matching the complexity of the surrogate and the target model enhances the attack effectiveness. Transferability has also been investigated when considering surrogate objective functions. More concretely, optimizing attacks against a smoother objective function may find effective, or even better, local optima than the ones of the target function [45, 92, 116, 131]. For example, optimizing a non-differentiable loss can be harder, thus using a smoothed version may turn out to be more effective [92]. More recently, Suciu et al. [158] showed that the attacker can leverage transferability even when the attacker has limited knowledge about the feature representation, at the cost of reducing the attack effectiveness. However, Zhu et al. [212] and Aghakhani et al. [2] independently hypothesize that the stability of *feature collision* attacks is compromised when the feature representation in the representation space is changed. To mitigate this problem, they craft poisoning samples to attack an ensemble of models, encouraging their transferability against multiple networks.

3.6 Unifying Framework

Although the three poisoning attacks are detailed in Sects. 3.1-3.3 aim to cause different violations, they can be described by the following generalized bilevel programming problem:

$$\max_{\boldsymbol{\delta} \in \Delta} \quad \alpha L(\mathcal{V}, \mathcal{M}, \boldsymbol{\theta}^\star) - \beta L(\mathcal{V}_t^{\boldsymbol{t}}, \mathcal{M}, \boldsymbol{\theta}^\star)\,, \tag{8}$$

$$\text{s.t.} \quad \boldsymbol{\theta}^\star \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D} \cup \mathcal{D}_p^{\boldsymbol{\delta}}, \mathcal{M}, \boldsymbol{\theta})\,, \tag{9}$$

The optimization program in Eqs. (8)-(9) aims to accomplish the attacker's goal, considering their capacity of tampering with the training set and knowledge of the victim model, by optimizing the perturbation $\boldsymbol{\delta}$ used to poison the training samples in $\mathcal{D}_p$. Additionally, as in Eqs. (1)-(7), the poisoning noise $\boldsymbol{\delta}$ belongs to Δ which encompass possible domain constraints or feature constraints to improve stealthiness of the attack (e.g., invisibility of the trigger). The test data perturbation $\boldsymbol{t}$ is absent (i.e., $\boldsymbol{t} = \boldsymbol{0}$), for indiscriminate and target poisoning. For backdoor poisoning, $\boldsymbol{t}$ is pre-defined/optimized by the attacker before training, unlike from adversarial examples [14, 66] where the perturbation $\boldsymbol{t}$ is optimized at test time. The coefficients α and β are calibrated according to the attacker's desired violation. We can set: (i) $\alpha = 1(-1)$ and $\beta = 0$ for error-generic (specific) indiscriminate poisoning; (ii) $\alpha = -1$ and $\beta = -1(1)$ for error-specific (generic) targeted poisoning; (iii) $\alpha = -1$ and $\beta = -1(1)$ for error-specific (generic) backdoor poisoning.

In conclusion, although backdoor, indiscriminate and targeted attacks are designed to cause distinct security violations, they can be formulated under a unique bilevel optimization program. Therefore, as we will explore in Sec. 7, solutions for optimizing bilevel optimization programs fast can pave the way towards developing novel effective and stealthy poisoning attacks capable of mitigating the scalability limit of current strategies.

4 DEFENSES

Many defenses have been proposed to mitigate poisoning attacks. In this section, we discuss each of the six defense classes identified in Sect. 2.3. For each group, we review the related learning and defense settings, and the various approaches suggested by prior works. Some defenses can be assigned to several groups. In these cases, we assigned a defense to the most suitable group in terms of writing flow. A compact summary of all defenses is given in Table 3. We further match attack strategies and defenses at training and test time in Table 4. Having reviewed all defense groups, we conclude the section by discussing current defense evaluation issues, outlining three main open challenges.

4.1 Training Data Sanitization

These defenses aim to identify and remove poisoning samples *before training*, to alleviate the effect of the attack. The underlying rationale is that, to be effective, poisoning samples have to be *different* from the rest of the training points. Otherwise, they would have no impact at all on the training process. Accordingly, poisoning samples typically exhibit an outlying behavior with respect to the training data distribution, which enables their detection. The defenses that fall into this category require access to the training data $\mathcal{D}'$, and in a few cases also access to clean validation data $\mathcal{V}$, i.e., to an untainted dataset that can be used to facilitate detection of *outlying* poisoning samples in the training set. No capabilities are required to alter the learning algorithm $\mathcal{W}$ or to train the model parameters $\boldsymbol{\theta}$. Theoretically, these defenses can be applied in all learning settings. We can however not exclude the possibility in the *model-training* setting that the attacker tampers with the data provided, which is beyond the defender's control. We first discuss defenses against indiscriminate poisoning. Paudice et al. [133] target label-flip attacks by using label propagation. As Steinhardt et al. [157] show, the difference between poisons and benign data allows to use outlier detection as a defense. Detection can also be eased by taking into account both features and labels, using clustering techniques for indiscriminate [96, 162] and backdoor/targeted attacks [149]. Backdoor and targeted poisoning attacks can also be detected using outlier detection, where the outlier is determined in the networks' latent features on the potentially tampered data [75, 135, 168]. An orthogonal line of work, by Xiang et al. [183, 186], reconstructs the backdoor trigger and removes samples containing it. As shown in Table 4, training data sanitization has been applied against various attack strategies. Attack strategies that have not been mitigated yet are only indiscriminate clean-label bilevel attacks, semantical trigger backdoors and bilevel backdoors.

Table 3. Overview of defenses in the area of classification. When several approaches are named, we order them according to publication year and alphabetical order of the authors. For each paper we report the defender's knowledge and capability required, consisting in access to the training data $\mathcal{D}'$, clean validation data $\mathcal{V}$ and access to the training procedure $\mathcal{W}$ and the model's parameters θ. ○ indicates that the corresponding knowledge or capability is present, ● that it is not. In θ, ◐ refers to the ability to fine-tune the model. We further denote whether the approach provides a certification (Cert.), or it is suited to deep neural networks (DNN). * intended as forensic tool to determine which points were poisoned in retrospect, not as a defense.

	Defense strategy	Defense	$\mathcal{D}'$	$\mathcal{V}$	$\mathcal{W}$	θ	Cert.	DNN
Indiscriminate	4.1 Training Data Sanitization	Taheri et al. [162]	○	○	●	●		
		Curie [96], Paudice et al. [133], Frederickson et al. [58]	○	●	●	●		
		Sphere / Slab Defense [157]	○	●	●	●	✓	
	4.2 Robust Training	RONI [125], Biggio et al. [15], Demontis et al. [44]	○	●	○	○		
		Sever [46], Jia et al. [86], Rosenfeld et al. [140]	○	●	○	○	✓	
		Hong et al. [77]	○	●	○	○		✓
		(SS-)DPA [97]	○	●	●	○	✓	✓
		Weighted Bagging [13]	○	●	●	○		
		Diff. Private Learners [116], Chen et al. [32] Wang et al. [175]	○	●	●	○	✓	
Backdoor / Targeted	4.1 Training Data Sanitization	Shan et al. [149]*	○	●	○	○		✓
		Spectral Signatures [168], CI [183], Peri et al. [135] RE [186], SPECTRE [75]	○	●	●	●		✓
	4.2 Robust Training	Du et al. [51], Hong et al. [77], Borgnia et al. [21], Geiping et al. [61] ABL [102], Huang et al. [79], Sun et al. [160], Yang et al. [195]	○	●	○	○		✓
		DP-InstaHide [22], RAB [176]	○	●	○	○	✓	✓
	4.3 Model Inspection	AEGIS [155]	●	○	○	○		✓
		Neuroninspect [81], Bajcsy and Majurski [5], L-RED [185]	●	○	●	○		✓
		Activation Clustering [28], Tang et al. [163]	○	●	●	●		✓
		MNTD [193], Litmus patterns [94], AEVA [71]	●	○	●	●		✓
		DeepInspect [30]	●	●	●	●		✓
	4.4 Model Sanitization	I-BAU [199]	●	○	○	○		✓
		Yoshida et al. [197]	○	●	○	◐		✓
		Cheng et al. [35], Zhao et al. [208], ANP [180], CLEAR [214]	●	○	●	○		✓
		Re-training [112]	○	●	●	○		✓
		Liu et al. [107], Neural Attention Distillation [100]	●	○	●	◐		✓
		DeepSweep [200]	○	●	●	◐		✓
Backdoor	4.5 Trigger Reconstruction	ABS [109], Neural Cleanse [173], Shen et al. [151] NEO [170], Hu et al. [78]	●	○	●	○		✓
		MESA [138], Gangsweep [213], Xiang et al. [184] TAD [204], AEVA [71], Xiang et al. [182]	●	○	●	●		✓
		Tabor [73], B3D-SS [50]	●	●	●	●		✓
	4.6 Test Data Sanitization	NNoculation [171]	●	○	○	◐		✓
		Anomaly Detection [112]	○	●	●	○		✓
		Input Preprocessing [112], SentiNet [37], ConFoc [172], CleaNN [85], Februus [47]	●	○	●	○		✓
		STRIP [60]	●	○	●	●		✓
		Li et al. [104], Sarkar et al. [144]	●	●	●	●		✓

4.2 Robust Training

Another possibility to mitigate poisoning attacks is *during training*. The underlying idea is to design a training algorithm that limits the influence of malicious samples and thereby alleviates the influence of the poisoning attack. Intuitively, as reported in Table 3, all of these defenses require access to the training data $\mathcal{D}'$ and none to clean validation data $\mathcal{V}$. Nonetheless, they require altering the learning algorithm $\mathcal{W}$ and access to the model's parameters θ. Hence, robust training can only be implemented when the defender trains the model, e.g., in the *training-from-scratch* or *fine-tuning* setting. To alleviate the effect of indiscriminate poisoning attacks, the training data can be split into small subsets. The high-level idea is that a larger number of poisoning samples is needed to alter all small classifiers. The defender can build such ensembles using bagging [13, 97, 175] or voting mechanisms [86] or a combination thereof [32, 97]. An

Table 4. Matching poisoning attack strategies and defenses. For each defense, we depict on which attack strategy (as defined in Section 3) the defense was evaluated. We mark cells with ▬ if the corresponding defense category have not been investigated so far for the corresponding attack. Conversely, we mark cells with ✗ if corresponding defense has no sense and cannot be applied.

