Journal of Cybersecurity and Privacy

MDPI

Review

Cybersecurity for AI Systems: A Survey

Raghvinder S. Sangwan, Youakim Badr * and Satish M. Srinivasan

School of Graduate Professional Studies, The Pennsylvania State University, 30 E. Swedesford Road, Malvern, PA 19355, USA
* Correspondence: yzb61@psu.edu

Abstract: Recent advances in machine learning have created an opportunity to embed artificial intelligence in software-intensive systems. These artificial intelligence systems, however, come with a new set of vulnerabilities making them potential targets for cyberattacks. This research examines the landscape of these cyber attacks and organizes them into a taxonomy. It further explores potential defense mechanisms to counter such attacks and the use of these mechanisms early during the development life cycle to enhance the safety and security of artificial intelligence systems.

Keywords: machine learning; cybersecurity; AI attacks; defense mechanism

check for updates

Citation: Sangwan, R.S.; Badr, Y.; Srinivasan, S.M. Cybersecurity for AI Systems: A Survey. *J. Cybersecur. Priv.* **2023**, *3*, 166–190. https://doi.org/10.3390/jcp3020010

Academic Editor: Giorgio Giacinto

Received: 24 January 2023
Revised: 8 March 2023
Accepted: 11 March 2023
Published: 4 May 2023

1. Introduction

Advances in Artificial Intelligence (AI) technology have contributed to the enhancement of cybersecurity capabilities of traditional systems with applications that include detection of intrusion, malware, code vulnerabilities and anomalies. However, these systems with embedded machine learning models have opened themselves to a new set of vulnerabilities, commonly known as AI attacks. Currently, these systems are prime targets for cyberattacks, thus compromising the security and safety of larger systems that encompass them. Modern day AI attacks are not only limited to just coding bugs and errors. They manifest due to the inherent limitations or vulnerabilities of systems [1]. By exploiting the vulnerabilities in the AI system, attackers aim at either manipulating its behavior or obtaining its internal details by tampering with its input, training data, or the machine learning (ML) model. McGraw et al. [2] have classified AI attacks broadly as manipulation and extraction attacks. Based on the inputs given to the system, the training dataset used for learning, and manipulation of the model hyperparameters, attacks on AI systems can manifest in different types, with different degrees of severity. For example, adversarial or evasion attack can be launched by manipulating the input to the AI system, which results in the system producing an unintended outcome. A poisoning or causative attack can be launched by tainting the training dataset, which would result in the AI system exhibiting unethical behavior.

Therefore, it is important that we start thinking about designing security into AI systems, rather than retrofitting it as an afterthought. This research addresses the following research questions:

RQ1: What are the cyberattacks that AI systems can be subjected to?

RQ2: Can the attacks on AI systems be organized into a taxonomy, to better understand how the vulnerabilities manifest themselves during the system development.

RQ3: What are possible defense mechanisms to prevent AI systems being subjected to cyberattacks?

RQ4: Is it possible to devise a generic defense mechanism against all kinds of AI attacks.

To address these research questions and determine the extent of risk to safety and security of AI systems, we first conducted a systematic literature review looking for AI attacks on systems reported in the literature. We then organized these attacks into a

taxonomy to not only understand the types of vulnerabilities, but also the stage in the development of AI systems when these vulnerabilities manifest themselves. We then conducted further literature search looking for any defense mechanisms to counter these attacks and improve the safety and security of AI systems.

This study is organized as follows. In Section 2, we report the results of the systematic literature review and identify the attacks, from an AI system development perspective, and their vulnerabilities. In Section 3, we introduce a taxonomy of AI attacks along with defense mechanisms and countermeasures to mitigate their threats. Section 4 concludes the study and highlights major findings.

2. Literature Review

This survey was founded on searching, by keywords, to find related articles to cybersecurity of AI systems. The top most used keywords are as follow: cybersecurity, cyberattack, and vulnerabilities. We searched Scopus, an Elsevier abstracts and citation database, for articles having titles that matched the search query ("cyber security" OR "cybersecurity" OR "security" OR "cyberattack" OR "vulnerability" OR "vulnerabilities" OR "threat" OR "attack" OR "AI attack") AND ("AI" OR "ML" OR "Artificial Intelligence" OR "Machine Learning") AND ("system")).

The search resulted in a total of 1366 articles. Within these articles, we looked for those in computer science or computer engineering subject areas that were published in journals in the English language, leaving us with 415 manuscripts. We carefully reviewed the abstracts of the papers to determine their relevance. Only articles that discussed the vulnerabilities of AI systems to attacks and/or their defense mechanisms were considered.

During the learning or training stage, an AI system needs data for training a machine learning model. The training data are subject to manipulation attacks, requiring that their integrity be verified. Ma et al. [3] used a visual analytics framework for explaining and exploring ML model vulnerabilities to data poisoning attacks. Kim and Park [4] proposed a blockchain-based environment that collects and stores learning data whose confidentiality and integrity can be guaranteed. Mozaffari-Kermani et al. [5] focused on data poisoning attacks on, and the defenses for, machine learning algorithms in healthcare.

During the inference or testing stage, an AI system can be subjected to manipulation attacks by presenting falsified data to be classified as legitimate data. Adversarial or evasion attacks and/or potential defenses against such attacks are discussed in [6–14]. Chen et al. [15] looked at such attacks in the context of reinforcement learning. Li et al. [16] proposed a low latency decentralized framework for identifying adversarial attacks in deep learning-based industrial AI systems. Garcia-Ceja et al. [17] described how biometric profiles can be generated to impersonate a user by repeatedly querying a classifier and how the learned profiles can be used to attack other classifiers trained on the same dataset. Biggio et al. [18] examined vulnerabilities of biometric recognition systems and their defense mechanisms. Ren et al. [19] also looked at querying-based attacks against black-box machine learning models and potential defense mechanisms against such attacks. Wang et al. [20] looked at a variant, termed the Man-in-the-Middle attack, using generative models for querying. Threats from, and potential defense against, attacks on machine learning models in 5G networks is discussed in [21,22]. Apruzzese et al. [23] provided an approach to mitigating evasion attacks on AI-based network intrusion detection systems. Zhang et al. [24] explored adversarial attacks against commonly used ML-based cybersecurity systems. Liu et al. [25] discussed how to improve robustness of ML-based CAD systems against adversarial attacks. Building malware detection systems that are more resilient to adversarial attacks was the focus of [26,27], and Gardiner and Nagaraja [28] provided a comprehensive survey on vulnerabilities of ML models in malware detection systems. Dasgupta and Collins [29] surveyed game theoretical approaches that can be used to make ML algorithms robust against adversarial attacks.

During the inference or testing stage, extraction attacks are possible using the feature vector of a model for model inversion or reconstruction and gaining access to private data that was used as input or for training an AI system [30].

Hansman and Hunt [31] and Gao et al. [32] proposed a taxonomy of network and computer attacks to categorize different attack types. Their taxonomy includes four dimensions to categorize attacks on AI systems, including attack classes, attack targets, vulnerabilities and exploits used by the attacks and whether the attack has a payload or effect beyond itself. Their taxonomical structure is very comprehensive and can be used to analyze a system for its dependability, reliability and security.

Despite the benefits of machine learning technologies, the learning algorithms can be abused by cybercriminals to conduct illicit and undesirable activities. It was shown in [33,34] that attackers might gain a significant benefit by exploiting vulnerabilities in the learning algorithms, which can sometimes become a weakest link in the security chain. Several studies related to attacks on machine learning algorithms have been reported in the literature using different threat models. Barreno et al. [35,36], Huang et al. [36], Biggio et al. [37] and Munoz-Gonzalez et al. [38] discussed different attack scenarios against machine learning models with different attack models. The frameworks they proposed characterize the attacks according to the attacker's goal, their capabilities to manipulate the data and influence the learning system, familiarity with the algorithms, the data used by the defender and the attacker's strategy. For example, data poisoning attacks, also known as causative attacks, are a major emerging security threat to data-driven technologies. In these types of attacks, it can be assumed that the hacker has control over the training dataset that is being used by the learning algorithm. The hacker can actively influence the training dataset in order to subvert the entire learning process, thus decreasing the overall performance of the system, or to produce particular types of errors in the system output. For example, in a classification task, the hacker may poison the data to modify the decision boundaries learned by the learning algorithm, thus resulting in misclassification of instances, or a higher error rate for a specific type of class. This is a kind of threat that is related to the reliability of the large amount of data collected by the systems [38,39].

This survey is distinct from [31,39] in studying attacks on an AI system from the perspective of a software engineering team, that organizes its work around different stages of an AI system's development life cycle. For these different stages of an AI system, and their corresponding attacks, potential defense mechanisms are also provided. Organizing the literature using this perspective can be valuable to systematically study the design of AI systems for security purposes, to explore the trade offs that result from using different defense mechanisms, and to develop a catalog of patterns and tactics for designing AI systems for security purposes.

Table 1 lists various attacks carried out at different stages of the AI system development processes and the countermeasures that are taken against these attacks.

Table 1. Attacks on AI systems at different stages of its development.

Attacks	AI System Development	Vulnerabilities	Defense Mechanisms
Poisoning attacks [1,37]	During training of the model	Weakness in the federated learning algorithms, resulting in stealing of the data and algorithm from individual user devices.	See list of defense mechanisms for both the data and model poisoning attacks.
Data poisoning attacks [38–40]	During the training stage	Tampering of the features and class information in the training dataset	Adversarial training, Feature squeezing, Transferability blocking, MagNet, Defense-GAN, Local intrinsic dimensionality, Reject On Negative Impact (RONI), L-2 Defense, Slab Defense, Loss Defense and K-NN Defense.

Table 1. *Cont.*

Attacks	AI System Development	Vulnerabilities	Defense Mechanisms
Model poisoning attacks [41–44]	During the training stage	Trust ability of the trainer, based on a privately held validation dataset. Use of pre-trained models that are corrupted.	Securely hosting and disseminating pre-trained models in virtual repositories that guarantee integrity to preclude benevolent models from being manipulated. Identifying backdoors in malevolently trained models acquired from untrustworthy trainers by fine-tuning untrusted models.
Transfer learning attacks [42,44–46]	During the training stage	Similarity of the model structures.	Obtain pre-trained models from trusted source. Employ activation-based pruning with different training examples.
Model poisoning in federated learning [41,45,47,48]	During the training stage	Obstruct the convergence of the execution of the distributed Stochastic Gradient Descent (SGD) algorithm,	Robust aggregation methods, robust learning rate.
Model inversion attack [49–52]	During Inference and/or testing stage	Models are typically trained on rather small, or imbalanced, training sets.	L2 Regularizer [49], Dropout and Model Staking [50], MemGuard [51] and Differential privacy [52].
Model extraction attack [53,54]	During Inference and/or training stage	Models having similar characteristics (parameters, shape and size, similar features etc.)	Hiding or adding noises to the output probabilities while keeping the class label of the instances intact. Suppressing suspicious queries or input data.
Inference attack [55]	During Inferencing, Training, and Testing	Model Leaking information leading to inferences being made on private data.	Methods proposed in [55] have leveraged heuristic correlations between the records of the public data and attribute values to defending against inference attacks. Modifying the identified k entries that have large correlations with the attribute values to any given target users.

