THE UNIVERSITY OF CHICAGO

POISON FORENSICS: TRACEBACK OF DATA POISONING ATTACKS IN NEURAL NETWORKS

A DISSERTATION SUBMITTED TO

THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCE DIVISION

IN CANDIDACY FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY

SIXIONG SHAN

CHICAGO, ILLINOIS

7/22/2022

Dedication Text

Epigraph Text

TABLE OF CONTENTS

ABSTRACT

In adversarial machine learning, new defenses against attacks on deep learning systems are routinely broken soon after their release by more powerful attacks. In this context, *forensic tools* can offer a valuable complement to existing defenses, by tracing back a successful attack to its root cause, and offering a path forward for mitigation to prevent similar attacks in the future.

In this paper, we describe our efforts in developing a forensic traceback tool for poison attacks on deep neural networks. We propose a novel iterative clustering and pruning solution that trims "innocent" training samples, until all that remains is the set of poisoned data responsible for the attack. Our method clusters training samples based on their impact on model parameters, then uses an efficient data unlearning method to prune innocent clusters. We empirically demonstrate the efficacy of our system on three types of dirty-label (backdoor) poison attacks and three types of clean-label poison attacks, across domains of computer vision and malware classification. Our system achieves over 98.4% precision and 96.8% recall across all attacks. We also show that our system is robust against four anti-forensics measures specifically designed to attack it.

CHAPTER 1

INTRODUCTION

For external facing systems in real world settings, few if any security measures can offer full protection against all attacks. In practice, digital forensics and incident response (DFIR) provide a complementary security tool that focuses on using post-attack evidence to trace back a successful attack to its *root cause*. For packet routing on the wide-area Internet, for example, forensic IP traceback tools can identify the true source of a Denial of Service (DoS) attack. Not only can forensic tools help operators identify (and hopefully patch) vulnerabilities responsible for successful attacks, but strong forensics can provide a strong deterrent against future attackers by threatening them with post-attack identification.

Such an approach would be particularly attractive in the context of attacks against deep learning systems, where new defenses are routinely broken soon after their release by more powerful attacks [80, 65, 11, 3, 12]. Consider for example "poisoning attacks," a threat that arises from the reliance of ML trainers and operators on external data sources, either purchasing data from or outsourcing data collection to third parties [55]. An attacker can inject manipulated training data into the training data pipeline, thus causing the resulting model to produce targeted misclassification on specific inputs. Recent advances in poisoning attacks have made them more powerful [1, 48], more realistic [65, 80, 83], and more stealthy [45, 4]. In a recent survey, industrial practitioners ranked data poisoning attacks as the most worrisome threat to industry machine learning systems [42].

For data poisoning attacks, effective forensics would add a valuable complement to existing defenses, by helping to identify *which* training samples led to the misclassification behavior used in the attack. We call this the "poison traceback problem." Starting with evidence of the attack (an input sample that triggers the misclassification), a forensic tool would seek to identify a particular subset of training data responsible for corrupting the model with the observed misclassification behavior. Combined with metadata or logs that track the provenance of training data, this enables practitioners to identify either the source of the poison data, or a vulnerability in the data pipeline where the poison data was inserted. Either result leads to direct mitigation steps (e.g. removing an unreliable data vendor or securing a breached server on the training data pipeline) that would patch the pipeline and improve robustness to similar attacks in the future.

Several factors make the poison traceback problem quite challenging in practice. First, today's deep learning models employ large complex architectures that do not easily lend themselves to explainability. Specific behaviors do not localize themselves to specific neurons as once speculated. Second, the effects of poisoning attacks generally require training on a *group* of poisoned data, and a subset of the poisoned training data is unlikely to produce the same behavior. Thus a brute force search for the subset of poisoned training data would involve testing an exponential number of sample combinations from a large training corpus. Finally, a poison traceback tool produces evidence that can lead to the identification of parties responsible for an attack. Thus these tools must have very high precision, since false positives could lead to false accusations and negative consequences.

In this paper, we introduce the poison traceback problem, and propose the first solution that accurately identifies poisoned training data responsible for an observed attack. Our solution utilizes an iterative clustering and pruning algorithm. At each step, it groups training samples into clusters based on their impact on model parameters, then identifies benign clusters using an efficient data unlearning algorithm. As benign clusters are pruned away, the algorithm converges on a minimal set of training samples responsible for inducing the observed misclassification behavior. In detailed experiments covering a variety of tasks/datasets and attacks, our approach produces highly accurate (high precision and recall) identification of poison data for both dirty-label and clean-label poison attacks.