Attack				Defenses					
				Training Time					Test Time
	δ	t	Clean Label	Training Data Sanitization	Robust Training	Model Inspection	Model Sanitization	Trigger Reconstruction	Test Data Sanitization
Indiscr.	LF	-		[96, 133, 162]	[15, 32, 44, 77, 97, 140, 175]	▬	▬	✗	✗
Indiscr.	BL	-		[58, 157]	[13, 86, 116, 125]	▬	▬	✗	✗
Indiscr.	BL	-	✓	▬	▬	▬	▬	✗	✗
Targeted	BL	-		[58]	▬	▬	▬	✗	▬
Targeted	FC	-	✓	[135, 149, 195]	[22, 61, 77, 102]	[163, 214]	[214]	✗	▬
Targeted	BL	-	✓	[149, 195]	[21, 22, 61]	▬	▬	✗	▬
Backdoor	T^P	T^P		[75, 149, 168, 172, 183, 186]	[21, 51, 61, 79, 86, 102, 160, 176, 197]	[5, 28, 30, 71, 78, 94, 151, 155, 163, 182, 185, 193, 204]	[30, 100, 107, 138, 180, 197, 199, 200, 208, 213]	[35, 50, 71, 73, 78, 109, 138, 170, 173, 182, 186]	[37, 47, 60, 104, 137, 144, 171–173, 200]
Backdoor	T^S	T^S		▬	[79, 149]	[71, 151]	[107, 112, 199, 200]	[71]	[112, 171]
Backdoor	T^F	T^F		[75, 186]	[79, 102, 176]	[78, 81, 151, 155, 163, 185, 193]	[100, 180, 199, 200, 213]	[78, 109, 184–186, 213]	[60, 200]
Backdoor	FC	T^P	✓	[75]	[22, 61, 79, 102, 195]	[71, 214]	[100, 180, 213, 214]	[71]	[104]
Backdoor	BL	T^F		▬	▬	[71, 151]	[180]	[213]	[200]
Backdoor	BL	T^P	✓	▬	[195]	▬	▬	▬	▬

alternative approach by Nelson et al. [125] is to exclude a sample from training if it leads to a significant decrease in accuracy when used in training. In addition, Diakonikolas et al. [46] apply techniques from robust optimization and robust statistics, thereby limiting the impact of individual, poisonous points. Alternatively, the influence of poisons can be limited by increasing the level of regularization [15, 44]. The alleviating effect of regularization against backdoors has been described by Carnerero-Cano et al. [25], with a more detailed analysis by Cinà et al. [40]. The latter work shows that hyperparameters related to regularization affect backdoor performance. Backdoor and targeted poisoning attacks can also be mitigated using data augmentations like mix-up [21, 22], or based on the model's gradients wrt. the input [61]. Analogously, the data can be augmented using noise to mitigate indiscriminate [140] and backdoor [176] attacks. Furthermore, differences in the loss between backdoored/targeted and clean data allow to unlearn [102] or identify [195] poisons later in training. Alternatively, a trained preprocessor can alleviate the threat of backdoors [160]. Furthermore, Huang et al. [79] show that pre-training the network unsupervisedly (e.g., without wrong labels) can alleviate backdoors. Finally, in both indiscriminate [77, 116] and backdoor/targeted [22, 51, 77] attacks, the framework of differential privacy can be used to alleviate the effect of poisoning. The intuition behind this approach is that differential privacy limits the impact individual data points have, thereby limiting the overall impact of outlying poisoning samples too [77]. However, further investigation is still required to defend against some bilevel strategies, as visible in Table 4.

4.3 Model Inspection

Starting with model inspection, we discuss groups of defenses operating before the model is deployed. The approaches in these groups mitigate only backdoor and targeted attacks. In model inspection, we determine for a given model whether a backdoor is implanted or not. The defense settings in this group are diverse, and encompass all combinations. In principle, model inspection can be used in all learning settings, where exceptions for specific defenses might apply. To inspect a model can be formulated as classifications tasks. For example, Kolouri et al. [94] and Xu et al. [193] show that crafting specific input patterns and training a meta-classifier on the outputs of a given model computed on such inputs can reveal whether the model is backdoored. Bajcsy and Majurski [5] follow a similar approach, using clean data and a pruned model. A different observation is that when relying on the backdoor trigger to output a class, the network behaves somehow *unusual*: it will rely on normally irrelevant features. Thus, outlier detection can be used. For example, Zhu et al. [214] alternatively search for a set of points that are reliably misclassified to detect feature-collision attacks. To detect backdoors and backdoored models, outlier detection can be used on top of interpretability techniques [81], or latent representations [28, 155, 163]. Alternatively, Xiang et al. [185] show that finding a trigger that is reliably misclassified indicates the model is backdoored. As reported in Table 4, model inspection has primarily been evaluated on backdoor attacks with a predefined trigger strategy.

4.4 Model Sanitization

Once a backdoored model is detected, the question becomes how to sanitize it. Sanitization requires diverse defense settings encompassing all possibilities. Model sanitization often involves (re-)training or fine-tuning. Depending on the exact *model-training* setting, sanitizing the model might be impossible (e.g., if the model is provided as a service accessible only via queries). To sanitize a model, pruning [35, 180], retraining [199], or fine-tuning [107, 112] can be used. Given knowledge of the trigger, Zhu et al. [214] propose to relabel the identified poisoned samples after the trigger is removed. Alternatively, approaches such as data augmentation [200] or distillation [100, 197] can augment small, clean datasets. Finally, Zhao et al. [208] show that path connection between two backdoored models, using a small amount of clean data, also reduces the success of the attack. As shown in Table 4, model sanitization has been evaluated mainly against backdoor attacks. Extensions to other kinds of triggers and targeted attacks might however be possible.

4.5 Trigger Reconstruction

As an alternative to model sanitization, this category of defenses aim to reconstruct the implanted trigger. The assumptions on the defender's knowledge and capabilities are diverse, and encompass many possibilities, although the learning algorithm $\mathcal{W}$ is never altered. As for model inspection, trigger reconstruction can in theory be used in all learning settings, where exceptions for specific defenses might apply. While a trigger can be randomly generated [170, 204], the question remains on how to verify that the reconstructed pattern is a trigger. Many techniques leverage the fact that a trigger changes the classifier's output reliably. This finding has been in detail investigated by Grosse et al. [69], who show that backdoor patterns lead to a very stable or smooth output of the target class. In other words, the classifier ignores other features and only relies on the backdoor trigger. Such a stable output also enables to reformulate trigger reconstruction as an optimization problem [173]. In the first approach of its kind, Wang et al.'s Neural Cleanse [173] optimizes a pattern that leads to reliable misclassification of a batch of input points. The idea is that if there is such a pattern, and it is small, it must be similar to the backdoor trigger. Wang et al.'s approach has been improved in terms of how to determine whether a pattern is indeed a trigger [73, 184], how to decrease runtime for many classes [78, 151, 182], how many triggers can be recovered at once [78], or how to reverse-engineer without computing gradients [50, 71]. Zhu et al. [213] establish that not an optimization, but also a GAN can be used to generate triggers. In general, a reconstruction can be based on the intuition that triggers themselves form distributions that can be learned [138, 213]. Finally, Liu et al. [109] successfully use stimulation analysis of individual neurons to retrieve implanted trigger patterns. Trigger reconstruction has been evaluated on almost all trigger-based backdoor attacks (see Table 4), as their applicability is naturally limited to the existence of a trigger.

4.6 Test Data Sanitization

As the name suggests, this is the only group of defenses operating during *test time*, where the defender attempts to sanitize malicious test inputs. The assumptions on the defender's knowledge and capabilities, as in other cases, are diverse and encompass all possible settings. Test data sanitization can be used in all learning settings, where exceptions for specific cases might apply. This group can, in principle, be applied in all learning scenarios, but is the only sanitization applicable if the model is only available as an online service, and accessible via queries. There are three strategies overall when sanitizing test data. The first one boils down to removing the trigger [37, 47, 104]. For example Chou et al. [37] use interpretability techniques to identify crucial parts of the input and then mask these to identify whether they are adversarial or not. A second group is build on the agreement of ensembles on input [144, 171, 172]. In Sarkar et al. [144], this ensemble results indirectly from noising the input, but can also be build with a second, retrained version of the original model on different styles [172] or augmentations [171]. Finally, and as used for trigger generation, the consistency of a classifier's output can also help to detect an attack [60, 85]. While Gao et al. [60] superimpose images to check the consistency, Javaheripi et al. [85] instead consider the consistency of noised images in the inner layers. As shown in Table 4, test data sanitization has been tested only on trigger-based backdoor attacks. However, the latter two strategies mentioned above do have the potential to also detect targeted poisoning attacks, as these lead to locally implausible behavior. A detection of indiscriminate attacks at test time is however not possible.

4.7 Current Limitations

Although there is a large body of work on defenses, there are still unresolved challenges, as detailed in the following.

4.7.1 Inconsistent Defense Settings. The assumptions on the defender's knowledge and capabilities reflect what is required to deploy a defense. In indiscriminate defenses, or robust training and training data sanitization in general, these are very homogeneous. When it comes to model inspection, trigger reconstruction, model sanitization, and test data sanitization, there is a larger variation in both the defender's knowledge and capabilities. In particular, we lack understanding on the effect of individual capabilities or knowledge, for example not having direct access to the model when provided as a service and interacting via queries. More work is required that enables comparison across approaches here, and that sheds light on the individual components of the defense setting.