The following section systematically explores attacks on AI systems and their defenses in more detail.

3. AI Attacks and Defense Mechanisms

Research has been carried out to identify new threats and attacks on different levels of design and implementation of AI systems. Kaloudi and Li [56], stressed the dearth of proper understanding of the malicious intention of the attacks on AI-based systems. The authors introduced 11 use cases divided into five categories: (1) next generation malware, (2) void synthesis, (3) password-based attacks, (4) social bots, and (5) adversarial training. They developed a threat framework to categorize the attacks. Turchin [57] pointed out the lack of desired behaviors of AI systems that could be exploited to design attacks in different phases of system development. The research lists the following modes of failure of AI systems:

- The need for better resources for self-upgradation of AI systems can be exploited by adversaries
- Implementation of malicious goals make the AI systems unfriendly
- Flaws in the user-friendly features
- Use of different techniques to make different stages of AI free from the boundaries of actions expose the AI systems to adversaries

Similar research is carried out by Turchin and Denkenberger [58] where the classification of attacks was based on intelligence levels of AI systems. The authors introduced three levels of AI intelligence with respect to human intelligence: (1) "Narrow AI" which requires human assistance, (2) "Young AI" which has capability a bit better than human, and (3) "Mature AI" whose intelligence is super-human. While classifying the intelligence levels of AI systems, the authors investigated several vulnerabilities during the evolution of capabilities of AI systems. Yampolsky [59] projected a holistic view of tracks as to why an AI system could be malicious, classifying the tracks into two stages: (1) Pre-deployment and

(2) Post-deployment. This includes the intrinsic and extrinsic reasons for AI technologies to be malicious, such as design flaws, intentional activities, or environmental factors.

3.1. Types of Failures

Shiva Kemar et al. [60] discussed two modes of failures of machine learning (ML) systems. They claimed that AI systems can fail either due to the inherent design of the systems (unintentional failures) or by the hand of an adversary (intentional failures).

Unintentional Failures: The unintentional failure mode leads to the failure of an AI/ML system when the AI/ML system generates formally correct, but completely unsafe, behavior.

Intentional failures: Intentional failures are caused by the attackers attempting to destabilize the system either by (a) misclassifying the results, by introducing private training data, or b) by stealing the foundational algorithmic framework. Depending on the accessibility of information about the system components (i.e., knowledge), intentional failures can be further subdivided into different subcategories.

3.1.1. Categories of Unintentional Failures

Unintentional failures happen when AI/ML systems produce an unwanted or unforeseen outcome from a determined action. It happens mainly due to system failures. In this research we further categorize different types of unintentional failures.

- **Reward Hacking:** Reward hacking is a failure mode that an AI/ML system experiences when the underlying framework is a reinforcement learning algorithm. Reward hacking appears when an agent has more return as reward in an unexpected manner in a game environment [61]. This unexpected behavior unsettles the safety of the system. Yuan et al. [62] proposed a new multi-step reinforcement learning framework, where the reward function generates a discounted future reward and, thus, reduces the influence of immediate reward on the current state action pair. The proposed algorithm creates the defense mechanism to mitigate the effect of reward hacking in AI/ML systems.
- **Distributed Shift:** This type of mode appears when an AI/ML model that once performed well in an environment generates dismal performance when deployed to perform in a different environment. One such example is when the training and test data come from two different probability distributions [63]. The distribution shift is further subdivided into three types [64]:

1. Covariate Shift: The shifting problem arises due to the change in input features (covariates) over time, while the distribution of the conditional labeling function remains the same.
2. Label Shift: This mode of failure is complementary to covariate shift, such that the distribution of class conditional probability does not change but the label marginal probability distribution changes.
3. Concept Shift: Concept shift is a failure related to the label shift problem where the definitions of the label (i.e., the posteriori probability) experience spatial or temporal changes.

Subbaswamy and Saria proposed an operator-based hierarchy of solutions that are stable to the distributed shift [65]. There are three operators (i.e., conditioning, intervening and computing counterfactuals) that work on a graph specific to healthcare AI. These operators effectively remove the unstable component of the graph and retain the stable behavior as much as possible. There are also other algorithms to maintain robustness against the distributed shift. Rojas-Carulla et al. [66] proposed a data-driven approach, where the learning of models occurs using data from diverse environments, while Rothenhausler et al. [67] devised bounded magnitude-based robustness, where the shift is assumed to have a known magnitude.

- **Natural Adversarial Examples:** The natural adversarial examples are real-world examples that are not intentionally modified. Rather, they occur naturally, and result in considerable loss of performance of the machine learning algorithms [68]. The instances are semantically similar to the input, legible and facilitate interpretation (e.g., image data) of the outcome [69]. Deep neural networks are susceptible to natural adversarial examples.

3.1.2. Categories of Intentional Failures

The goal of the adversary is deduced from the type of failure of the model. Chakraborty et al. [70] identify four different classes of adversarial goals, based on the machine learning classifier output, which are the following: (1) confidence reduction, where the target model prediction confidence is reduced to a lower probability of classification, (2) misclassification, where the output class is altered from the original class, (3) output misclassification, which deals with input generation to fix the classifier output into a particular class, and (4) input/output misclassification, where the label of a particular input is forced to have a specific class.

Shiv Kumar et al. [60] identified the taxonomy of intentional failures/attacks, based on the knowledge of the adversary. It deals with the extent of knowledge needed to trigger an attack for the AI/ML systems to fail. The adversary is better equipped with more knowledge [70] to perform the attack.

There are three types of classified attacks based on the adversary's access to knowledge about the system.

1 **Whiteb ox Attack:** In this type of attack, the adversary has access to the parameters of the underlying architecture of the model, the algorithm used for training, weights, training data distribution, and biases [71,72]. The adversary uses this information to find the model's vulnerable feature space. Later, the model is manipulated by modifying an input using adversarial crafting methods. An example of the whitebox attack and adversarial crafting methods are discussed in later sections. The researchers in [73,74] showed that adversarial training of the data, filled with some adversarial instances, actually helps the model/system become robust against whitebox attacks.

2 **Blackbox Attack:** In blackbox attacks the attacker does not know anything about the ML system. The attacker has access to only two types of information. The first is the hard label, where the adversary obtained only the classifier's predicted label, and the second is confidence, where the adversary obtained the predicted label along with the confidence score. The attacker uses information about the inputs from the past to understand vulnerabilities of the model [70]. Some blackbox attacks are discussed in later sections. Blackbox attacks can further be divided into three categories:

- **Non-Adaptive Blackbox Attack:** In this category of blackbox attack, the adversary has the knowledge of distribution of training data for a model, T. The adversary chooses a procedure, P, for a selected local model, T′, and trains the model on known data distribution using P for T′ to approximate the already learned T in order to trigger misclassification using whitebox strategies [53,75].
- **Adaptive Blackbox Attack:** In adaptive blackbox attack the adversary has no knowledge of the training data distribution or the model architecture. Rather, the attacker approaches the target model, T, as an oracle. The attacker generates a selected dataset with a label accessed from adaptive querying of the oracle. A training process, P, is chosen with a model, T′, to be trained on the labeled dataset generated by the adversary. The model T′ introduces the adversarial instances using whitebox attacks to trigger misclassification by the target model T [70,76].
- **Strict Blackbox Attack:** In this blackbox attack category, the adversary does not have access to the training data distribution but could have the labeled dataset (x, y) collected from the target model, T. The adversary can perturb the input to identify the changes in the output. This attack would be successful if the adversary has a large set of dataset (x,y) [70,71].

Grayb ox attacks: In whitebox attacks the adversary is fully informed about the target model, i.e., the adversary has access to the model framework, data distribution, training procedure, and model parameters, while in blackbox attacks, the adversary has no knowledge about the model. The graybox attack is an extended version of either whitebox attack or blackbox attack. In extended whitebox attacks, the adversary is partially knowledgeable about the target model setup, e.g, the model architecture, T, and the training procedure, P, is known, while the data distribution and parameters are unknown. On the other hand, in the extended blackbox attack, the adversarial model is partially trained, has different model architecture and, hence, parameters [77].

3.2. Anatomy of Cyberattacks

To build any machine learning model, the data needs to be collected, processed, trained, and tested and can be used to classify new data. The system that takes care of the sequence of data collection, processing, training and testing can be thought of as a generic AI/ML pipeline, termed the attack surface [70]. An attack surface subjected to adversarial intrusion may face poisoning attack, evasion attack, and exploratory attack. These attacks exploit three pillars of the information security, i.e., Confidentiality, Integrity, and Availability, known as the CIA triad [78]. Integrity of a system is compromised by the poisoning and evasion attacks, confidentiality is subject to intrusion by extraction, while availabilty is vulnerable to poisoning attacks. The entire AI pipeline, along with the possible attacks at each step, are shown in Figure 1.