This paper makes the following contributions to the forensics of poison attacks:

- We define forensics in the context of data poisoning attacks and design a forensics system that effectively traces back misclassification events to poison training data responsible.
- We empirically demonstrate the effectiveness of our system on three types of dirty-label poison

attacks and three types of clean-label poison attacks, across two domains of computer vision and malware classification. Our system achieves over 98.4% precision and 96.8% recall on identifying poison training data,

- We test our system against 4 alternative forensic designs adapted from prior defenses, and show our system consistently succeeds on attacks where alternatives fall short.
- We consider potential anti-forensics techniques that can be used to evade our system. We test our system and show that it is robust against 6 adaptive attacks specifically designed to overcome this forensic system.

To the best of our knowledge, this is the first work to explore a forensics approach to address data poisoning attacks on deep learning systems. This is a significant departure from existing works that focus entirely on attack prevention. Given our initial results, we believe poison traceback is a promising direction worthy of further exploration.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this section, we present the background and related work on data poisoning and digital forensics.

2.0.1 Data Poisoning

In data poisoning attacks, the attacker gains access to the training data pipeline of the victim ML system, *e.g.,* via a malicious data provider, and injects a set of poison data into the training dataset. The poison data causes the victim's model to have certain vulnerabilities, *i.e.,* misclassifying certain inputs targeted by the attacker.

Data Poisoning Attacks. We can divide existing poison attacks into two categories based on their attack assumptions: dirty-label attacks where attacker can modify both the data and their semantic labels, and clean-label attacks where attacker can only modify the data. Dirty-labels attacks [29, 48, 80], often called as backdoor attacks, seek to inject a *trigger* into the victim model. A trigger is a unique input signal (*e.g.,* a yellow sticker on an image, a trigger word in a sentence) that once present can lead the victim model to misclassify any inputs to a target label selected by the attacker (*e.g.,* the presence of yellow sticker leads the model to classify stop signs as speed limits [29]).

Clean-label attacks further divide into clean-label backdoor attacks and clean-label triggerless attacks. Clean-label backdoor attacks [65, 76, 61] are similar to dirty-label backdoor attacks except that attacker cannot modify the label of the poison data. Clean-label triggerless attack aims to misclassify a single unmodified test data. Shafahi *et al.* [66] proposed the first clean-label triggerless attack where an attacker injects poison data to disrupt the feature region of the targeted data. Several proposals [1, 87] significantly improve the performance of clean-label attacks by positioning poison data on a convex polytope around the target data. These clean-label attacks only perform well when the victim model's feature space is known, *i.e.,* assuming the victim uses transfer learning and the attacker has white-box access to the pretrained model's parameters. A recent attack,

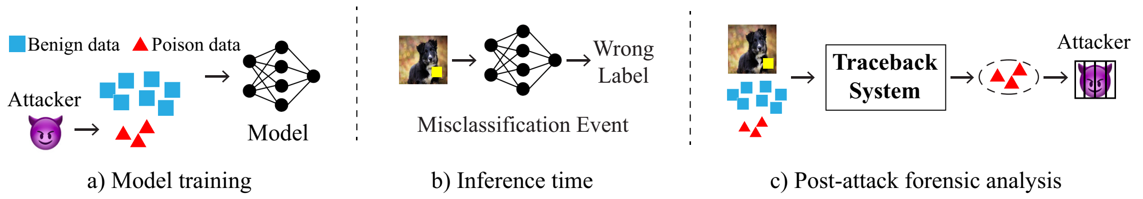


Figure 2.1: The general scenario for our trackback system. a) the attacker poisoned the training data to inject vulnerability into the model; b) at run-time, the attacker submits an attack input to cause a misclassification event; c) our traceback system inspects the misclassification event to identify its root cause.

WitchBrew [26], targets the from-scratch training scenario leveraging gradient alignment of poison and target data.

Data Poisoning Defenses. A large body of research seeks to defend against poison attacks. Robust training defenses modify the training of neural networks to be resilient against data poisoning. Existing robust training defenses leverage ensemble training [36], kNN majority voting [35], adversarial training [27], random smoothing [78], and data augmentation [7]. Other defenses try to diagnose and patch an already poisoned model. Neural Cleanse [79] assumes backdoor triggers are small input signals and reverse engineers the injected backdoor trigger. Fine-Pruning and STRIP assume neurons related to poison are not activated by benign data, and thus remove unused neurons [47, 24]. SPECTRE [31] assumes a Gaussian distribution of benign feature representations, and filters out anomalous inference queries.