4.7.2 Insufficient Defense Evaluations. In Table 4, we match poisoning attack strategies and defenses by reporting in each cell the defense papers that evaluate against the corresponding attack strategy. In some cases, indicated with a cross (✗), a defense of this strategy is not possible as there is no trigger to reconstruct (indiscriminate or targeted) or the test data is not altered by the attacker and can thus not be sanitized (indiscriminate attacks). Furthermore, Table 4 shows that the amount of defenses per attack strategies varies greatly. Whereas for backdoor attacks using patch triggers there are around fifty defenses, only eleven defenses have been considered against semantic triggers, one against bilevel targeted attacks [58], one against bilevel patch backdoor attacks [195], and none against indiscriminate clean

Table 5. Attacks breaking defenses in the areas of indiscriminate, targeted, and backdoor attacks. We provide the reference for the adaptive attack, which defenses are broken, and a high-level description of the strategy of the adaptive attack.

	Broken Defenses			
Attack	Indiscriminate	Targeted	Backdoor	Strategy
Koh et al. [93]	[141, 157]			constrain poison point's features
Shokri et al. [152]			[28, 168, 173]	regularize trigger pattern
Tang et al. [163]			[28, 37, 60, 173]	add trigger images with correct label
Lin et al. [105]			[109, 173]	add trigger images mixed from source and target

label bilevel attacks. With only a few defenses [163, 214], there is also a shortage of model inspection and sanitization defenses when no trigger manifests in the model.

Beyond this shortage, there is a need to thoroughly test existing defenses using adaptive attacks, which are depicted in Table 5. Adaptive attacks are tailor to circumvent one or several defenses. In other words, the attack identifies essential components like for example a threshold within a defense and adapts the poisoning points to be below this threshold. For example, Koh et al. [93] constrain the indiscriminate poisons features so that several points are in close vicinity to avoid outlier detection. In the case of backdoors, Shokri et al. [152] regularize the trigger to be less detectable within the network. Tang et al. [163] and Lin et al. [105] employ different strategies to make training data with trigger more similar to benign data. Yet, as visible in Table 5, current adaptive backdoor attacks tend to break the same defenses. More work is thus needed to understand all defenses' limitations through adaptive attacks, even though systematizing the design of such attacks and automating the corresponding evaluations is not trivial. To this end, it may be interesting to design *indicators of failure* that automate the identification of faulty, non-adaptive evaluations for poisoning attacks, as recently shown in [136] for adversarial examples.

4.7.3 Overly-specialized Defenses. Furthermore, few defenses (only roughly one sixth) have been evaluated against different kinds of triggers. Only one defense in test data sanitization [200] and two defenses in trigger reconstruction [78, 109] have been evaluated against more than one trigger type. There are three defenses for each training data sanitization [75, 149, 186], model sanitization [100, 199, 200] and robust training [22, 61, 102]. In model inspection, five [71, 151, 155, 163, 193] defenses tests on more than one attack type. There are even more general defenses that are able to handle multiple poisoning attacks, such as indiscriminate, targeted, and backdoor attacks, as for example Geiping et al. [61] and Hong et al. [77] show.

5 POISONING ATTACKS AND DEFENSES IN OTHER DOMAINS

While in this survey we focus on poisoning ML in the context of supervised learning tasks, and mostly in computer-vision applications, it is also worth remarking that several poisoning attacks and defense mechanisms have been developed also in the area of federated learning [4, 12, 23, 76, 161, 165, 191, 201–203, 210], regression learning [46, 54, 83, 106, 123, 177], reinforcement learning [3, 6, 10, 52, 74, 82, 89, 115, 139, 174, 192, 205], and unsupervised clustering [17, 18, 41, 90, 141] or anomaly detection [43, 141] algorithms. Furthermore, notable examples of poisoning attacks and defenses have also been shown in computer-security applications dealing with ML, including spam filtering [13, 46, 58, 126, 133], network traffic analysis [141], and malware detection [134, 147, 162], audio [1, 91, 107, 110, 193] and video analysis [168, 209], natural language processing [29, 34, 110, 137, 206], and even in graph-based ML applications [20, 108, 181, 207, 215]. While, for the sake of space, we do not give a more detailed description of such research findings in this survey, we do believe that the systematization offered in our work provides a useful starting point for the interested reader to gain a better understanding of the main contributions reported in these other research areas.

6 RESOURCES: SOFTWARE LIBRARIES, IMPLEMENTATIONS, AND BENCHMARKS

Unified test frameworks play a huge role when evaluating and benchmarking both poisoning attacks and defenses. We thus attempt to give an overview of available resources in this section. Libraries and available code ease the evaluation and benchmarking of both attacks and defenses. Ignoring the many repositories containing individual attacks, to date, only a few libraries provide implementations of poisoning attacks and defenses.[10] The library with the largest number of attacks and defenses is the adversarial robustness toolbox (ART) [130]. ART implements indiscriminate poisoning attacks [16], targeted [2, 62, 148, 169] and backdoor attacks [70, 142], as well as an adaptive backdoor attack [152]. The library further provides a range of defenses [7, 28, 60, 125, 168, 173]. Furthermore, SecML [122] provides indiscriminate poisoning attacks against SVM, logistic, and ridge regression [16, 45, 188]. Finally, the library advBox [67] provides both indiscriminate and backdoor attacks on a toy problem.

Beyond the typical ML datasets that can be used for evaluation, there exists a large database from the NIST competition,[11] which contains a large number of models from image classification, object recognition, and reinforcement learning. Each model is labeled as poisoned or not. The module further allows to generate new datasets with poisoned and unpoisoned models. Schwarzschild et al. [146] recently introduced a framework to compare different poisoning attacks. They conclude that for many attacks, the success depends highly on the experimental setting. To conclude, albeit a huge number of attacks and defenses have been introduced, there is still a need of libraries that allow access to off-the-shelf implementations to compare new approaches. In general, few works benchmark poisoning attacks and defenses or provide guidelines to evaluate poisoning attacks or defenses.

7 DEVELOPMENT, CHALLENGES, AND FUTURE RESEARCH DIRECTIONS

In this section, we outline challenges and future research directions for poisoning attacks and defenses. We start by discussing the intertwined historical development of attacks and defenses, and then highlight the corresponding challenges, open questions, and promising avenues for further research.

7.1 Development Timelines for Poisoning Attacks and Defenses

We start by discussing the historical development of poisoning attacks (represented in Fig. 5), and afterwards that of defenses (depicted in Fig. 6). In both cases, we highlight the respective milestones and development over time.

7.1.1 Attack Timeline. The attack timeline is shown in Fig. 5. To the best of our knowledge, the first example of indiscriminate poisoning was developed in 2006 by Perdisci et al. [134], Barreno et al. [9], and Newsome et al. [127] in the computer security area. Such attacks, as well as subsequent attacks in the same area [90, 141], were based on heuristic approaches to mislead application specific ML models, and there was not a unifying mathematical formulation describing them. It was only later, in 2012, that indiscriminate poisoning against machine learning was formulated for the first time as a bilevel optimization [190], to compute optimal label-flip poisoning attacks. Since then, indiscriminate poisoning has been studied under two distinct settings, i.e., assuming either (i) that a small fraction of training samples can be largely perturbed [16, 45, 124]; or (ii) that all training points can be slightly perturbed [53, 55, 121].

[10] Analysis carried out in June 2022.
[11] https://pages.nist.gov/trojai/docs/index.html

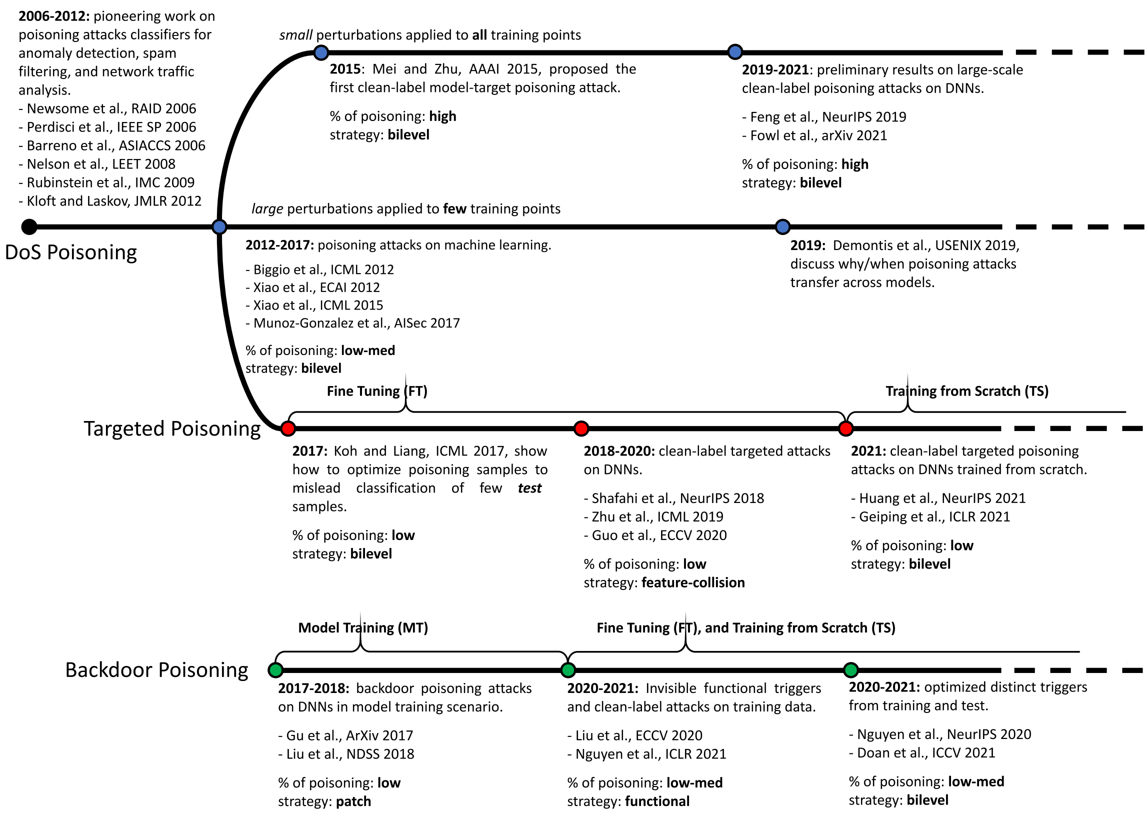


Fig. 5. Timeline for indiscriminate (blue), targeted (red) and backdoor (green) data poisoning attacks on machine learning. Related work is highlighted with markers of the same color and connected with dashed lines to highlight independent (but related) findings.