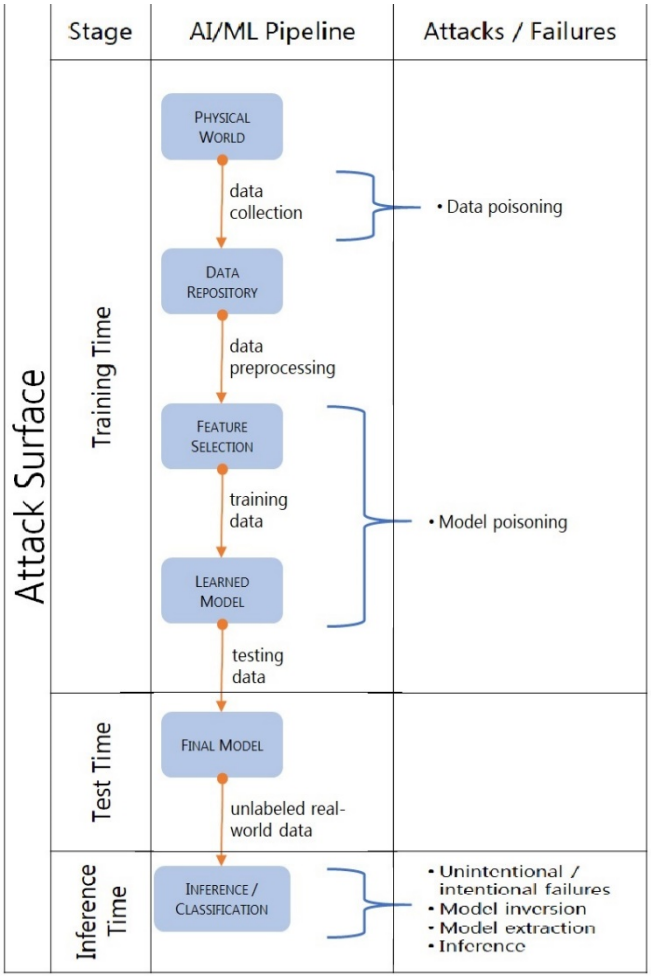

Figure 1. ML Pipeline with Cyberattacks Layout.

3.3. Poisoning Attack

Poisoning attack occurs when the adversary contaminates

the training data. Often ML algorithms, such as intrusion detection systems, are retrained on the training dataset. In this type of attack, the adversary cannot access the training dataset, but poisons the data by injecting new data instances [35,37,40] during the model training time. In general, the objective of the adversary is to compromise the AI system to result in the misclassification of objects.

Poisoning attacks can be a result of poisoning the training dataset or the trained model [1]. Adversaries can attack either at the data source, a platform from which a defender extracts its data, or can compromise the database of the defender. They can substitute a genuine model with a tainted model. Poisoning attacks can also exploit the limitations of the underlying learning algorithms. This attack happens in federated learning scenarios where the privacy on individual users' dataset is maintained [47]. The adversary takes advantage of the weakness of federated learning and may take control of both the data and algorithm on an individual user's device to deteriorate the performance of the model on that device [48].

3.3.1. Dataset Poisoning Attacks

The major scenarios of data poisoning attacks are error-agnostic poisoning attacks and error-specific poisoning attacks. In the error-agnostic type of poisoning attack the hacker aims to cause a Denial of Service (DOS) kind of attack. The hacker causes the system to produce errors, but it does not matter what type of error it is. For example, in a multi-class classification task a hacker could poison the data leading to misclassification of the data points irrespective of the class type, thus maximizing the loss function of the learning algorithm. To launch this kind of attack, the hacker needs to manipulate both the features and the labels of the data points. On the other hand, in error-specific poisoning attacks, the hacker causes the system to produce specific misclassification errors, resulting in security violation of both integrity and availability. Here, the hacker aims at misclassifying a small sample of chosen data points in a multi-class classification task. The hacker aims to minimize the loss function of the learning algorithm to serve the purpose, i.e., to force the system into misclassifying specific instances without compromising the normal system operation, ensuring that the attack is undetected [38,39].

A model is built up from a training dataset. So, attacking the dataset results in poisoning the model. By poisoning the dataset, the adversary could manipulate to generate natural adversarial examples, or inject instances with incorrect labels into the training dataset. The model may learn the pattern on misclassified examples in the data that serves the goal of the adversary. The dataset poisoning attacks can be further subdivided into two categories [79].

- **Data Modification**: The adversary updates or deletes training data. Here, the attacker does not have access to the algorithm. They can only manipulate labels. For instance, the attacker can draw new labels at random from the training pool, or can optimize the labels to cause maximum disruption.
- **Data Injection:** Even if the adversary does not have access to the training data or learning algorithm, he or she can still inject incorrect data into the training set. This is similar to manipulation, but the difference is that the adversary introduces new malicious data into the training pool, not just labels.

Support Vector Machines (SVMs) are widely used classification models for malware identification, intrusion detection systems, and filtering of spam emails, to name a few applications. Biggio, Nelson and Laskov [40] illustrated poisoning attacks on the SVM classifier, with the assumption that the adversary has information about the learning algorithm, and the data distribution. The adversary generates surrogate data from the data distribution and tampers with the training data, by introducing the surrogate data,

to drastically reduce the model training accuracy. The test data remains untouched. The authors formed an equation, expressing the adversarial strategy, as:

$$MAX_x\ A(x) = \sum_{i=1}^{k}(1 - y_i f_x(x_i)) = \sum_{i=1}^{k}(-g_i)$$

where $x_l \leq x \leq x_u$ and D = $(x_i,\ y_i\}_{i=1}^{k}$ is the validation data.

The goal of the adversary is to maximize the loss function $A(x)$ with the surrogate data instance $(x,\ y)$ to be added into the training set D_{tr} in order to maximally reduce the training accuracy of classification. $g_i$is the status of the margin, influenced by the surrogate data instance $(x,\ y)$.

Rubinstien et al. [80] presented the attack on SVM learning by exploiting training data confidentiality. The objective is to access the features and the labels of the training data by examining the classification on the test set.

Figure 2 explains the poison attack on the SVM classifier. The left sub-figure indicates the decision boundary of the linear SVM classifier, with support vectors and classifier margin. The right sub-figure shows how the decision boundary is drastically changed by tampering with one training data instance without changing the label of the instance. It was observed that the classification accuracy would be reduced by 11% by a mere 3% manipulation of the training set [81]. Nelson et al. [39] showed that an attacker can breach the functionality on the spam filter by poisoning the Bayesian classification model. The filter becomes inoperable under the proposed Usenet dictionary attack, wherein 36% of the messages are misclassified with 1% knowledge regarding the messages in the training set.

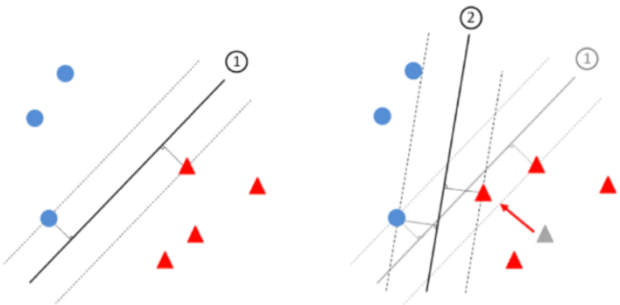

Figure 2. Poisoning attack changing the decision boundary.

Munoz-Gonzalez et al. [38] illustrated poisoning attacks on multi-class classification problems. The authors identified two attack scenarios for the multi-class problems: (1) error-generic poisoning attacks and (2) error-specific poisoning attacks. In the first scenario, the adversary attacks the bi-level optimization problem [40,82], where the surrogate data is segregated into training and validation sets. The model is learned on the generated surrogate training dataset with the tampered instances. The validation set measures the influence of the tampered instances on the original test set, by maximizing the binary class loss function. It is expressed in the following equation:

$$D_x^* = argmax_{D_x'} A\left(D_x', \sigma\right) = L(\overbrace{\widehat{D_{val}}, \widehat{\theta}})$$

$$Such\ as\ \widehat{\theta} = min_{w^*} L(\widehat{D_{tr}}\ U\ D_x',\ \theta^*)$$

The surrogate data $\widehat{D}$ is segregated into training $\widehat{D_{tr}}$ and validation sets $\widehat{D_{val}}$. The model is trained on $\widehat{D_{tr}}$ along with D_x' (i.e., the tampered instances). $\widehat{D_{val}}$ is used to measure the influence of the tainted samples on the genuine data via the function $A\left(D_x', \sigma\right)$ that explains the loss function, L, with respect to the validation dataset $\widehat{D_{val}}$ and the parameters $\widehat{\theta}$ of the surrogate model. In the multi-class scenario, the multi-class loss function is used for error-generic poisoning attacks.

In error-specific poisoning attacks, the objective remains to change the outcome of specific instances in a multi-class scenario. The goal of desired misclassification is expressed with the equation:

$$\left(D'_x, \sigma\right) = -L(\overbrace{\widehat{D^*_{val}}, \overbrace{\theta}})$$

$\widehat{D^*_{val}}$ is the same as the $\widehat{D_{val}}$ with different labels for desired misclassified instances that the adversary chose. The attacker aims to minimize the loss of the chosen misclassified samples.

In separate research, Kloft and Laskov [83] explained the adversarial attack on detection of outliers (anomalies), where the adversary is assumed to have knowledge about the algorithm and the training data. Their work introduced a finite sliding window, while updating the centre of mass iteratively for each new data instance. The objective is to accept the poisoned data instance as a valid data point, and the update on the center of mass is shifted in the direction of the tainted point, that appears to be a valid one. They show that relative displacement, d, of the center of mass under adversarial attack is lower bounded by the following inequality when the training window length is infinite:

$$d_i \leq ln(1 + \frac{i}{n})$$

where i and n are the number of tampered points and number of training points, respectively.

The intuition behind the use of anomaly detection is to sanitize the data by removing the anomalous data points, assuming the distribution of the anomalies is different from that of the normal data points. Koh, Steinhardt, and Liang [84] presented data poisoning attacks that outsmart data sanitization defenses for traditional anomaly detection, by nearest neighbors, training loss and singular value decomposition methods. The researchers divided the attacks into two groups:

- **High Sensitive**: An anomaly detector usually considers points as anomalous when the point is far off from its closest neighbors. The anomaly detector cannot identify a specific point as abnormal if it is surrounded by other points, even if that tiny cluster of points are far off from remaining points. So, if an adversary/attacker concentrates poison points in a few anomalous locations, then the anomalous location is considered benign by the detector.
- **Low Sensitive :** An anomaly detector drops all points away from the centroid by a particular distance. Whether the anomaly detector deems a provided point as abnormal does not vary much by addition or deletion of some points, until the centroid of data does not vary considerably.

Attackers can take advantage of this low sensitivity property of detectors and optimize the location of poisoned points such that it satisfies the constraints imposed by the defender.

Shafahi et al. [85] discussed how classification results can be manipulated just by injecting adversarial examples with correct labels. which is known as the clean-label attack. The clean-label attack is executed by changing the normal ("base") instance to reflect the features of another class, as shown in Figure 3. The Gmail image is marked with blue dots and lies on the feature space of the target dataset. This poisoned data is used for training and shifts the decision boundary, as shown in Figure 4.