Still, defending against poison attacks remains a challenging problem, mainly because the injected vulnerability is hidden and has not been activated at defense time. Thus, existing defenses examine the training data or various behaviors of the model to identify anomalous signals that might be malicious. While existing defenses have shown promising signs by preventing many poison attacks, stronger and adaptive attackers are able to bypass existing defenses [80, 83, 65, 4, 63].

2.0.2 Background on Digital Forensics

First introduced in the 1970s, digital forensics has been a major and growing area of cybersecurity. Digital forensics seeks to trace the *source* of a cyberattack that has already happened leveraging traces that the attacker left in the victim system.

"Attack incidents" trigger the forensic analysis, *i.e.,* when the system administrator discovers a cyberattack after some catastrophic events have happened (*e.g.,* web servers overloaded with dummy requests, machine takeover, or sensitive data appearing in the dark web). Then, a forensics system is assigned to investigate the source of the attack. Forensic analysis often starts with evidence collection from the logs of the victim system. Then the system connects these pieces of evidence using their casual links to form a causal graph, and identifies the root cause of the attack by tracing through the casual graph starting from the attack incident.

Benefits of Forensics. Successful forensics can lead to prosecution of the perpetrator, stopping the attack from the source, and offering insights to build more secure systems. Forensics can even break the arms race between attackers and defenders, since an attacker faces a much higher cost of iterating with a forensics system, *i.e.,* the attacker is held accountable as long as the forensics system succeeds once. Consequently, the risk of being caught acts as a strong deterrent to discourage any attackers from launching the attack in the first place.

Forensics vs. Defenses. Forensics is a *complementary* approach to defenses (or security through prevention). While there is a significant amount of prior works focusing on defenses against adversarial attacks, history (in both machine learning security and multiple other security areas) shows that no defense is perfect in practice, and attackers will find ways to circumvent even strong defenses. Forensics addresses the incident response of successful attacks by tracing back to the root causes. Modern security systems leverage both defenses and forensics to achieve maximum security.

The same dynamic holds true in the context of poison attacks on neural networks. For example, a defense against backdoors that identifies poison training data can be circumvented by an attacker who breaches the server after the defense has been applied, but prior to model training.

Existing Digital Forensics Research. Forensics has been widely studied in security community to solve a wide variety of security problems, *e.g.,* tracing the source IP of DDoS attacks [62, 17, 72], origins of intrusion [39, 84], and the cause of advance persistent threats (APTs) [82, 85, 23]. Existing research addresses many technical challenges of forensics. [6, 38, 21, 43] seek to secure the integrity of traces left by the attacker against potential tampering. [81, 44, 50, 19] reduce the large storage overhead of logging while preserving enough information. [85, 51, 39] address the dependency-explosion problem where a forensics system cannot narrow down the true cause of the attack. Another line of research focuses on post-forensics, *i.e.,* after the root cause is identified. The post-forensics system can prosecute the attacker in court by generating causal proof [25, 20] and fingerprint the attack to prevent similar attacks in the future [54, 37, 58].

CHAPTER 3

TRACEBACK ON DATA POISONING ATTACKS

In this paper, we consider the task of applying forensics to uncover the presence of data poisoning attacks on deep neural network (DNN) models. Given a misclassification event at test time, we seek to identify the set of poisoned training data that resulted in the misclassification.

Example Scenario. Figure 2.1 illustrates the general scenario for post-attack forensic analysis. One or more attackers find a way to access the training data pipeline[1], and inject poison training data to introduce a specific vulnerability into the DNN model (Figure 2.1(a)). Once the corrupted model is deployed, the attacker submits a carefully crafted input that exploits the vulnerability to produce a misclassified result. When the administrator discovers this misclassification event (possibly after downstream events), they want to identify the root cause or entity responsible (Figure 2.1(b)). Information is sent to the traceback system, including the input that caused the misclassification, the DNN model, and its training data. The traceback system then identifies the poison training data responsible for the misclassification event (Figure 2.1(c)).

Here, we define our threat model, identify key goals and challenges of a forensic traceback system for poisoning attacks, and highlight our key intuition for our solution. We describe the details of our traceback system in §4.

Terminology. We use the following terminology:

- **Data poisoning attack**: the injection of poisoned training data that embeds vulnerability into the victim model.
- **Misclassification event**: an input (x_a) that the model misclassified, and the corresponding misclassified label (y_a).

3.0.1 Threat Model

We first describe our threat model and assumptions of the attacker and the traceback system.