Targeted and backdoor poisoning attacks only appeared in 2017, and interestingly, they both started from different strategies. Targeted poisoning started with the bilevel formulation in Koh and Liang [92], but evolved in more heuristic approaches, such as feature collision [72, 148, 212]. Only recently, targeted poisoning attacks were reformulated as bilevel problems, given the limitation of the aforementioned heuristic approaches [62, 80]. Backdoor poisoning started with the adoption of patch [70, 110] and functional [111, 128] triggers. However, in the last years, such heuristic choices have been put aside, and backdoor attacks are getting closer and closer to the idea of formulating them in terms of a bilevel optimization, not only to enhance their effectiveness, but also their ability to bypass detection [142, 156].

The historical development of the three types of attacks is primarily aimed at solving or mitigating as much as possible the challenges highlighted in Sect. 3.4, i.e., (i) considering more realistic threat models, and (ii) designing more effective and scalable poisoning attacks. In particular, recent developments in attacks seek to improve the applicability of their threat models, by tampering with the training data as little as possible (e.g., a few points altered with invisible perturbations) to evade defenses, and by considering more practical settings (e.g, *training-from-scratch*). Moreover, more recent poisoning attacks aim to tackle the computational complexity and time required to solve the bilevel problem, not only to improve attack scalability but also their ability to stay undetected against current defenses. In Sect. 7.2 we more thoroughly discuss these challenges, along with some possible future research directions to address them.

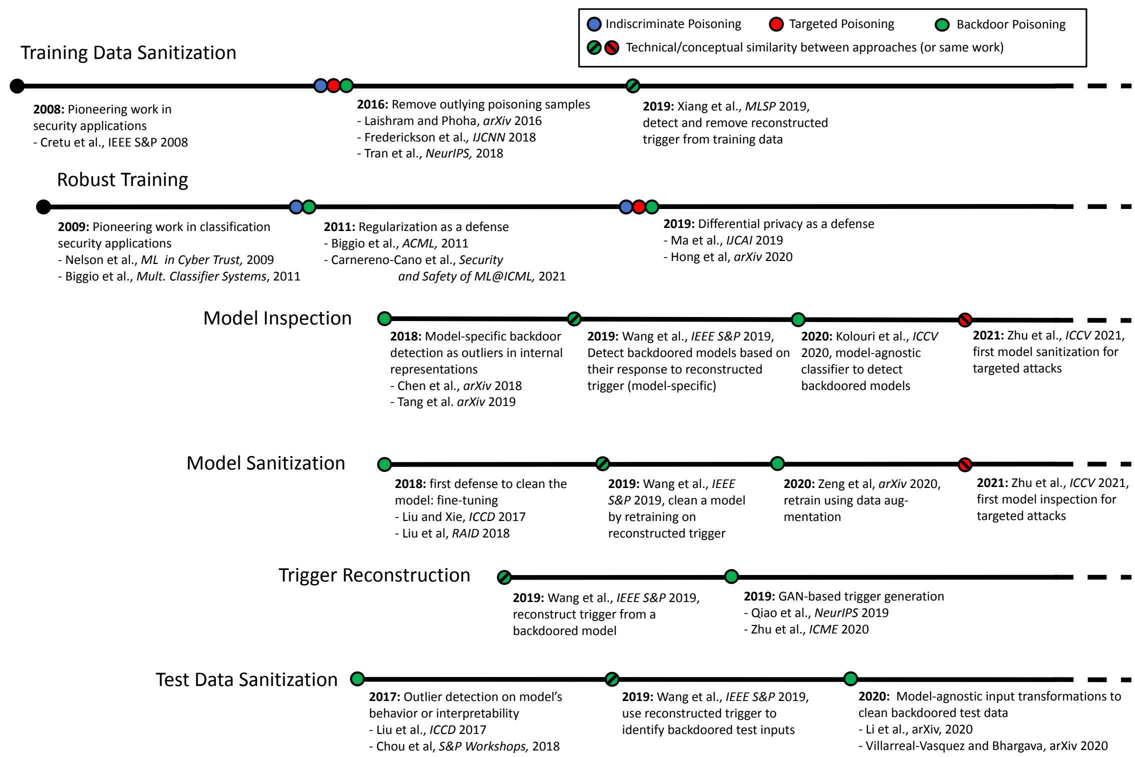


Fig. 6. Timeline of the six kinds of defenses described in Sect. 4. The dots remark against which class of attacks defenses have been introduced, dashed lines denote related approaches, and thin gray lines connect the same work across defense groups.

7.1.2 Defense Timeline. The defense timeline is shown in Fig. 6. The first defenses, training data sanitization and robust training variants, were introduced 2008 and 2009 in a security context [15, 43, 125]. Following works in training data sanitization were based on outlier detection, and mitigated backdoor [168], indiscriminate [96] and targeted [58] attacks. To train robustly, Biggio et al. [13] showed 2011 that regularization can serve as a defense, a finding recently confirmed for backdoors [25]. In 2019, differential privacy was shown to be able to mitigate poisoning attacks [77, 116]. This connection to privacy underlines the need to study poisoning also in relation to other ML security issues, as we will discuss in Sect. 7.2.4. The remaining kinds of defenses are characterized by more diverse threat models, as we discussed in Sect. 4.7.1. The type of attack mitigated is however less diverse, and focuses mainly on backdoors, as explained in Sect. 4.7.3. We start with model inspection approaches, which were first introduced by Chen et al. [28] and were based on outlier detection on latent representations. In 2020, Kolouri et al. [94] generalized the backdoor inspections to be model independent using a meta-classifier. Recently, Zhu et al. [214] introduced a search-based approach to determine whether a model suffers from targeted poisoning. The latter approach also proposed how to sanitize the model. The first defenses for such model sanitization against backdoors were trigger agnostic and based on fine-tuning [107, 112] and later on data augmentation Zeng et al. [200]. Another possibility, introduced by Wang et al. [173], is to retrain a model based on a reconstructed trigger. Wang et al. [173] introduced the idea of reconstruction a trigger in 2019. They generated a trigger based on optimization of a pattern that causes backdoor behavior, e.g., misclassification of many samples when added to them. More recent approaches improve trigger reconstruction by considering distributions over triggers [138], not individual patterns. A reconstructed trigger can also serve to inspect a model [173], or serve to sanitize test data [173]. However, the first approaches to sanitize test data in 2017 were based on outlier detection to inspect the model inspection and sanitize training data. Analogous to model inspection, initial works relied on latent, model specific features [112] whereas later works from 2020 use model-agnostic input transformations [104].

One historical development which is highly relevant but left out in both timeline figures is the study of adaptive attacks against defenses to assess their robustness, as discussed in Sect. 4.7.2. We elaborate on this challenge in Sect. 7.2.3.

7.2 Challenges and Future Work

Building on the development timelines and the corresponding overview provided in Sect. 7.1, we formulate some future research challenges for both poisoning attacks and defenses in the remainder of this section.

7.2.1 Considering Realistic Threat Models. One pertinent challenge arising from the discussion on poisoning attacks in Sect. 3.4.1 demands considering more realistic threat models and attack scenarios, as also recently pointed out in [150]. While assessing machine learning models in real-world settings is not straightforward [154], the need to develop realistic threat models is still an open question in machine learning security and has so far only received recognition for test-time attacks [63]. Here we define some guidelines that can serve as a basis for future work that wants to assess the real safety impact of poisoning versus real applications. First, limit the attacker's knowledge of the target system and their capacity to tamper during training. For example, an attack that assumes only a small percentage of control over the training set can be broadly applied. Second, develop more stealthy poisoning strategies to avoid detection against defenses. Some attack strategies, e.g., patch trigger or feature collision, are computationally efficient, but several defensive countermeasures exist to detect them (see Table 4). Finally, evaluating poisoning attacks against real-world applications and making them adaptive to the presence of a defender. Therefore, we invite the research community to evaluate poisoning attacks with more realistic or less favorable assumptions for the attacker, which also take into account the specific application domain.

7.2.2 Designing More Effective and Scalable Poisoning Attacks. The other challenge we highlighted in Sect. 3.4.2 is the computational complexity of poisoning attacks when relying on bilevel optimization. However, the same limitation is also encountered in other research domains such as hyperparameter optimization and meta-learning which naturally are formulated within the mathematical framework of bilevel programming [57]. More concretely, the former is the process of determining the optimal combination of hyperparameters that maximizes the performance of an underlying learning algorithm. On the other hand, meta-learning encompasses feature selection, algorithm selection, learning to learn, or ensemble learning, to which the same reasoning applies. Having formulated poisoning attacks within the bilevel framework (see Sect. 3.6) hints that strategies developed to speed up the optimization of bilevel programs involved in meta-learning or hyperparameter optimization taks can be adapted to facilitate the development of novel scalable attacks. In principle, by imagining poisoning samples as the attacker-controlled learning hyperparameters, we could apply the approaches proposed in these two fields to mount an attack. Notably, we find some initial works connecting these two fields with data poisoning. For example, Shen et al. [151] rely in their approach on a k-arms technique, a technique similar to bandits, as done by Jones et al. [87]. Further, Muñoz-González et al. [124] exploited the back-gradient optimization technique proposed in [49, 117], originally proposed for hyperparameter optimization, and subsequently, Huang et al. [80] inherited the same approach making the attack more effective against deep neural networks. Apart from the work just mentioned, the connection between the two fields and poisoning is still currently under-investigated, and other ideas could still be explored. For example, the optimization proposed by [114] can further reduce run-time complexity and memory usage even when dealing with millions of hyperparameters. Or another way might be to move away from gradient-based approaches and consider gradient-free approaches, thus overcoming the complexity of the inverting the Hessian matrix seen in Sect. 3.4.2. In the area of gradient-free methods, the most straightforward way is to use grid or random search [11], which can be sped up using reinforcement learning [98]. Also, Bayesian optimization has been used, given a few sampled points from the objective and constraint functions, to approximate the target function [87]. Last but not least, evolutionary algorithms [198] as well as particle swarm optimization [113] have shown to be successful.