Due to the shift, the target instance is classified as "base" instance. Here, the adversary tries to craft a poison instance such that it is indistinguishable from the base instance, i.e., the instance looks similar, and also minimizes the feature representation between the target and poison instances so that it triggers misclassification while training. This attack can be crafted using the optimization problem by means of the following equation:

$$p = \ argmin_x \ ||f(x) - f(t)||_2^2 + \ \beta * ||x - b||_2^2$$

where b is the base instance, and t and p are the target and poison instances, respectively. The parameter β identifies the degree to which p appears to be a normal instance to the human expert.

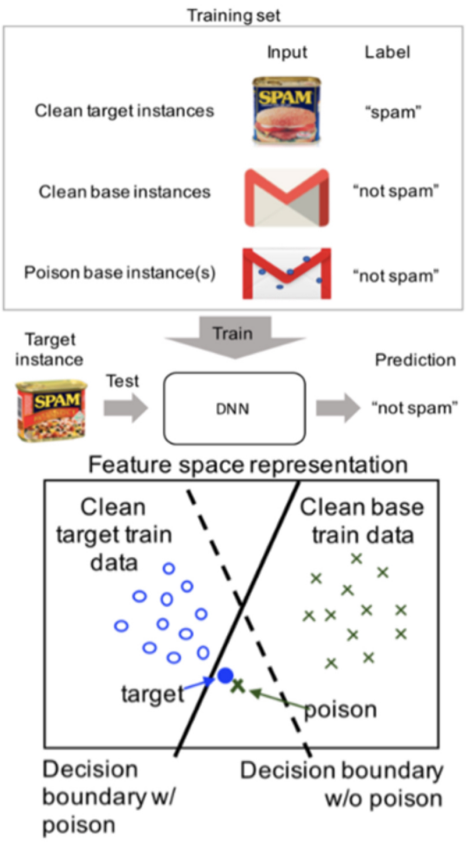

Figure 3. Clean-label attack procedure and example.

Figure 4. Badnet of MINST sample [42].

Suciu et al. [86] presented a similar type of attack on neural networks, but with the constraint that at least 12.5% of every mini-batch of training data should have tainted examples.

3.3.2. Data Poisoning Defense Mechanisms

There are studies that propose potential defense mechanisms to resolve the problems related to the data poisoning attacks discussed thus far. Devising a generic defense strategy against all attacks is not possible. The defense strategies are specific to the attack and a defense scheme specific to an attack makes the system susceptible to a different kind of attack. Some advanced defense strategies include:

1. **Adversarial Training :** The goal of adversarial training is to inject instances generated by the adversary into the training set to increase the strength of the model [87,88]. The defender follows the same strategy, by generating the crafted samples, using the

brute force method, and training the model by feeding the clean and the generated instances. Adversarial training is suitable if the instances are crafted on the original model and not on a locally-trained surrogate model [89,90].

2. **Feature Squee zing:** This defense strategy hardens the training models by diminishing the number of features and, hence, the complexity of data [91]. This, in turn, reduces the sensitivity of the data, which evades the tainted data marked by the adversary.
3. **Transferability blocking:** The true defense mechanism against blackbox attacks is to obstruct the transferability of the adversarial samples. The transferability enables the usage of adversarial samples in different models trained on different datasets. Null labeling [92] is a procedure that blocks transferability, by introducing null labels into the training dataset, and trains the model to discard the adversarial samples as null labeled data. This approach does not reduce the accuracy of the model with normal data instances.
4. **MagNet:** This scheme is used to arrest a range of blackbox attacks through the use of a detector and a reformer [93]. The detector identifies the differences between the normal and the tainted samples by measuring the distance between them with respect to a threshold. The reformer converts a tampered instance to a legitimate one by means of an autoencoder.
5. **Defense-GAN**: To stave off both blackbox and whitebox attacks, the capability of General Adversarial Network (GAN) [94] is leveraged [95]. GAN uses a generator to construct the input images by minimizing the reconstruction error. The reconstructed images are fed to the system as input, where the genuine instances are closer to the generator than the tainted instances. Hence, the performance of the attack degrades.
6. **Local Intrinsic Dimensionality**: Weerashinghe et al. [96] addressed resistance against data poisoning attack on SVM classifiers during training. They used Local Intrinsic Dimensionality (LID), a metric of computing dimension of local neighborhood subspace for each data instance. They also used K-LID approximation for each sample to find the likelihood ratio of K-LID values from the distribution of benign samples to that from tainted samples. Next, the function of the likelihood ratio is fitted to predict the likelihood ratio for the unseen data points' K-LID values. The technique showed stability against adversarial attacks on label flipping.
7. **Reject On Negative Impact (RONI):** The functioning of the RONI technique is very similar to that of the Leave-One-Out (LOO) validation procedure [97]. Although effective, this technique is computationally expensive and may suffer from overfitting if the training dataset used by the algorithm is small compared to the number of features. RONI defense is not well suited for applications that involve deep learning architectures, as those applications would demand a larger training dataset [39]. In [98], a defensive mechanism was proposed based on the k-Nearest Neighbors technique, which recommends relabeling possible malicious data points based on the labels of their neighboring samples in the training dataset. However, this strategy fails to detect attacks in which the subsets of poisoning points are close. An outlier detection scheme was proposed in [99] for classification tasks. In this strategy, the outlier detectors for each class are trained with a small fraction of trusted data points. This strategy is effective in attack scenarios where the hacker does not model specific attack constraints. For example, if the training dataset is poisoned only by flipping the labels, then this strategy can detect those poisoned data points which are far from the genuine ones. Here, it is important to keep in mind that outlier detectors used in this technique need to first be trained on small curated training points that are known to be genuine [99].

In many studies, the defense strategies are for the time of filtering of data during anomaly detection (i.e., before the model is trained). Koh, Steinhardt, and Liang [84] considered data sanitization defenses of five different types, from the perspective of anomaly detection, each with respective anomaly detection parameters β and parametrarized scores

s_β which identify the degree of anomaly. D_{clean} and D_{poison} are the datasets for clean and poisoned instances $D = D_{clean} \cup D_{poisin}$ and β is derived from D.

(1) **L-2 Defense:** This type of defense discards the instances that are distant from the center of the corresponding class they belong to, from the perspective of the L-2 distance measure. The outlier detection parameter and parametrarized score for the L-2 defense are expressed as:

$$\beta_y = Expectation_D(x|y)$$

$$S_\beta(x,y) = \ ||x - \ \beta_y||_2$$

(2) **Slab Defense:** Slab defense [81] draws the projections of the instances on the lines or planes joining the class centers and discards those that are too distant from the centers of the classes. Unlike the L-2 defense, this mechanism considers only the distances between the class centers as pertinent dimensions. The outlier detection parameter and parametrarized score for the slab defense are expressed as:

$$\beta_y = Expectation_D(x|y)$$

$$S_\beta(x,y) = |(\beta_1 - \beta_{-1})^T(x - \ \beta_y)|$$

where θ is the learning parameter that minimizes the training loss, x denotes the data point and y is the class.

(3) **Loss Defense:** Loss defense removes points that are not fitted well by the trained model on D. The feature dimensions are learned based on loss function l. The outlier detection parameter and parametrarized score for the loss defense are expressed as:

$$\beta_y = argmin_\theta Expectation_D l_\theta[(x|y)]$$

$$S_\beta(x,y) = l_\beta(x|y)$$

(4) **SVD Defense :** SVD defense is the mechanism that works on the basis of sub-space assumption [100]. In this defense mechanism the normal instances are assumed to lie in low-ranked sub-space while the tampered instances have components that are too large to fit into this sub-space. The outlier detection parameter and parametrarized score for the loss defense are expressed as:

$$\beta = |M|^k_{RSV}$$

$$S_\beta(x,y) = \ ||\left(I - \ \beta\beta^T\right)x||_2$$

The term $|M|^k_{RSV}$ is the matrix of $S_\beta(x,y) = \ |((I - \ \beta\beta^T)x||_2$ right singular vector of data matrix d.

(5) **K-NN Defense:** The K-NN defense discards data instances that are distant from the K nearest neighbors. The outlier detection parameter and parametrarized score for the k-NN defense are expressed as:

$$\beta = D$$

$$S_\beta(x,y) = dist_{\text{k-NN}} \in \beta$$

Koh, Steinhardt, and Liang [84] have tested these 5 types of data sanitization defenses on four types of datasets: The MNIST dataset [101], Dogfish [102], Enron spam detection [103] and the IMDB sentiment classification datasets [104]. The first two datasets are image datasets. The results showed that these defenses could still be evaded with concentrated attacks where the instances concentrated in a few locations appear to be normal. However, it was observed that L-2, slab and loss defenses still diminished the test error (which is exploited by the adversary to launch a data poisoning attack) considerably, compared to the SVD and k-NN defenses.

Peri et al. [105] proposed a defense mechanism resisting clean-label poison attacks, based on k-NN, and identified 99% of the poisoned instances, which were eventually discarded before model training. The authors claimed that this scheme, known as Deep K-NN, worked better than the schemes provided by [84], without reducing the model's performance.

3.3.3. Model Poisoning Attacks

Poisoning of models is more like a traditional cyberattack. If attackers breach the AI system, then either they can compromise the existing AI model with the poisoned one or they can execute "A man in the middle" attack [106] to have the wrong model downloaded, while transferring learning.

Model poisoning is generally done using Backdoored Neural Network (BadNet) attack [45]. BadNets are modified neural networks, in which the model is trained on clean and poisoned inputs. In this, the training mechanism is fully or partly outsourced to the adversary, who returns the model with secret backdoor inputs. Secret backdoor inputs are inputs added by the attacker which result in misclassification. The inputs are known only to the attacker. BadNet is categorized into two related classes:

1. **Outsource training attack**, when training is outsourced, and
2. **Transfer learning attack**, when a pre-trained model is outsourced and used.

In the following subsections, we also explore model poisoning attacks on the federated learning scenario, where the training of the model is distributed on multiple computing devices and the results of the training are aggregated from all the devices to form the final training model. Bhagoji et al. [41] classified the model poisoning attack strategies on federated learning scenarios as: (1) explicit boosting, and (2) alternating minimization.

Outsourced Training Attack

We want to train the parameters of a model, M. using the training data. We outsource the description of M to the trainer who sends the learned parameters back to us β_M. Our trustability of the trainer depends on a privately held validation dataset, with a targeted accuracy, or on the service agreement between us and the trainer.

The objective of the adversary is to return a corrupted model with backdoored trained parameters β'_M. This is different from β_M and either should not lower the validation accuracy or decrease the model accuracy of the inputs with a backdoor trigger. Thus, the training attack can be targeted or untargeted. In a targeted attack, the adversary switches the label of the outputs for specific inputs, while in an untargeted attack, the input of the backdoored property remains misclassified to degrade the overall model accuracy.