1. The training data pipeline often includes multiple layers of data collectors, labelers, and brokers.

Data Poisoning Attacker. We adopt common assumptions made by existing work on poisoning attacks and defenses. We assume the attacker:

- can modify any portion of their controlled training data;
- can poison at most half of the entire training dataset;
- has no access to other parts of the model training pipeline, including the final model parameters of the trained model[2];
- is aware of the existence of a potential traceback system and can adopt anti-forensics techniques to evade traceback (more details in §7);
- at inference time, submits an attack input that utilizes the injected vulnerability to cause model misclassification.

Traceback System. We assume the traceback system is deployed by the model owner or a trusted third party, and thus has full access to the following resources:

- the DNN model (its parameters and architecture), the model training pipeline and the training data;
- information on the misclassification event, *i.e.,* the exact attack input x_a and misclassification output y_a.

We do not assume the traceback system has any access to information on other attacks (beyond the current misclassification event), and make no assumption about types or parameters of the poisoning attacks. Note that the traceback system has full access to the training dataset, unlike assumptions made by some existing defenses, which prevent poison attacks without leveraging the full training dataset [79].

We note that in practice, a misclassification event may also arise from low model accuracy or from an evasion attack. Either can cause misclassification without poisoning/modifying training data. In §8, we discuss how our forensic tool can also be used to determine if a misclassification

2. There exists a few parameter-space poison attack [33, 70] and we consider them outside of our threat model since these attacks require additional access to the victim's training pipeline.

was caused by a poisoning attack. A comprehensive study of robust recognition of non-poison misclassification events is beyond the scope of this paper. In the rest of the paper, we only limit ourselves to data poisoning attacks.

3.0.2 Design Requirements and Challenges

To identify what/who is responsible for the misclassification event, a practical DNN traceback system should meet the following requirements:

- **High precision** – In forensics, false positives can lead to false accusations, and thus must be minimized. Under our problem context, this means that for any misclassification event caused by a specific poisoning attack $\mathbb{A}$, traceback should identify only those poisoned training data injected to implement $\mathbb{A}$ but not others.
- **High recall** – Recall measures the percentage of poison training data responsible for the misclassification event that are identified by the traceback system. Achieving a high recall rate is crucial for identifying all the attack parties, especially when multiple parties worked together to inject poison data in order to train a vulnerability into the model.
- **Generalizability** – An effective traceback system should address a wide range of poisoning attacks against DNN models, without requiring knowledge of the attack type or parameters (*e.g.,* the amount of poison training data).

We further note two *non-goals* of our system. The first is *attack scope*. The goal of the traceback system is to respond to a specific, observed attack. In a scenario where one or more attackers have performed multiple, independent poisoning attacks on the same model, the traceback system focuses on identifying the poison data that caused the observed misclassification event. The second is on *computational latency*. Unlike real-time attack detection tools, forensic traceback is a post-attack operation and does not face strict latency requirements. This is a common assumption for digital forensics [14, 28].

Potential Solutions and Key Challenges. Traditional digital forensics traces the root cause of an attack by building causal graphs using causal links among system events [85, 51, 39]. In our problem context, DNN model training allows each individual training sample to potentially contribute to the final model parameters, and by extension, the misclassification result. This is commonly known in forensics as the *dependency explosion* problem, and combined with the large size of training data (*e.g.,* millions of training samples) makes conventional causal graph analysis intractable.

The key challenge facing any DNN forensic system is how to efficiently connect a (mis)classification result to specific samples in the training data. For non-DNN, linear ML models, existing works use the well-known *influence function* [40] to estimate the contribution of each training data point towards a classification result, leveraging the first-order Taylor's approximation. However, recent work [5] showed that when applied to DNNs, the influence function produces poor performance and requires costly computation of second-order derivatives. We confirmed these observations experimentally. Using the influence function to traceback BadNet poisoning attacks on a CIFAR10 model (details in §5.1) achieved less than 69% precision and recall. When testing it on models trained on the larger ImageNet, our influence function computation timed out after running 15 days on 4 NVidia TitanX GPUs.

Another alternative is to adapt existing poison defenses into forensic tools for use after an attack has been detected. While adapting defense techniques as forensics is itself slightly paradoxical (waterproof defenses would obviate the very need for forensics), we can nonetheless test to see if such techniques can be effective after an attack. Later in §6.1, we adapt four defenses (Spectral Signature, Neural Cleanse, Deep K-NN, and L_2-Norm) into potential forensic tools, and compare them with our traceback system. Some of the adapted systems have success against simple attacks, but all of them fail on stronger poison attacks.