In conclusion, we consider these two fields as possible future research directions to find more effective and scalable poisoning attacks for assessing ML robustness in practice.

7.2.3 Systematizing and Improving Defense Evaluations. Regardless of future attacks, we need to systematize and understand the limits of existing (and future) defenses better. As we have seen in Sect. 6, there is no coherent benchmark for defenses. Such a benchmark exposes flawed evaluations and assesses the robustness of a defense per se or in relation to other defenses (taking into account the defense's setting, as discussed in Sect. 4.7.2). Jointly with benchmarks, evaluation guidelines, as discussed for ML evasion by Carlini et al. [24], help to improve defense evaluation. More specifically, these guidelines can encompass knowledge when attacks fail and why, similar to work on evasion attack failure [136]. Crucial in this context is also, as discussed in Sect. 4.7.2, to expand our understanding of adaptive attacks.

An orthogonal question is how to increase existing knowledge about trade-offs between for example attack strength and stealthiness for indiscriminate attacks [58] or backdoors [33, 143, 169]. Further trade-offs relate clean accuracy and accuracy under the poisoning attack by hyperparameter tuning. More concretely, Demontis et al. [45] and Cinà et al. [40] showed that more regularized classifiers tend to resist better to poisoning attacks, at the cost of slightly reducing clean accuracy. Ideally, impossibility results further increase our knowledge about hard limitations. To the best of our knowledge, the only impossibility results provided thus far for subpopulation poisoning attacks can be found in [84], showing that it is impossible to defend poisoning attacks that target only a fraction of the data. Expanding our knowledge about trade-offs and impossibilities will help to design and configure effective defenses.

7.2.4 Designing Generic Defenses against Multiple Attacks. Such effective defenses also need to overcome, as discussed in the Sect. 4.7.3, that they often specialize and to on one or several poisoning attacks (for example backdoor and targeted). Such one-sided evaluations introduce a bias, and the effect of this overfitting on biased datasets has been recognized in image recognition [166], but received relatively little attention in security so far. Some defenses, however, do evaluate several poisoning attacks [61, 77, 151], or even different ML security threats like backdoors and evasion [160] or poisoning and privacy [77].

In addition to creating more robust defenses, such interdisciplinary works also increase our understanding of how poisoning interferes with non-poisoning ML attacks [27, 178]. One attack is evasion, where a small perturbation is added to a sample at test time to force the model to misclassify an output. Evasion is closely related, but different from backdoors, which add a *fixed* perturbation at *training time*, causing an upfront known vulnerability at test time. Only a few works study evasion and poisoning together. For example, Sun et al. [160] introduce a defense against both backdoors and adversarial examples. Furthermore, Fowl et al. [56] show that adversarial examples with the original labels are strong poisons at training time. In the opposite direction, Weng et al. [178] find that if backdoor accuracy is high, evasion tends to be less successful and vice versa. Furthermore, Mehra et al. [120] study poisoning of certified evasion defenses: using poisoning, they decrease the certified radius and accuracy. Two works, namely by Manoj and

Blum [118] and Goldwasser et al. [65], relate evasion and backdoors in a theoretical way. Both share rigid assumptions, however Manoj and Blum [118] show an impossibility results in terms of non-existence of backdoor for some natural learning problems. Goldwasser et al. [65], on the other hand, show that backdoor detection might be impossible. In relation to privacy or intellectual property, poisoning can be used to increase the information leakage from training data at test time on collaborative learning [27]. Privacy can further be a defense against poisoning [22, 77], or poisoning can be a tool to obtain [55]. Summarizing, there is little knowledge on how poisoning interacts with other attacks. More work is needed to understand this relationship and secure machine learning models in practice against several threats at the same time.

8 CONCLUDING REMARKS

The increasing adoption of data-driven models in production systems demands a rigorous analysis of their reliability in the presence of malicious users aiming to compromise them. Within this survey, we systematize a broad spectrum of data poisoning attacks and defenses according to our modeling framework, and we exploit such categorization to match defenses with the corresponding attacks they prevent. Moreover, we provide a unified formalization for poisoning attacks via bilevel programming, and we spotlight resources (e.g., software libraries, datasets) that may be exploited to benchmark attacks and defenses. Finally, we trace the historical development of data literature since the early developments dating back to more than 20 years ago and find the open challenges and possible research directions that can pave the way for future development. In conclusion, we believe our contribution can help clarify what threats an ML system may encounter in adversarial settings and encourage further research developments in deploying trustworthy systems even in the presence of data poisoning threats.

ACKNOWLEDGMENTS

This work has been partly supported by: the PRIN 2017 project RexLearn (grant no. 2017TWNMH2), funded by the Italian Ministry of Education, University and Research; the EU's Horizon Europe research and innovation program under the project ELSA, grant agreement No 101070617; the project "TrustML: Towards Machine Learning that Humans Can Trust", CUP: F73C22001320007, funded by Fondazione di Sardegna; the NRRP MUR program funded by the EU - NGEU under the project SERICS (PE00000014); and the COMET Programme managed by FFG in the COMET Module S3AI, funded by BMK, BMDW, and the Province of Upper Austria.

REFERENCES

[1] Hojjat Aghakhani, Thorsten Eisenhofer, Lea Schönherr, Dorothea Kolossa, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. 2020. VENOMAVE: Clean-Label Poisoning Against Speech Recognition. *arXiv:2010.10682* (2020).

[2] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. 2021. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *EuroS&P*. 159–178.

[3] Chace Ashcraft and Kiran Karra. 2021. Poisoning Deep Reinforcement Learning Agents with In-Distribution Triggers. *arXiv:2106.07798* (2021).

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How To Backdoor Federated Learning. In *The 23rd Int. Conf. on AI and Statistics, AISTATS*. PMLR, 2938–2948.

[5] Peter Bajcsy and Michael Majurski. 2021. Baseline Pruning-Based Approach to Trojan Detection in Neural Networks. *arXiv:2101.12016* (2021).

[6] Kiarash Banihashem, Adish Singla, and Goran Radanovic. 2021. Defense Against Reward Poisoning Attacks in Reinforcement Learning. *arXiv:2102.05776* (2021).

[7] Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Amir Safavi, and Rui Zhang. 2018. Detecting poisoning attacks on ML in iot environments. In *IEEE Int. congress on internet of things, ICIOT 2018*. IEEE, 57–64.

[8] Mauro Barni, Kassem Kallas, and Benedetta Tondi. 2019. A New Backdoor Attack in CNNS by Training Set Corruption Without Label Poisoning. In *2019 IEEE Int. Conf. on Image Proc., ICIP 2019*. IEEE, 101–105.

[9] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. 2006. Can ML be secure?. In *ACM Symposium on Inf., Computer and Communications Security, ASIACCS*. ACM, 16–25.

[10] Vahid Behzadan and Arslan Munir. 2017. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In *ML and Data Mining in Pattern Recognition - 13th Int. Conf., MLDM 2017*. Springer, 262–275.

[11] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of ML research* 13, 2 (2012).

[12] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. 2019. Analyzing Federated Learning through an Adversarial Lens. In *36th Int. Conf. on ML, ICML 2019*. PMLR, 634–643.

[13] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. 2011. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *Int. workshop on multiple classifier Sys.* Springer, 350–359.

[14] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion Attacks against ML at Test Time. In *ML and Knowl. Disc. in Databases - Eur. Conf., ECML PKDD 2013*. Springer, 387–402.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2011. Support Vector Machines Under Adversarial Label Noise. In *ACML2011*. 97–112.

[16] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *ICML 2012*. icml.cc / Omnipress.

[17] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli. 2013. Is data clustering in adversarial settings secure?. In *6th ACM Workshop on Art. Intell. and Sec., AISec 2013*. ACM, 87–98.

[18] Battista Biggio, Konrad Rieck, Davide Ariu, Christian Wressnegger, Igino Corona, Giorgio Giacinto, and Fabio Roli. 2014. Poisoning behavioral malware clustering. In *7th ACM Workshop on Art. Intell. and Sec., AISec 2014*. ACM, 27–36.

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

[20] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *ICML*. 695–704.

[21] 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 *IEEE ICASSP 2021*. IEEE, 3855–3859.

[22] Eitan Borgnia, Jonas Geiping, Valeriia Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. 2021. DP-InstaHide: Provably Defusing Poisoning and Backdoor Attacks with Differentially Private Data Augmentations. *arXiv:2103.02079* (2021).

[23] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding Distributed Poisoning Attack in Federated Learning. In *IEEE Int. Conf. on Parallel and Distributed Sys.* 233–239.

[24] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv:1902.06705* (2019).

[25] Javier Carnerero-Cano, Luis Muñoz-González, Phillippa Spencer, and Emil C Lupu. 2021. Regularization Can Help Mitigate Poisoning Attacks... with the Right Hyperparameters. *arXiv:2105.10948* (2021).

[26] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv:1810.00069* (2018).

[27] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. 2021. Property Inference From Poisoning. *arXiv:2101.11073* (2021).

[28] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *arXiv:1811.03728* (2018).

[29] Chuanshuai Chen and Jiazhu Dai. 2020. Mitigating backdoor attacks in LSTM-based Text Classification Sys. by Backdoor Keyword Identification. *arXiv:2007.12070* (2020).

[30] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In *Int. Joint Conf. on AI, IJCAI 2019*. 4658–4664.

[31] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks Without Training Substitute Models. In *10th ACM Workshop on AI and Security (AISec '17)*. 15–26.