Figure 4 depicts an example of backdoor attacks where the second and third images are the original image's backdoored version, whereas Figure 5 depicts an example of BadNet attacks on traffic images.

Figure 5. Badnet Example [42].

Figure 6 illustrates a special type of potential BadNet (i.e., BadNet with backdoor detector) which makes use of a parallel link to identify the backdoor trigger. It also uses a combining layer to produce misclassifications if the backdoor appears. This perturbed model would not impact the results on a cleaned dataset, so the user would not be able to identify if the model has been compromised.

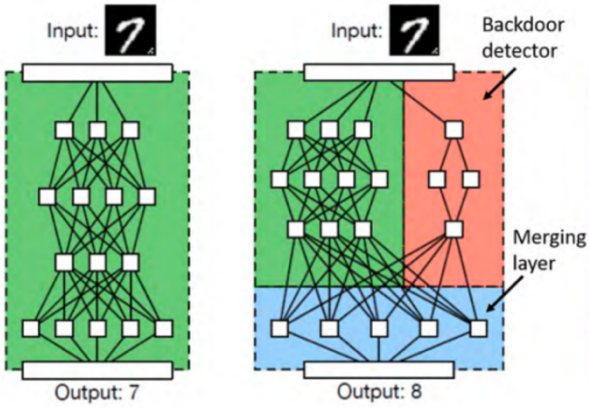

Figure 6. Badnet Model [42].

In terms of defense mechanisms, Backdoor attacks like BadNet happen when we use pre-trained models. So, the less pre-trained the model, the less the attack. However, today, almost all networks are built using pre-trained models.

To make the models robust against backdoor attacks, Gu et al. [42] proposed the following defense strategies:

- Securely hosting and disseminating pre-trained models in virtual repositories that guarantee integrity, to preclude benevolent models from being manipulated. The security is characterized by the fact that virtual archives should have digital signatures of the trainer on the pre-trained models with the public key cryptosystem [43].
- Identifying backdoors in malevolently trained models acquired from an untrustworthy trainer by retraining or fine-tuning the untrusted model with some added computational cost [44,46]. These researchers considered fully outsourced training attacks. Another research [107], proposed a defense mechanism with an assumption that the user has access to both clean and backdoored instances.

Transfer Learning Attack

The objective of transfer learning is to save computation time, by transferring the knowledge of an already-trained model to the target model [45]. The models are stored in online repositories from where a user can download them for an AI/ML application. If the downloaded model, M_{cor}, is a corrupted model, then, while transferring learning, the user generates his/her model and parameters based on M_{cor}. In transfer learning attacks, we assume that the newly adapted model, M_{cor}, and the uncorrupted model have the same input dimensions but differ in number of classes.

Figure 7 compares a good network (left), that rightly classifies its input, to BadNet (right), that gives misclassifications but has the same architecture as the good network.

Figure 8 describes the transfer learning attack setup with backdoor strengthening factor to enhance the impact of weights.

In terms of potential defense mechanisms, the obvious defense strategy is to obtain pre-trained models from trusted online sources, such as Caffe Model Zoo and Keras trained Model Library [108], where a secure cryptographic hashing algorithm (e.g., SHA-1) is used as a reference to verify the downloads. However, the researchers in [42] showed that downloaded BadNet from "secure" online model archives can still hold the backdoor property, even when the user re-trains the model to perform his/her tasks.

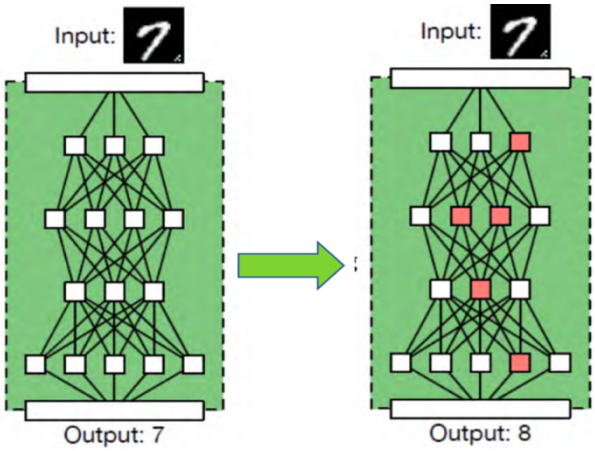

Figure 7. Transfer learning using the BadNet [42].

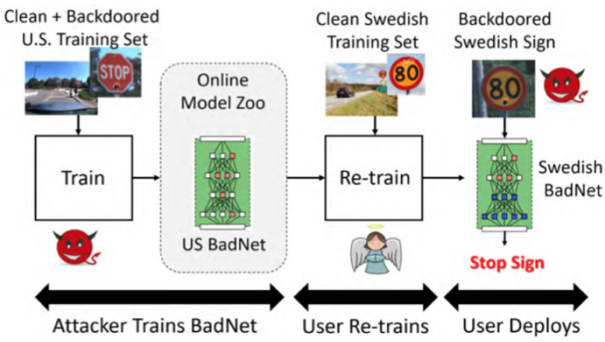

Figure 8. Transfer Learning set up attacks [42].

Wu et al. [109] devised methodologies to resolve transfer learning attacks related to misclassification. They proposed activation-based pruning [110] and developed the distilled differentiator, based on pruning. To augment strength against attacks, the ensemble construct from the differentiators is implemented. As the individual distilled differentiators are diverse, in activation-based pruning, different training examples promote divergence among the differentiators; hence, increasing the strength of ensemble models. Pruning changes the model structure and arrests the portability of attack from one system to the other [44,46]. Network pruning removes the connectives between the model and generates a sparse model from a dense network model. The sparsity helps in fine tuning the model and eventually discarding the virulence of the attacks. Comprehensive evaluations, based on classification accuracy, success rate, size of the models, and time for learning, regarding the defense strategies suggested by the authors, on image recognition showed the new models, with only five differentiators, to be invulnerable against more than 90% of adversarial inputs, with accuracy loss less than 10%.

Attack on Federated Learning

In the federated learning scenario, each and every individual device has its own model to train, securing the privacy of the data stored in that device [47]. Federated learning algorithms are susceptible to model poisoning if the owner of the device becomes malicious. Research [111,112] introduced a premise for federated learning, where a single adversary attacks the learning by changing the gradient updates to arbitrary values, instead of introducing the backdoor property into the model. The objective of the attacker is to obstruct the convergence of the execution of the distributed Stochastic Gradient Descent (SGD) algorithm. In a similar study, Bagdasaryan et al. [48] proposed a multi-agent framework, where multiple adversaries jointly conspired to replace the model during model covergence. Bhagoji et al. [41] worked on targeted misclassification by introducing a sequence of attacks

induced by a single adversary: (1) explicit boosting, and (2) alternating minimization. The underlying algorithm is SGD.

- **Explicit Boosting:** The adversary updates the boosting steps to void the global aggregated effect of the individual models locally distributed over different devices. The attack is based on running of boosting steps of SGD until the attacker obtains the parameter weight vector, starting from the global weight, to minimize the training loss over the data and the class label. This enables the adversary to obtain the initial update, which is used to determine the final adversarial update. The final update is obtained by the product of the final adversarial update and the inverse of adversarial scaling (i.e., the boosting factor), so that the server cannot identify the adversarial effect.
- **Alternating Minimization:** The authors in [45] showed that, in an explicit boosting attack, the malicious updates on boosting steps could not evade the potential defense related to measuring accuracy. Alternating minimization was introduced to exploit the fact that it is updates related only to the targeted class that need to be boosted. This strategy improves adversarial attack that can bypass the defense mechanism with the goal of minimizing training loss and boosting parameter updates for the adversarial goals and achieved a high success rate.

In terms of potential defense mechanisms, two typical strategies are deployed, depending on the nature of the attacks on federated learning: (1) robust aggregation methods, and (2) robust learning rate.

- **Robust aggregation methods:** These methods incorporate security into federated learning by exploring different statistical metrics that could replace the average (mean) statistic, while aggregating the effects of the models, such as trimmed mean, geometric median, coordinate-median, etc [47,111,113–116]. Introducing the new statistic while aggregating has the primary objective of staving off attacks during model convergence. Bernstein et al. [117] proposed a sign aggregation technique on the SGD algorithm, distributed over individual machines or devices. The devices interact with the server by communicating the signs of the gradients. The server aggregates the signs and sends this to the individual machines, which use it to update their model weights. The weight update rule can be expressed by the following equation:

$$w_{t+1} = w_t + \gamma(sgn\sum_{i \in A_t} sgn(\Delta_t^i))$$

where Δ_t^i is the weight update of the device i at time t. $\Delta_t^i = w_t^k - w_t.w_t$ is the weight the server sent to the set of devices A_t at time t and γ is the server learning rate.

This approach is robust against convergence attacks, but susceptible to backdoor attacks in federated learning scenarios.

In a recent study, [118] the authors modified the mean estimator of the aggregate by introducing weight-cutoff and addition of noise [119] during weight update to deter backdoor attacks. In this method, the server snips the weights when the L2 norm of a weight update surpasses a pre-specified threshold, and then aggregates the snipped weights, along with the noise, during aggregation of weights.

- **Robust Learning Rate:** Ozdayi, Katancioglu, and Gel [120] introduced the defense mechanism by making the model learning rate robust with a pre-specified boundary of malicious agents. With the help of the updated learning rate, the adversarial model weight approaches the direction of the genuine model weight. This work is an extension of the signed aggregation proposed in [117]. The authors proposed a parameter-learning threshold δ. The learning rate for the i-th dimension of the data can be represented as:

$$\gamma_{\delta,i} = \begin{cases} \gamma & \text{if } \left|\sum_{k \in S_t} \text{sgn}\left(\Delta_{t,i}^k\right)\right| \geq \delta \\ -\gamma & \text{otherwise} \end{cases}$$

The server weight update at time $t + 1$ is

$$w_{t+1} = w_t + \gamma_\delta \odot \frac{\sum_{k \in S_t} n_k \triangle_t^k}{\sum_{k \in S_t} n_k}$$

where γ_δ is the overall learning rate, including all dimensions, and $\odot$ is the feature-wise product operation. Δ^k is the update on the gradient descent update of the k-th player in the system, and k may be the adversary or the regular user.