[32] Ruoxin Chen, Zenan Li, Jie Li, Junchi Yan, and Chentao Wu. 2022. On Collective Robustness of Bagging Against Data Poisoning. In *ICML*.

[33] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv:1712.05526* (2017).

[34] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML Workshop on Adversarial ML (2021)*.

[35] Hao Cheng, Kaidi Xu, Sijia Liu, Pin-Yu Chen, Pu Zhao, and Xue Lin. 2020. Defending against backdoor attack on deep neural networks. *arXiv:2002.12162* (2020).

[36] Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. 2021. Deep Feature Space Trojan Attack of Neural Networks by Controlled Detoxification. In *Thirty-Fifth AAAI Conf. on AI 2021*. AAAI Press, 1148–1156.

[37] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. 2020. Sentinet: Detecting localized universal attacks against deep learning systems. In *IEEE Security and Privacy Workshops, SPW 2020*. IEEE, 48–54.

[38] Antonio Emanuele Cinà, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2022. Energy-Latency Attacks via Sponge Poisoning. *arXiv:2203.08147* (2022).

[39] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2022. Machine Learning Security against Data Poisoning: Are We There Yet? *CoRR* abs/2204.05986 (2022).

[40] Antonio Emanuele Cinà, Kathrin Grosse, Sebastiano Vascon, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2021. Backdoor Learning Curves: Explaining Backdoor Poisoning Beyond Influence Functions. *arXiv:2106.07214* (2021).

[41] Antonio Emanuele Cinà, Alessandro Torcinovich, and Marcello Pelillo. 2022. A black-box adversarial attack for poisoning clustering. *Pattern Recognition* 122 (2022), 108306.

[42] Antonio Emanuele Cinà, Sebastiano Vascon, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2021. The Hammer and the Nut: Is Bilevel Optimization Really Needed to Poison Linear Classifiers?. In *Int. Joint Conf. on Neural Networks, IJCNN 2021*. IEEE, 1–8.

[43] G.F. Cretu, A. Stavrou, M.E. Locasto, S.J. Stolfo, and A.D. Keromytis. 2008. Casting out Demons: Sanitizing Training Data for Anomaly Sensors. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. 81 –95.

[44] Ambra Demontis, Battista Biggio, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. 2017. Infinity-Norm Support Vector Machines Against Adversarial Label Contamination. In *ITASEC (CEUR Workshop Proceedings, Vol. 1816)*. CEUR-WS.org, 106–115.

[45] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2019. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *USENIX Sec. Symp.* USENIX Association, 321–338.

[46] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. 2019. Sever: A robust meta-algorithm for stochastic optimization. In *Int. Conf. on ML*. PMLR, 1596–1606.

[47] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. 2020. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Computer Security Applications Conf.* 897–912.

[48] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. 2021. LIRA: Learnable, Imperceptible and Robust Backdoor Attacks. In *IEEE ICCV*. 11966–11976.

[49] Justin Domke. 2012. Generic Methods for Optimization-Based Modeling. In *15th Int. Conf. on Art. Intell. and Statistics, AISTATS 2012*. JMLR, 318–326.

[50] Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. 2021. Black-box Detection of Backdoor Attacks with Limited Information and Data. In *ICCV*.

[51] Min Du, Ruoxi Jia, and Dawn Song. 2020. Robust anomaly detection and backdoor attack detection via differential privacy. In *ICLR, 2020*.

[52] Tom Everitt, Victoria Krakovna, Laurent Orseau, and Shane Legg. 2017. Reinforcement learning with a corrupted reward channel. In *26th Int. Joint Conf. on AI, IJCAI 2017*. 4705–4713.

[53] Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. 2019. Learning to Confuse: Generating Training Time Adversarial Data with Auto-Encoder. In *NeurIPS*.

[54] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. 2014. Robust Logistic Regression and Classification. In *Advances in Neural Inf. Proc. Sys., NIPS*. 253–261.

[55] Liam Fowl, Ping-yeh Chiang, Micah Goldblum, Jonas Geiping, Arpit Bansal, Wojtek Czaja, and Tom Goldstein. 2021. Preventing unauthorized use of proprietary data: Poisoning for secure dataset release. *arXiv:2103.02683* (2021).

[56] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojtek Czaja, and Tom Goldstein. 2021. Adversarial Examples Make Strong Poisons. *arXiv:2106.10807*.

[57] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *ICML*, Vol. 80. PMLR, 1563–1572.

[58] Christopher Frederickson, Michael Moore, Glenn Dawson, and Robi Polikar. 2018. Attack strength vs. detectability dilemma in adversarial ML. In *Int. Joint Conf. on Neural Networks, IJCNN 2018*. IEEE, 1–8.

[59] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. 2020. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv:2007.10760* (2020).

[60] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Computer Security Applications Conf.* 113–125.

[61] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. 2021. What Doesn't Kill You Makes You Robust (er): Adversarial Training against Poisons and Backdoors. *arXiv:2102.13624* (2021).

[62] Jonas Geiping, Liam H. Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. 2021. Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching. In *ICLR 2021*. OpenReview.

[63] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. 2018. Motivating the rules of the game for adversarial example research. *arXiv:1807.06732* (2018).

[64] 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 Transactions on PAMI* (2022), 1–1.

[65] Shafi Goldwasser, Michael P Kim, Vinod Vaikuntanathan, and Or Zamir. 2022. Planting undetectable backdoors in machine learning models. *arXiv:2204.06974* (2022).

[66] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR 2015*.

[67] Dou Goodman, Hao Xin, Wang Yang, Wu Yuesheng, Xiong Junfeng, and Zhang Huan. 2020. Advbox: a toolbox to generate adversarial examples that fool neural networks. *arXiv:2001.05574* (2020).

[68] Kathrin Grosse, Lukas Bieringer, Tarek Richard Besold, Battista Biggio, and Katharina Krombholz. 2023. Machine Learning Security in Industry: A Quantitative Survey. *IEEE Transactions on Information Forensics and Security* (2023).

[69] Kathrin Grosse, Taesung Lee, Battista Biggio, Youngja Park, Michael Backes, and Ian Molloy. 2022. Backdoor Smoothing: Demystifying Backdoor Attacks on Deep Neural Networks. *Computers & Security* (2022), 102814.

[70] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the ML model supply chain. *arXiv* (2017).

[71] Junfeng Guo, Ang Li, and Cong Liu. 2022. AEVA: Black-box Backdoor Detection Using Adversarial Extreme Value Analysis. In *ICLR,2022*.

[72] Junfeng Guo and Cong Liu. 2020. Practical Poisoning Attacks on Neural Networks. In *16th Eur. Conf. on Com. Vis., ECCV 2020*. Springer, 142–158.

[73] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in AI Systems. *arXiv:1908.01763* (2019).

[74] Yi Han, David Hubczenko, Paul Montague, Olivier De Vel, Tamas Abraham, Benjamin IP Rubinstein, Christopher Leckie, Tansu Alpcan, and Sarah Erfani. 2020. Adversarial Reinforcement Learning under Partial Observability in Autonomous Computer Network Defence. In *Int. Joint Conf. on Neural Networks, IJCNN 2020*. IEEE, 1–8.

[75] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. 2021. SPECTRE: Defending Against Backdoor Attacks Using Robust Statistics. In *Int. Conf. on ML, ICML 2020*. PMLR, 4129–4139.

[76] Jamie Hayes and Olga Ohrimenko. 2018. Contamination Attacks and Mitigation in Multi-Party ML. *Advances in Neural Inf. Proc. Sys., NIPS* 31 (2018), 6604–6615.

[77] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. 2020. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv:2002.11497* (2020).

[78] Xiaoling Hu, Xiao Lin, Michael Cogswell, Yi Yao, Susmit Jha, and Chao Chen. 2022. Trigger Hunting with a Topological Prior for Trojan Detection. In *ICLR, 2022*.

[79] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. 2022. Backdoor Defense via Decoupling the Training Process. In *ICLR, 2022*.

[80] W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. 2020. MetaPoison: Practical General-purpose Clean-label Data Poisoning. In *Advances in Neural Inf. Proc. Sys., NeurIPS*.

[81] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. 2019. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv:1911.07399* (2019).

[82] Yunhan Huang and Quanyan Zhu. 2019. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *Int. Conf. on Decision and Game Theory for Security*. Springer, 217–237.

[83] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating ML: Poisoning attacks and countermeasures for regression learning. In *IEEE Symposium on Security and Privacy, S&P 2018*. IEEE, 19–35.

[84] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. 2021. Subpopulation data poisoning attacks. In *ACM SIGSAC Conf. on Computer and Communications Security, CCS 2021*. 3104–3122.

[85] Mojan Javaheripi, Mohammad Samragh, Gregory Fields, Tara Javidi, and Farinaz Koushanfar. 2020. CLEANN: Accelerated trojan shield for embedded neural networks. In *IEEE/ACM Int. Conf. On Computer Aided Design, ICCAD 2020*. IEEE, 1–9.

[86] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. 2020. Certified robustness of nearest neighbors against data poisoning attacks. *arXiv* (2020).

[87] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

[88] Sara Kaviani and Insoo Sohn. 2021. Defense against neural trojan attacks: A survey. *Neurocomputing* 423 (2021), 651–667.

[89] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. 2020. TrojDRL: evaluation of backdoor attacks on deep reinforcement learning. In *ACM/IEEE Design Automation Conf., DAC 2020*. IEEE, 1–6.

[90] Marius Kloft and Pavel Laskov. 2010. Online Anomaly Detection under Adversarial Impact. In *AISTATS 2010*. JMLR, 405–412.

[91] Stefanos Koffas, Jing Xu, Mauro Conti, and Stjepan Picek. 2021. Can You Hear It? Backdoor Attacks via Ultrasonic Triggers. *arXiv* (2021).

[92] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *Int. Conf. on ML, ICML*. PMLR, 1885–1894.

[93] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. 2022. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning* 111 (2022), 1–47.

[94] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. 2020. Universal litmus patterns: Revealing backdoor attacks in cnns. In *IEEE/CVF Int. Conf. on Computer Vision, ICCV 2021*. 301–310.

[95] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial ML-industry perspectives. (2020), 69–75.

[96] Ricky Laishram and Vir Virander Phoha. 2016. Curie: A method for protecting SVM Classifier from Poisoning Attack. *arXiv:1606.01584* (2016).

[97] Alexander Levine and Soheil Feizi. 2021. Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks. In *ICLR 2021*.

[98] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of ML Research* 18, 1 (2017), 6765–6816.

[99] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang. 2020. Invisible Backdoor Attacks on Deep Neural Networks via Steganography and Regularization. *IEEE Trans. on Dependable and Secure Computing* PP (09 2020), 1–1.

[100] Yige Li, Nodens Koren, Lingjuan Lyu, Xixiang Lyu, Bo Li, and Xingjun Ma. 2021. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. *ICLR, 2021*.

[101] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible Backdoor Attack With Sample-Specific Triggers. In *IEEE/CVF Int. Conf. on Computer Vision, ICCV 2021*. 16463–16472.

[102] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Inf. Proc. Sys., NeurIPS* 34.

[103] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor learning: A survey. *arXiv:2007.08745* (2020).

[104] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. 2020. Rethinking the trigger of backdoor attack. *arXiv* (2020).

[105] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. 2020. Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features. In *SIGSAC Conf. on Computer and Communications Security, CCS 2020*. 113–131.

[106] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. 2017. Robust Linear Regression Against Training Data Poisoning. In *10th ACM Workshop on Art. Intell. and Sec., AISec 2017*. ACM, 91–102.

[107] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Int. Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 273–294.

[108] Xuanqing Liu, Si Si, Jerry Zhu, Yang Li, and Cho-Jui Hsieh. 2019. A Unified Framework for Data Poisoning Attack to Graph-based Semi-supervised Learning. In *Advances in Neural Inf. Proc. Sys., NeurIPS 2019*. 9777–9787.

[109] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation. In *ACM SIGSAC Conf. on Computer and Communications Security, CCS 2019*. ACM, 1265–1282.

[110] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Network and Distributed System Sec. Symp., NDSS*. 45–48.

[111] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. In *Computer Vision - ECCV 2020 - 16th Eur. Conf.* Springer, 182–199.

[112] Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural Trojans. In *IEEE Int. Conf. on Computer Design, ICCD 2017*. 45–48.

[113] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. 2017. Particle swarm optimization for hyper-parameter selection in deep neural networks. In *the genetic and evolutionary computation conference*. 481–488.

[114] Jonathan Lorraine, Paul Vicol, and David Duvenaud. 2020. Optimizing millions of hyperparameters by implicit differentiation. In *Int. Conf. on AI and Statistics*. PMLR, 1540–1552.

[115] Yuzhe Ma, Kwang-Sung Jun, Lihong Li, and Xiaojin Zhu. 2018. Data poisoning attacks in contextual bandits. In *Int. Conf. on Decision and Game Theory for Security*. Springer, 186–204.

[116] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. 2019. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *IJCAI*. 4732–4738.

[117] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. 2015. Gradient-based Hyperparameter Optimization through Reversible Learning. In *32nd Int. Conf. on ML, ICML 2015*. JMLR, 2113–2122.

[118] Naren Manoj and Avrim Blum. 2021. Excess capacity and backdoor poisoning. *Advances in Neural Inf. Proc. Sys., NeurIPS* 34 (2021), 20373–20384.

[119] Sean McGregor. 2020. Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database. *arXiv:2011.08512* (2020).

[120] Akshay Mehra, Bhavya Kailkhura, Pin-Yu Chen, and Jihun Hamm. 2021. How Robust are Randomized Smoothing based Defenses to Data Poisoning?. In *IEEE/CVF Int. Conf. on Computer Vision, ICCV 2021*. 13244–13253.

[121] Shike Mei and Xiaojin Zhu. 2015. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *AAAI*. 2871–2877.

[122] Marco Melis, Ambra Demontis, Maura Pintor, Angelo Sotgiu, and Battista Biggio. 2019. secml: A Python Library for Secure and Explainable ML. *arXiv:1912.10013* (2019).

[123] Nicolas M. Müller, Daniel Kowatsch, and Konstantin Böttinger. 2020. Data Poisoning Attacks on Regression Learning and Corresponding Defenses. In *25th IEEE Pacific Rim Int. Symposium on Dependable Computing, PRDC 2020*. IEEE, 80–89.

[124] 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 *10th ACM Workshop on Art. Intell. and Sec., AISec 2017*. ACM, 27–38.

[125] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, JD Tygar, and Kai Xia. 2009. Misleading learners: Co-opting your spam filter. In *ML in Cyber Trust*. Springer, 17–51.

[126] 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 ML to Subvert Your Spam Filter. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET 2008*. USENIX, 1–9.

[127] James Newsome, Brad Karp, and Dawn Xiaodong Song. 2006. Paragraph: Thwarting Signature Learning by Training Maliciously. In *RAID 2006 (Lecture Notes in Computer Science, Vol. 4219)*. Springer, 81–105.

[128] Tuan Anh Nguyen and Anh Tran. 2020. Input-Aware Dynamic Backdoor Attack. In *Advances in Neural Inf. Proc. Sys., NeurIPS*.

[129] Tuan Anh Nguyen and Anh Tuan Tran. 2021. WaNet - Imperceptible Warping-based Backdoor Attack. In *ICLR, 2021*. OpenReview.net.

[130] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. 2018. Adversarial Robustness Toolbox v1.2.0. *CoRR* 1807.01069 (2018).

[131] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in ML: from phenomena to black-box attacks using adversarial samples. *arXiv:1605.07277* (2016).

[132] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against ML. In *ACM Asia Conf. on Computer and Communications Security, AsiaCCS 2017*. ACM, 506–519.

[133] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. 2018. Label sanitization against label flipping poisoning attacks. In *Joint Eur. Conf. on ML and Knowledge Discovery in Databases*. Springer, 5–15.

[134] R. Perdisci, D. Dagon, Wenke Lee, P. Fogla, and M. Sharif. 2006. Misleading worm signature generators using deliberate noise injection. In *IEEE Symposium on Security and Privacy, S& P 2006*. 15 pp.–31.

[135] 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 *Eur. Conf. on Computer Vision*. Springer, 55–70.

[136] Maura Pintor, Luca Demetrio, Angelo Sotgiu, Giovanni Manca, Ambra Demontis, Nicholas Carlini, Battista Biggio, and Fabio Roli. 2021. Indicators of Attack Failure: Debugging and Improving Optimization of Adversarial Examples. *arXiv preprint arXiv:2106.09947* (2021).

[137] Fanchao Qi, Yangyi Chen, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2020. ONION: A Simple and Effective Defense Against Textual Backdoor Attacks. *arXiv:2011.10369* (2020).

[138] Ximing Qiao, Yukun Yang, and Hai Li. 2019. Defending neural backdoors via generative distribution modeling. In *NeurIPS*. 14004–14013.

[139] 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 *Int. Conf. on ML, ICML 2020*. PMLR, 7974–7984.

[140] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. 2020. Certified robustness to label-flipping attacks via randomized smoothing. In *ICML*. PMLR, 8230–8241.

[141] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *9th ACM SIGCOMM Conf. on Internet Measurement*. 1–14.

[142] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden Trigger Backdoor Attacks. In *AAAI*. AAAI Press, 11957–11965.

[143] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. Dynamic backdoor attacks against ML models. *arXiv* (2020).

[144] Esha Sarkar, Yousif Alkindi, and Michail Maniatakos. 2020. Backdoor suppression in neural networks using input fuzzing and majority voting. *IEEE Design & Test* 37, 2 (2020), 103–110.

[145] Esha Sarkar, Hadjer Benkraouda, and Michail Maniatakos. 2020. FaceHack: Triggering backdoored facial recognition Sys. using facial characteristics. arXive:2006.11623 (2020).

[146] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. 2021. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *Int. Conf. on ML, ICML 2021*. PMLR, 9389–9398.

[147] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers. In *USENIX Sec. Symp.*

[148] 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 *Advances in Neural Inf. Proc. Sys., NeurIPS 2018*. 6106–6116.

[149] Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y Zhao. 2022. Traceback of Targeted Data Poisoning Attacks in Neural Networks. In *USENIX Sec. Symp.* USENIX Association.

[150] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Federated Learning. In *IEEE Symposium on Security and Privacy*.

[151] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. 2021. Backdoor Scanning for Deep Neural Networks through K-Arm Optimization. *arXiv:2102.05123* (2021).

[152] Reza Shokri et al. 2020. Bypassing Backdoor Detection Algorithms in Deep Learning. In *IEEE Euro S&P 2020*. IEEE, 175–183.

[153] David Solans, Battista Biggio, and Carlos Castillo. 2020. Poisoning Attacks on Algorithmic Fairness. In *ECML PKDD*. Springer, 162–177.

[154] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using ML for network intrusion detection. In *IEEE S&P 2010*. IEEE, 305–316.

[155] Ezekiel Soremekun, Sakshi Udeshi, Sudipta Chattopadhyay, and Andreas Zeller. 2020. Exposing backdoors in robust ML models. *arXiv* (2020).

[156] Hossein Souri, Micah Goldblum, Liam Fowl, Rama Chellappa, and Tom Goldstein. 2021. Sleeper Agent: Scalable Hidden Trigger Backdoors for Neural Networks Trained from Scratch. *arXiv:2106.08970* (2021).

[157] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified Defenses for Data Poisoning Attacks. In *Neural Inf. Proc. Sys., NIPS*. 3517–3529.

[158] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. When does ML {FAIL}? generalized transferability for evasion and poisoning attacks. In *USENIX Sec. Symp.* 1299–1316.

[159] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. 2018. Adversarial attack and defense on graph data: A survey. *arXiv:1812.10528* (2018).