3.4. Model Inversion Attack

The model inversion attack is a way to reconstruct the training data, given the model parameters. This type of attack is a concern for privacy, because there are a growing number of online model repositories. Several studies related to this attack hve been under both the blackbox and whitebox settings. Yang et al. [121] discussed the model inversion attack in the blackbox setting, where the attacker wants to reconstruct an input sample from the confidence score vector determined by the target model. In their study, they demonstrated that it is possible to reconstruct specific input samples from a given model. They trained a model (inversion) on an auxiliary dataset, which functioned as the inverse of the given target model. Their model then took the confidence scores of the target model as input and tried to reconstruct the original input data. In their study, they also demonstrated that their inversion model showed substantial improvement over previously proposed models. On the other hand, in a whitebox setting, Fredrikson et al. [122] proposed a model inversion attack that produces only a representative sample of a training data sample, instead of reconstructing a specific input sample, using the confidence score vector determined by the target model. Several related studies were proposed to infer sensitive attributes [122–125] or statistical information [126] about the training data by developing an inversion model. Hitaj et al. [71] explored inversion attacks in federated learning where the attacker had whitebox access to the model.

Several defense strategies against the model inversion attack have been explored that include L2 Regularizer [49], Dropout and Model Staking [50], MemGuard [51], and Differential privacy [52]. These defense mechanisms are also well-known for reducing overfitting in the training of deep neural network models.

3.5. Model Extraction Attack

A machine learning model extraction attack arises when an attacker obtains black-box access to the target model and is successful in learning another model that closely resembles. or is exactly the same as, the target model. Reith et al. [54] discussed model extraction against the support vector regression model. Juuti et al. [127] explored neural networks and showed an attack, in which an adversary generates queries for DNNs with simple architectures. Wang et al., in [128], proposed model extraction attacks for stealing hyperparameters against a simple architecture similar to a neural network with three layers. The most elegant attack, in comparison to the others, was shown in [129]. They showed that it is possible to extract a model with higher accuracy than the original model. Using distillation, which is a technique for model compression, the authors in [130,131], executed model extraction attacks against DNNs and CNNs for image classification.

To defend against model extraction attacks, the authors in [53,132,133] proposed either hiding or adding noises to the output probabilities, while keeping the class label of the instances intact. However, such approaches are not very effective in label-based extraction attacks. Several others have proposed monitoring the queries and differentiating suspicious queries from others by analyzing the input distribution or the output entropy [127,134].

3.6. Inference Attack

Machine learning models have a tendency to leak information about the individual data records on which they were trained. Shokri et al. [49] discussed the membership

inference attack, where one can determine if the data record is part of the model's training dataset or not, given the data record and blackbox access to the model. According to them, this is a concern for privacy breach. If the advisory can learn if the record was used as part of the training, from the model, then such a model is considered to be leaking information. The concern is paramount, as such a privacy beach not only affects a single observation, but the entire population, due to high correlation between the covered and the uncovered dataset [135]. This happens particularly when the model is based on statistical facts about the population.

Studies in [136–138] focused on attribute inference attacks. Here an attacker gets access to a set of data about a target user, which is mostly public in nature, and aims to infer the private information of the target user. In this case, the attacker first collects information from users who are willing to disclose it in public, and then uses the information as a training dataset to learn a machine learning classifier which can take a user's public data as input and predict the user's private attribute values.

In terms of potential defense mechanisms, methods proposed in [55,139] leveraged heuristic correlations between the records of the public data and attribute values to defend against attribute inference attacks. They proposed modifying the identified k entries that have large correlations with the attribute values to any given target users. Here k is used to control the privacy–utility trade off. This addresses the membership inference attack.

4. Conclusions

Using an extensive survey of the literature, this research addresses two research questions regarding attacks on AI systems and their potential defense mechanisms.

RQ1: What are the cyberattacks that AI systems can be subjected to?

To answer this question, we discussed different categories of intentional and unintentional failures, along with the details of poisoning attacks on data and machine learning models. We also introduced backdoored neural network (discussing it from the perspective of research carried out on outsourced training attacks, transfer learning attack and federated learning attacks), model inversion, model extraction and inference attacks.

RQ2: Can the attacks on AI systems be organized into a taxonomy, to better understand how the vulnerabilities manifest themselves during system development?

Upon reviewing the literature related to attacks on AI systems, it was evident that, at different stages of the AI/ML pipeline development, vulnerabilities manifest; thus, providing an opportunity to launch attacks on the AI system. Table 1 and Figure 1 organize the AI attacks into a taxonomy, to better understand how vulnerabilities manifest and how attacks can be launched during the entire system development process.

RQ3: What are possible defense mechanisms to defend AI systems from cyberattacks?

While addressing the second research question, we reviewed multiple state of the art methods that are used as potential defense mechanisms for each type of attack.

RQ4: Is it possible to device a generic defense mechanism against all kinds of AI attacks?

Based on the literature review of cyberattacks on AI systems. it is clearly evident that there is no single. or generic, defense mechanism that can address diverse attacks on AI systems. Vulnerabilities that manifest in AI systems are more specific to the system design and its composition. Therefore, a defense mechanism has to be tailored, or designed, in such a way that it can suit the specific characteristics of the system.

This survey sheds light on the different types of cybersecurity attacks and their corresponding defense mechanisms in a detailed and comprehensive manner. Growing threats and attacks in emerging technologies, such as social media, cloud computing, AI/ML systems, data pipelines and other critical infrastructures, often manifest in different forms. It is worth noting that it is challenging to capture all patterns of threats and attacks. Therefore, this survey attempted to capture a common set of general threat and attack patterns that are specifically targeted towards AI/ML systems. Organizing this body of knowledge. from the perspective of an AI system's life cycle, can be useful for software engineering teams when designing and developing intelligent systems. In addition, this survey offers

a profound benefit to the research community focused on analyzing the cybersecurity of AI systems. Researchers can implement and replicate these attacks on an AI system, systematically apply defenses against these attacks, understand the trade offs that arise from using defense mechanisms, and create a catalog of patterns or tactics for designing trustworthy AI systems.

Author Contributions: Conceptualization, Y.B., R.S.S. and S.M.S.; methodology, Y.B., R.S.S. and S.M.S.; writing and editing, Y.B. and R.S.S.; review, R.S.S. and S.M.S., funding acquisition, Y.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Penn State InudstryXchange 2021.

Acknowledgments: In memoriam: "Partha, the bond between friends cannot be broken by death. You will be greatly missed." (Y.B.).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Comiter, M. Attacking artificial intelligence: AI's security vulnerability and what policymakers can do about it. *Harv. Kennedy Sch. Belfer Cent. Sci. Int. Aff.* **2019**, 1–90. Available online: https://www.belfercenter.org/sites/default/files/2019-08/AttackingAI/AttackingAI.pdf (accessed on 8 March 2023) .
2. Mcgraw, G.; Bonett, R.; Figueroa, H.; Shepardson, V. Security engineering for machine learning. *IEEE Comput.* **2019**, *52*, 54–57. [CrossRef]
3. Ma, Y.; Xie, T.; Li, J.; Maciejewski, R. Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE Trans. Vis. Comput. Graph.* **2019**, *26*, 1075–1085. [CrossRef] [PubMed]
4. Kim, J.; Park, N. Blockchain-based data-preserving AI learning environment model for AI cybersecurity systems in IoT service environments. *Appl. Sci.* **2020**, *10*, 4718. [CrossRef]
5. Mozaffari-Kermani, M.; Sur-Kolay, S.; Raghunathan, A.; Jha, N.K. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE J. Biomed. Health Inform.* **2014**, *19*,1893–1905. [CrossRef] [PubMed]
6. Sadeghi, K.; Banerjee, A.; Gupta, S.K.S. A system-driven taxonomy of attacks and defenses in adversarial machine learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2020**, *4*, 450–467. [CrossRef]
7. Sagar, R.; Jhaveri, R.; Borrego, C. Applications in security and evasions in machine learning: A survey. *Electronics* **2020**, *9*, 97. [CrossRef]
8. Pitropakis, N.; Panaousis, E.; Giannetsos, T.; Anastasiadis, E.; Loukas, G. A taxonomy and survey of attacks against machine learning. *Comput. Sci. Rev.* **2019**, *34*, 100199. [CrossRef]
9. Cao, N.; Li, G.; Zhu, P.; Sun, Q.; Wang, Y.; Li, J.; Yan, M.; Zhao, Y. Handling the adversarial attacks. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 2929–2943. [CrossRef]
10. Wang, X.; Li, J.; Kuang, X.; Tan, Y.; Li, J. The security of machine learning in an adversarial setting: A survey. *J. Parallel Distrib. Comput.* **2019**, *130*, 12–23. [CrossRef]
11. Rouani, B.D.; Samragh, M.; Javidi, T.; Koushanfar, F. Safe machine learning and defeating adversarial attacks. *IEEE Secur.* **2019**, *17*, 31–38. [CrossRef]
12. Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of artificial intelligence adversarial attack and defense technologies. *Appl. Sci.* **2019**, *9*, 909. [CrossRef]
13. Biggio, B.; Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognit.* **2018**, *84*, 317–331. [CrossRef]
14. Sethi, T.S.; Kantardzic, M.; Lyu, L.; Chen, J. A dynamic-adversarial mining approach to the security of machine learning. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1245. [CrossRef]
15. Chen, T.; Liu, J.; Xiang, Y.; Niu, W.; Tong, E.; Han, Z. Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
16. Li, G.; Ota, K.; Dong, M.; Wu, J.; Li, J. DeSVig: Decentralized swift vigilance against adversarial attacks in industrial artificial intelligence systems. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3267–3277. [CrossRef]
17. Garcia-Ceja, E.; Morin, B.; Aguilar-Rivera, A.; Riegler, M.A. A Genetic Attack Against Machine Learning Classifiers to Steal Biometric Actigraphy Profiles from Health Related Sensor Data. *J. Med. Syst.* **2020**, *44*, 1–11. [CrossRef]
18. Biggio, B.; Russu, P.; Didaci, L.; Roli, F. Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective. *IEEE Signal Process. Mag.* **2015**, *32*, 31–41. [CrossRef]
19. Ren, Y.; Zhou, Q.; Wang, Z.; Wu, T.; Wu, G.; Choo, K.K.R. Query-efficient label-only attacks against black-box machine learning models. *Comput. Secur.* **2020**, *90*, 101698. [CrossRef]
20. Wang, D.; Li, C.; Wen, S.; Nepal, S.; Xiang, Y. Man-in-the-middle attacks against machine learning classifiers via malicious generative models. *IEEE Trans. Dependable Secur. Comput.* **2020**, 18, 2074–2087. [CrossRef]