[160] Mingjie Sun, Zichao Li, Chaowei Xiao, Haonan Qiu, Bhavya Kailkhura, Mingyan Liu, and Bo Li. 2021. Can Shape Structure Features Improve Model Robustness under Diverse Adversarial Settings?. In *IEEE CVF Int. Conf. on Computer Vision*. 7526–7535.

[161] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning? *arXiv* (2019).

[162] Rahim Taheri, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Ali Miri, and Mauro Conti. 2020. On defending against label flipping attacks on malware detection system. *Neural Computing and Applications* (2020), 1–20.

[163] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. 2021. Demon in the Variant: Statistical Analysis of {DNNs} for Robust Backdoor Contamination Detection. (2021), 1541–1558.

[164] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *Comput. Surveys* (2022).

[165] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data Poisoning Attacks Against Federated Learning Sytems. In *Eur. Symposium on Research in Computer Security, ESORICS*. Springer, 480–501.

[166] Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.

[167] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing ML Models via Prediction APIs. In *USENIX Sec. Symp.* 601–618.

[168] Brandon Tran, Jerry Li, and Aleksander Mądry. 2018. Spectral signatures in backdoor attacks. In *Conf. on Neural Inf. Proc. Sys., NIPS 2018*. 8011–8021.

[169] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2019. Label-consistent backdoor attacks. *arXiv:1912.02771* (2019).

[170] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. 2019. Model agnostic defence against backdoor attacks in ML. *arXiv:1908.02203* (2019).

[171] Akshaj Kumar Veldanda, Kang Liu, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. 2021. NNoculation: Catching BadNets in the Wild. In *14th ACM Workshop on AI and Security*. 49–60.

[172] Miguel Villarreal-Vasquez and Bharat Bhargava. 2020. Confoc: Content-focus protection against trojan attacks on neural networks. *arXiv* (2020).

[173] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy, S&P 2019*. IEEE, 707–723.

[174] Jingkang Wang, Yang Liu, and Bo Li. 2020. Reinforcement learning with perturbed rewards. In *AAAI Conf. on Art. Intell.* AAAI Press, 6202–6209.

[175] Wenxiao Wang, Alexander J Levine, and Soheil Feizi. 2022. Improved Certified Defenses against Data Poisoning with (Deterministic) Finite Aggregation. In *ICML*. PMLR, 22769–22783.

[176] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. 2020. Rab: Provable robustness against backdoor attacks. *arXiv:2003.08904* (2020).

[177] Jialin Wen, Benjamin Zi Hao Zhao, Minhui Xue, Alina Oprea, and Haifeng Qian. 2021. With Great Dispersion Comes Greater Resilience: Efficient Poisoning Attacks and Defenses for Linear Regression Models. *IEEE Trans. on Inf. Forensics and Security* (2021).

[178] Cheng-Hsin Weng, Yan-Ting Lee, and Shan-Hung Brandon Wu. 2020. On the Trade-off between Adversarial and Backdoor Robustness. *Neural Inf. Proc. Sys., NeurIPS* (2020).

[179] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y. Zhao. 2021. Backdoor Attacks Against Deep Learning Sys. in the Physical World. *IEEE/CVF Int. Conf. on Computer Vision, ICCV 2021* (2021), 6202–6211.

[180] Dongxian Wu and Yisen Wang. 2021. Adversarial Neuron Pruning Purifies Backdoored Deep Models. In *NeurIPS*.

[181] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph Backdoor. In *USENIX Sec. Symp.* 1523–1540.

[182] Zhen Xiang, David Miller, and George Kesidis. 2022. Post-Training Detection of Backdoor Attacks for Two-Class and Multi-Attack Scenarios. In *ICLR,2022*.

[183] Zhen Xiang, David J Miller, and George Kesidis. 2019. A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense. In *IEEE 29th Int. Workshop on ML for Signal Proc., MLSP 2019*. IEEE, 1–6.

[184] Zhen Xiang, David J Miller, and George Kesidis. 2020. Detection of Backdoors in Trained Classifiers Without Access to the Training Set. *IEEE Trans. on Neural Networks and Learning Sys.* (2020).

[185] Zhen Xiang, David J Miller, and George Kesidis. 2021. L-Red: Efficient Post-Training Detection of Imperceptible Backdoor Attacks Without Access to the Training Set. In *IEEE Int. Conf. on Acoustics, Speech and Signal Proc., ICASSP 2021*. IEEE, 3745–3749.

[186] Zhen Xiang, David J Miller, and George Kesidis. 2021. Reverse engineering imperceptible backdoor attacks on deep neural networks for detection and training set cleansing. *Computers & Security* 106 (2021), 102280.

[187] Chaowei Xiao, Xinlei Pan, Warren He, Jian Peng, Mingjie Sun, Jinfeng Yi, Mingyan Liu, Bo Li, and Dawn Song. 2019. Characterizing attacks on deep reinforcement learning. *arXiv:1907.09470* (2019).

[188] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is Feature Selection Secure against Training Data Poisoning?. In *32nd Int. Conf. on ML, ICML 2015*. JMLR, 1689–1698.

[189] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. 2015. Support vector machines under adversarial label contamination. *Neurocomputing* 160 (2015), 53–62.

[190] Han Xiao, Huang Xiao, and Claudia Eckert. 2012. Adversarial Label Flips Attack on Support Vector Machines. In *ECAI 2012 - 20th Eur. Conf. on AI. Including Prestigious Applications of AI, PAIS-2012*. IOS Press, 870–875.

[191] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2020. DBA: Distributed Backdoor Attacks against Federated Learning. In *ICLR, 2020*. OpenReview.

[192] Hang Xu, Rundong Wang, Lev Raizman, and Zinovi Rabinovich. 2021. Transferable Environment Poisoning: Training-time Attack on Reinforcement Learning. In *20th Int. Conf. on Autonomous Agents and MultiAgent Sys.* 1398–1406.

[193] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. 2021. Detecting AI Trojans Using Meta Neural Analysis. In *IEEE Symposium on Security and Privacy, S&P*. IEEE, 103–120.

[194] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. 2017. Generative Poisoning Attack Method Against Neural Networks. *CoRR* abs/1703.01340 (2017).

[195] Yu Yang, Tian Yu Liu, and Baharan Mirzasoleiman. 2022. Not All Poisons are Created Equal: Robust Training against Data Poisoning. In *ICML*. PMLR, 25154–25165.

[196] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent backdoor attacks on deep neural networks. In *ACM SIGSAC Conf. on Computer and Communications Security, CCS 2019*. 2041–2055.

[197] Kota Yoshida and Takeshi Fujino. 2020. Disabling Backdoor and Identifying Poison Data by using Knowledge Distillation in Backdoor Attacks on Deep Neural Networks. In *13th ACM Workshop on AI and Security*. 117–127.

[198] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Workshop on ML in High-Performance Computing Environments*. 1–5.

[199] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. 2022. Adversarial Unlearning of Backdoors via Implicit Hypergradient. In *ICLR, 2022*.

[200] Yi Zeng, Han Qiu, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. 2020. DeepSweep: An Evaluation Framework for Mitigating DNN Backdoor Attacks using Data Augmentation. *arXiv:2012.07006* (2020).

[201] Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. 2019. Poisoning Attack in Federated Learning using Generative Adversarial Nets. In *IEEE Int. Conf. On Trust, Security and Privacy*. IEEE, 374–380.

[202] Jiale Zhang, Di Wu, Chengyong Liu, and Bing Chen. 2020. Defending Poisoning Attacks in Federated Learning via Adversarial Training Method. In *Int. Conf. on Frontiers in Cyber Security*. Springer, 83–94.

[203] Rui Zhang and Quanyan Zhu. 2017. A game-theoretic analysis of label flipping attacks on distributed support vector machines. In *Conf. on Inf. Sciences and Sys., CISS 2017*. IEEE, 1–6.

[204] Xinqiao Zhang, Huili Chen, and Farinaz Koushanfar. 2021. TAD: Trigger Approximation based Black-box Trojan Detection for AI. *arXiv* (2021).

[205] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. 2020. Adaptive reward-poisoning attacks against reinforcement learning. In *Int. Conf. on ML, ICML 2020*. PMLR, 11225–11234.

[206] Xinyang Zhang, Zheng Zhang, and Ting Wang. 2020. Trojaning Language Models for Fun and Profit. *CoRR* abs/2008.00312 (2020).

[207] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor Attacks to Graph Neural Networks. In *SACMAT '21: The 26th ACM Symposium on Access Control Models and Technologies*. ACM, 15–26.

[208] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. 2020. Bridging mode connectivity in loss landscapes and adversarial robustness. In *ICLR, 2021.*

[209] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-Label Backdoor Attacks on Video Recognition Models. In *IEEE/CVF Int. Conf. on Computer Vision, ICCV 2020*. IEEE, 14431–14440.

[210] Ying Zhao, Junjun Chen, Jiale Zhang, Di Wu, Jian Teng, and Shui Yu. 2019. PDGAN: a novel poisoning defense method in federated learning using generative adversarial network. In *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 595–609.

[211] Haoti Zhong, Cong Liao, Anna Cinzia Squicciarini, Sencun Zhu, and David J. Miller. 2020. Backdoor Embedding in Convolutional Neural Network Models via Invisible Perturbation. In *CODASPY '20: Tenth ACM Conf. on Data and Application Security and Privacy 2020*. ACM, 97–108.

[212] 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 *36th Int. Conf. on ML, ICML 2019*. PMLR, 7614–7623.

[213] Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. 2020. Gangsweep: Sweep out neural backdoors by gan. In *28th ACM Int. Conf. on Multimedia*. 3173–3181.

[214] Liuwan Zhu, Rui Ning, Chunsheng Xin, Chonggang Wang, and Hongyi Wu. 2021. CLEAR: Clean-Up Sample-Targeted Backdoor in Neural Networks. In *IEEE/CVF Int. Conf. on Computer Vision, ICCV 2021*. 16453–16462.

[215] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *ICLR, 2019*. OpenReview.net.