21. Qiu, J.; Du, L.; Chen, Y.; Tian, Z.; Du, X.; Guizani, M. Artificial intelligence security in 5G networks: Adversarial examples for estimating a travel time task. *IEEE Veh. Technol. Mag.* **2020**, *15*, 95–100. [CrossRef]
22. Benzaid, C.; Taleb, T. AI for beyond 5G networks: a cyber-security defense or offense enabler? *IEEE Networks* **2020**, *34*, 140–147. [CrossRef]
23. Apruzzese, G.; Andreolini, M.; Marchetti, M.; Colacino, V.G.; Russo, G. AppCon: Mitigating Evasion Attacks to ML Cyber Detectors. *Symmetry* **2020**, *12*, 653. [CrossRef]
24. Zhang, S.; Xie, X.; Xu, Y. A brute-force black-box method to attack machine learning-based systems in cybersecurity. *IEEE Access* **2020**, *8*, 128250–128263. [CrossRef]
25. Liu, K.; Yang, H.; Ma, Y.; Tan, B.; Yu, B.; Young, E.F.; Karri, R.; Garg, S. Adversarial perturbation attacks on ML-based cad: A case study on CNN-based lithographic hotspot detection. *ACM Trans. Des. Autom. Electron. Syst.* **2020**, *25*, 1–31. [CrossRef]
26. Katzir, Z.; Elovici, Y. Quantifying the resilience of machine learning classifiers used for cyber security. *Expert Syst. Appl.* **2018**, *92*, 419–429. [CrossRef]
27. Chen, S.; Xue, M.; Fan, L.; Hao, S.; Xu, L.; Zhu, H.; Li, B. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Comput. Secur.* **2018**, *73*, 326–344. [CrossRef]
28. Gardiner, J.; Nagaraja, S. On the security of machine learning in malware c&c detection: A survey. *ACM Comput. Surv.* **2016**, *49*, 1–39.
29. Dasgupta, P.; Collins, J. A survey of game theoretic approaches for adversarial machine learning in cybersecurity tasks. *AI Mag.* **2019**, *40*, 31–43. [CrossRef]
30. Al-Rubaie, M.; Chang, J.M. Privacy-preserving machine learning: Threats and solutions. *IEEE Secur. Priv.* **2019**, *17*, 49–58. [CrossRef]
31. Hansman, S.; Hunt, R. A taxonomy of network and computer attacks. *Comput. Secur.* **2005**, *24*, 31–43. [CrossRef]
32. Gao, J.B.; Zhang, B.W.; Chen, X.H.; Luo, Z. Ontology-based model of network and computer attacks for security assessment. *J. Shanghai Jiaotong Univ.* **2013**, *18*, 554–562. [CrossRef]
33. Gonzalez, L.M.; Lupu, E.; Emil, C. The secret of machine learning. *ITNow* **2018**, *60*, 38–39. [CrossRef]
34. Mcdaniel, P.; Papernot, N.; Celik, Z.B. Machine learning in adversarial settings. *IEEE Secur. Priv.* **2016**, *14*, 68–72. [CrossRef]
35. Barreno, M.; Nelson, B.; Joseph, A.D.; Tygar, J.D. The security of machine learning. *Mach. Learn.* **2010**, *81*, 121–148. [CrossRef]
36. Barreno, M.; Nelson, B.; Sears, R.; Joseph, A.D.; Tygar, J.D. Can machine learning be secure? In Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, Taipei, Taiwan, 21–24 March 2006; pp. 16–25.
37. Biggio, B.; Fumera, G.; Roli, F. Security evaluation of pattern classifiers under attack. *IEEE Trans. Knowl. Data Eng.* **2013**, *26*, 984–996. [CrossRef]
38. Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E.C.; Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017; pp. 27–38.
39. Nelson, B.; Barreno, M.; Chi, F.J.; Joseph, A.D.; Rubinstein, B.I.; Saini, U.; Sutton, C.; Tygar, J.D.; Xia, K. Exploiting machine learning to subvert your spam filter. In Proceedings of First USENIX Workshop on Large Scale Exploits and Emergent Threats, **2008**, *8*, 1–9.
40. Biggio, B.; Nelson, B.; Laskov, P. Poisoning attacks against support vector machines. *arXiv* **2012**, arXiv:1206.6389.
41. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Model poisoning attacks in federated learning. In Proceedings of the Workshop on Security in Machine Learning (SecML), collocated with the 32nd Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7 December 2018.
42. Gu, T.; Liu, K.; Dolan-Gavitt, B.; Garg, S. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* **2019**, *7*, 47230–47244. [CrossRef]
43. Samuel, J.; Mathewson, N.; Cappos, J.; Dingledine, R. Survivable key compromise in software update systems. In Proceedings of the 17th ACM conference on Computer and communications security, Chicago, IL, USA, 4–8 October 2010; pp. 61–72.
44. Liu, K.; Dolan-Gavitt, B.; Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, Heraklion, Crete, Greece, 10–12 September 2018; pp. 273–294.
45. Gu, T.; Dolan-Gavitt, B.; Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv* **2017**, arXiv:1708.06733.
46. Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; Zhao, B.Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 707–723.
47. Mcmahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference of Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
48. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to backdoor federated learning. In Proceedings of International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020; pp. 2938–2948.
49. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 3–18

50. Salem, A.; Zhang, Y.; Humbert, M.; Berrang, P.; Fritz, M.; Backes, M. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. *arXiv* **2018**, arXiv:1806.01246.
51. Jia, J.; Salem, A.; Backes, M.; Zhang, Y.; Gong, N.Z. Memguard: Defending against black-box membership inference attacks via adversarial examples. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 259–274.
52. Dwork, C.; Mcsherry, F.; Nissim, K.; Smith, A. Calibrating noise to sensitivity in private data analysis. *Theory Cryptogr. Conf.* **2006**, *3876*, 265–284.
53. Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M.K.; Ristenpart, T. Stealing machine learning models via prediction apis. *USENIX Secur. Symp.* **2016**, *16*, 601–618.
54. Reith, R.N.; Schneider, T.; Tkachenko, O. Efficiently stealing your machine learning models. In Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, London, UK, 11 November 2019; pp. 198–210.
55. Weinsberg, U.; Bhagat, S.; Ioannidis, S.; Taft, N. BlurMe: Inferring and obfuscating user gender based on ratings. In Proceedings of the sixth ACM conference on Recommender systems, Dublin, Ireland, 9–13 September 2012; pp. 195–202.
56. Kaloudi, N.; Li, J. The AI-based cyber threat landscape: A survey. *ACM Comput. Surv.* **2020**, *53*, 1–34. [CrossRef]
57. Turchin, A. A Map: AGI Failures Modes and Levels, 2023. Available online: https://www.lesswrong.com/posts/hMQ5iFiHkChqgrHiH/a-map-agi-failures-modes-and-levels (accessed on 8 March 2023).
58. Turchin, A.; Denkenberger, D. Classification of global catastrophic risks connected with artificial intelligence. *AI Soc.* **2020**, *35*, 147–163. [CrossRef]
59. Yampolskiy, R.V. Taxonomy of pathways to dangerous artificial intelligence. In Proceedings of the Workshops at the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–13 February 2016; pp. 143–158.
60. Kumar, R.S.S.; Brien, D.O.; Albert, K.; Viljöen, S.; Snover, J. 2019. Failure Modes in Machine Learning. Available online: https://arxiv.org/ftp/arxiv/papers/1911/1911.11034.pdf (accessed on 8 March 2023).
61. Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S.; Dragan, A. Inverse Reward Design. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. Available online: https://proceedings.neurips.cc/paper/2017/hash/32fdab6559cdfa4f167f8c31b9199643-Abstract.html (accessed on 8 March 2023)
62. Yuan, Y.; Yu, Z.L.; Gu, Z.; Deng, X.; Li, Y. A novel multi-step reinforcement learning method for solving reward hacking. *Appl. Intell.* **2019**, *49*, 2874–2888. [CrossRef]
63. Leike, J.; Martic, M.; Krakovna, V.; Ortega, P.A.; Everitt, T.; Lefrancq, A.; Orseau, L.; Legg, S. AI safety Gridworlds. *arXiv* **2017**, arXiv:1711.09883.
64. Zhang, A.; Lipton, Z.C.; Li, M.; Smola, A. Dive into Deep Learning. *arXiv* **2021**, arXiv:2106.11342.
65. Subbaswamy, A.; Saria, S. From development to deployment: dataset shift, causality, and shift-stable models in health AI. *Biostatistics* **2020**, *21*, 345–352. [CrossRef]
66. Rojas-Carulla, M.; Schölkopf, B.; Turner, R.; Peters, J. Invariant models for causal transfer learning. *J. Mach. Learn. Res.* **2018**, *19*, 1309–1342.
67. Rothenhäusler, D.; Meinshausen, N.; Bühlmann, P.; Peters, J. Anchor regression: Heterogeneous data meet causality. *J. R. Stat. Soc. Ser. B* **2021**, *83*, 215–246. [CrossRef]
68. Gilmer, J.; Adams, R.P.; Goodfellow, I.; Andersen, D.; Dahl, G.E. Motivating the Rules of the Game for Adversarial Example Research. *arXiv* **2018**, arXiv:1807.06732.
69. Zhao, Z.; Dua, D.; Singh, S. Generating natural adversarial examples *arXiv* **2017**, arXiv:1710.11342.
70. Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; Mukhopadhyay, D. Adversarial attacks and defences: A survey. *arXiv* **2018**, arXiv:1810.00069
71. Hitaj, B.; Ateniese, G.; Perez-Cruz, F. Deep models under the GAN: information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 603–618.
72. Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; Mcdaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv* **2017**, arXiv:1705.07204.
73. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2013**, arXiv:1312.6199.
74. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv* **2017**, arXiv:1706.06083.
75. Papernot, N.; Mcdaniel, P.; Goodfellow, I. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv* **2016**, arXiv:1605.07277.
76. Pang, R.; Zhang, X.; Ji, S.; Luo, X.; Wang, T. AdvMind: Inferring Adversary Intent of Black-Box Attacks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, 6–10 July 2020; pp. 1899–1907.
77. Vivek, B.; Mopuri, K.R.; Babu, R.V. Gray-box adversarial training. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 203–218.
78. Fenrich, K. Securing your control system. *Power Eng.* **2008**, *112*, 1–11.

79. Ilmoi. Poisoning attacks on Machine Learning: A 15-year old security problem that's making a comeback. *Secur. Mach. Learn.* **2019**. Available online: https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db (accessed on 8 March 2023)
80. Rubinstein, B.I.; Bartlett, P.L.; Huang, L.; Taft, N. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *J. Priv. Confidentiality* **2012**, *4*, 65–100. [CrossRef]
81. Steinhardt, J.; Koh, P.W.; Liang, P. Certified defenses for data poisoning attacks. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 3520–3532.
82. Mei, S.; Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 2871–2877.
83. Kloft, M.; Laskov, P. Online anomaly detection under adversarial impact. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 405–412.
84. Koh, P.W.; Steinhardt, J.; Liang, P. Stronger data poisoning attacks break data sanitization defenses. *Mach. Learn.* **2022**, *111*, 1–47. [CrossRef]
85. Shafahi, A.; Huang, W.R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on Neural Networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, Canada, 3–8 December 2018; pp. 6106–6116.
86. Suciu, O.; Marginean, R.; Kaya, Y.; Daume, H.; Iii.; Dumitras, T. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In Proceedings of the 27th Security Symposium, USENIX, Baltimore, MD, USA, 15–17 August 2018; pp. 1299–1316.
87. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2014**, arXiv:1412.6572.
88. Lyu, C.; Huang, K.; Liang, H.N. A unified gradient regularization family for adversarial examples. In Proceedings of the 2015 IEEE international conference on data mining, Atlantic City, NJ, USA, 14–17 November 2015; pp. 301–309.
89. Papernot, N.; Mcdaniel, P. Extending defensive distillation. *arXiv* **2017**, arXiv:1705.05264.
90. Papernot, N.; Mcdaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z.B.; Swami, A. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia conference on computer and communications security, Abu Dhabi, United Arab Emirates, 2–6 April 2017; pp. 506–519.
91. Xu, W.; Evans, D.; Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv* **2017**, arXiv:1704.01155.
92. Hosseini, H.; Chen, Y.; Kannan, S.; Zhang, B.; Poovendran, R. Blocking transferability of adversarial examples in black-box learning systems. *arXiv* **2017**, arXiv:1703.04318.
93. Meng, D.; Chen, H. Magnet: A two-pronged defense against adversarial examples. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 135–147.
94. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
95. Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv* **2018**, arXiv:1805.06605.
96. Weerasinghe, S.; Alpcan, T.; Erfani, S.M.; Leckie, C. Defending Distributed Classifiers Against Data Poisoning Attacks. *arXiv* **2020**, arXiv:2008.09284.
97. Efron, B. The jackknife, the bootstrap and other resampling plans. In *CBMS-NSF Regional Conference Series in Applied Mathematics*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1982.
98. Paudice, A.; Muñoz-González, L.; Lupu, E.C. Label sanitization against label flipping poisoning attacks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 5–15.
99. Paudice, A.; Muñoz-González, L.; Gyorgy, A.; Lupu, E.C. Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv* **2018**, arXiv:1802.03041.
100. Rubinstein, B.I.; Nelson, B.; Huang, L.; Joseph, A.D.; Lau, S.; Rao, S.; Taft, N.; Tygar, J.D. Antidote: Understanding and defending against poisoning of anomaly detectors. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, Chicago, IL, USA, 4–6 November 2009; pp. 1–14.
101. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
102. Koh, P.W.; Liang, P. Understanding black-box predictions via influence functions. In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 1885–1894.
103. Liubchenko, N.; Podorozhniak, A.; Oliinyk, V. Research Application of the Spam Filtering and Spammer Detection Algorithms on Social Media. *CEUR Workshop Proc.* **2022,** *3171*, 116–126.
104. Wang, Q.; Yuying, G.; Ren, J.; B., Z. An automatic classification algorithm for software vulnerability based on weighted word vector and fusion neural network. *Comput. Secur.* **2023**, *126*, 103070. [CrossRef]
105. Peri, N.; Gupta, N.; Huang, W.R.; Fowl, L.; Zhu, C.; Feizi, S.; Goldstein, T.; Dickerson, J.P. Deep k-NN defense against clean-label data poisoning attacks. In Proceedings of the European Conference on Computer, Glasgow, UK, 23–28 August 2020; pp. 55–70.
106. Natarajan, J. AI and Big Data's Potential for Disruptive Innovation. Cyber secure man-in-the-middle attack intrusion detection using machine learning algorithms. In *AI and Big Data's Potential for Disruptive Innovation*; IGI Global: Hershey, PA, USA, 2020; pp. 291–316.

107. Tran, B.; Li, J.; Madry, A. Spectral Signatures in Backdoor Attacks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, Canada, 3–8 December 2018; pp. 8011–8021
108. Nguyen, G.; Dlugolinsky, S. ; Bobak, M.; Tran, V.; Garcia, A.; Heredia, I.; Malik, P.; Hluchy, L. Machine Learning and Deep Learning frameworks and libraries for large-scale. *Artif. Intell. Rev.* **2019**, *52*, 77–124. [CrossRef]
109. Wu, B.; Wang, S.; Yuan, X.; Wang, C.; Rudolph, C.; Yang, X. Defending Against Misclassification Attacks in Transfer Learning. *ArXiv*, **2019**, arXiv:1908.11230
110. Polyak, A.; Wolf, L. Channel-level acceleration of deep face representations. *IEEE Access* **2015**, *3*, 2163–2175. [CrossRef]
111. Blanchard, P.; Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *31st Conf. Neural Inf. Process. Syst.* **2017**, *30*, 118–128.
112. Chen, Y.; Su, L.; Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. Acm Meas. Anal. Comput. Syst.* **2017**, *1*, 1–25. [CrossRef]
113. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4768–4777.
114. Guerraoui, R.; Rouault, S. The hidden vulnerability of distributed learning in byzantium. In Proceedings of the International Conference on Machine Learning; Stockholm, Sweden, 10–15 July 2018; pp. 3521–3530.
115. Pillutla, K.; Kakade, S.M.; Harchaoui, Z. Robust aggregation for federated learning. *IEEE Trans. Signal Process.* **2022**, *70*, 1142–1154. [CrossRef]
116. Yin, D.; Chen, Y.; Kannan, R.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5650–5659.
117. Bernstein, J.; Wang, Y.X.; Azizzadenesheli, K.; Anandkumar, A. signSGD: Compressed optimisation for non-convex problems. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 560–569.
118. Fung, C.; Yoon, C.J.; Beschastnikh, I. Mitigating sybils in federated learning poisoning. *arXiv* **2018**, arXiv:1808.04866.
119. Liu, Y.; Yi, Z.; Chen, T. Backdoor attacks and defenses in feature-partitioned collaborative learning. *arXiv* **2020**, arXiv:2007.03608.
120. Ozdayi, M.S.; Kantarcioglu, M.; Gel, Y.R. Defending against Backdoors in Federated Learning with Robust Learning Rate. 2020. Available online: https://ojs.aaai.org/index.php/AAAI/article/view/17118/16925 (accessed on 8 March 2023).
121. Yang, Z.; Zhang, J.; Chang, E.C.; Liang, Z. Neural network inversion in adversarial setting via background knowledge alignment. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 225–240.
122. Fredrikson, M.; Lantz, E.; Jha, S.; Lin, S.; Page, D.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1322–1333.
123. Hidano, S.; Murakai, T.; Katsumata, S.; Kiyomoto, S.; Hanaoka, G. Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017; pp. 115–11509.
124. Wu, X.; Fredrikson, M.; Jha, S.; Naughton, J.F. A methodology for formalizing model-inversion attacks. In Proceedings of the 2016 IEEE 29th Computer Security Foundations Symposium (CSF), Lisbon, Portugal, 27 June–1 July 2016; pp. 355–370.
125. Zhang, Y.; Jia, R.; Pei, H.; Wang, W.; Li, B.; Song, D. The secret revealer: Generative model-inversion attacks against deep neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13 June–19 June 2020; pp. 250–258.
126. Ateniese, G.; Mancini, L.V.; Spognardi, A.; Villani, A.; Vitali, D.; Felici, G. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *Int. J. Secur. Networks* **2015**, *10*, 137–150. [CrossRef]
127. Juuti, M.; Szyller, S.; Marchal, S.; Asokan, N. PRADA: protecting against DNN model stealing attacks. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 17–19 June 2019; pp. 512–527.
128. Wang, B.; Gong, N.Z. Stealing hyperparameters in machine learning. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–23 May 2018; pp. 36–52.
129. Takemura, T.; Yanai, N.; Fujiwara, T. Model Extraction Attacks on Recurrent Neural Networks. *J. Inf. Process.* **2020**, *28*, 1010–1024. [CrossRef]
130. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
131. Hsu, Y.C.; Hua, T.; Chang, S. ; Lou, Q.; Shen, Y.; Jin, H. Language model compression with weighted low-rank factorization, *arXiv* **2022**, arXiv:2207.00112. 10.48550/arXiv.2207.00112, 2022.
132. Chandrasekaran, V.; Chaudhuri, K.; Giacomelli, I.; Jha, S.; Yan, S. Exploring connections between active learning and model extraction. In Proceedings of the 29th Security Symposium (USENIX), Boston, MA, USA, 12–14 August 2020; pp. 1309–1326.
133. Lee, T.; Edwards, B.; Molloy, I.; Su, D. Defending against neural network model stealing attacks using deceptive perturbations. In Proceedings of the 2019 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 20–22 May 2019; pp. 43–49.
134. Kesarwani, M.; Mukhoty, B.; Arya, V.; Mehta, S. Model extraction warning in MLaaS paradigm. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 371–380.
135. Fredrikson, M.; Lantz, E.; Jha, S.; Lin, S.; Page, D.; Ristenpart, T. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. *Proc. Usenix Secur. Symp.* **2014**, *1*, 17–32.

136. Chaabane, A.; Acs, G.; Kaafar, M.A. You are what you like! information leakage through users' interests. In Proceedings of the 19th Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA, USA, 5–8 February 2012.
137. Kosinski, M.; Stillwell, D.; Graepel, T. Private traits and attributes are predictable from digital records of human behavior. *Proc. Natl. Acad. Sci. USA* **2013**, *110*, 5802–5805. [CrossRef] [PubMed]
138. Gong, N.Z.; Talwalkar, A.; Mackey, L.; Huang, L.; Shin, E.C.R.; Stefanov, E.; Shi, E.; Song, D. Joint link prediction and attribute inference using a social-attribute network. *Acm Trans. Intell. Syst. Technol.* **2014**, *5*, 1–20. [CrossRef]
139. Reynolds, N.A. An Empirical Investigation of Privacy via Obfuscation in Social Networks, 2022. Available online: https://figshare.mq.edu.au/articles/thesis/An_empirical_investigation_of_privacy_via_obfuscation_in_social_networks/19434461/1 (accessed on 8 March 2023).