Poisoning as a group effect. In current work on poisoning attacks, attack success relies on a critical mass of poison samples in the training set [29, 65, 69]. While a sufficiently large set of poison training data can shape model behavior and inject vulnerabilities, the contribution of each individual data sample is less and much harder to quantify. This explains the poor performance of the influence function when applied to smaller models such as CIFAR10 (see above). It motivates us to design a solution to search for groups of poison training data, not single samples.

3.0.3 Design Intuition

Instead of designing the traceback system to explicitly target individual training data samples, our intuition is to inspect training data in groups, and map the traceback problem to a *set searching* problem.

Set searching by iterative clustering and pruning. We propose to search for sets of training samples responsible for an observed misclassification event, by iteratively pruning groups of training data we can identify as *innocent* to the attack. Starting with the full training data set, we progressively identify and prune clusters of innocent training samples until only those *responsible* for the misclassification event are left. In each iteration, we only need to identify clusters of training data that *do not* contain any poison training data required to make the misclassification event successful. As such, our traceback design only needs a "binary" measure of event responsibility, which is much easier to compute than the actual contribution of any training data samples to the attack.

A binary measure of event responsibility. We propose a binary measure of event responsibility, which connects a misclassification event with the model training data. Here our hypothesis is that since data poisoning attacks focus on making the model learn new behavior that is different from those offered by the benign (or innocent) data, the attack confidence level should not degrade if some portion of the innocent data is *not* used for model training. In this work, we propose to use this condition to determine whether a cluster of training data contains only innocent data not responsible for the misclassification.

We formally define this condition as follows. Let the model's full training dataset D be divided into two distinct subsets: D_1 and $D \setminus D_1$. Let $\mathcal{F}$ be the DNN model trained on D and $\mathcal{F}^-$ be the model trained on $D \setminus D_1$. Let (x_a, y_a) represent the misclassification event. We use $\ell(\mathcal{F}(x_a), y_a)$ and $\ell(\mathcal{F}^-(x_a), y_a)$ to indirectly compare the confidence level of (x_a, y_a) on the two DNN models, where $\ell(.)$ is the cross-entropy loss function. Specifically, if removing D_1 from the model training data does not decrease the attack confidence level, *i.e.,*

$$\ell(\mathcal{F}(x_a), y_a) \leq \ell(\mathcal{F}^-(x_a), y_a), \tag{3.1}$$

then D_1 is *less responsible* for the misclassification event (x_a, y_a) than $D \setminus D_1$. This is in the sense that $D \setminus D_1$ has a ratio of benign to poison data that is more skewed towards poison than D, allowing us to use this measure of event responsibility to iteratively determine the subset of poison data. We note that in practice, we are able to use clustering to find splits such that D_1 does not contain *any* poison data. Our proposed measure *only* examines the attack confidence level, and does not consider the model's normal classification accuracy.

The proposed binary measure of event responsibility (Eq. 3.1) is supported by our theoretical analysis on how removing a portion of the training data affects attack performance. For brevity, we present the analysis in Appendix A.0.1.

CHAPTER 4
DETAILED POISON TRACEBACK DESIGN

This section presents the detailed design of our traceback system. We start from a high-level overview, followed by the detailed description of the two key components (clustering and pruning), which run iteratively to identify the set of poison training data responsible for the misclassification event.

4.0.1 High-level Overview

In a nutshell, our traceback system implements an iterative clustering and pruning process, which progressively identifies sets of innocent training data that are not responsible for the observed misclassification event. This ends when only the poison data responsible for the misclassification event is left and thus identified. Doing so requires two key operations: clustering and pruning.

(1) Clustering unmarked training data. The clustering component divides the unmarked training data into clusters, based on how they affect the model parameters (details in §4.0.2). Here the goal is to progressively separate innocent data from poison data, so that we can identify, mark and prune an innocent cluster (using the pruning component).

(2) Identifying and pruning innocent clusters. This pruning component examines the unmarked data clusters, applies the binary metric defined by Eq. (3.1) to identify an *innocent* cluster, if any, that is not responsible for the misclassification event. The identified cluster is marked and thus excluded from the next clustering operation, *i.e.,* pruned out. We note that pruning does not affect the computation of Eq. (3.1), where D is always the original full training dataset and D_1 is a cluster to be examined. The detailed pruning design is in §4.0.3.

An illustrative example. Figure 4.1 shows an example traceback process that completes in two iterations, visualized in a simplified 2D projection of the training data. The traceback starts from the full set of training data as unmarked, including both innocent (blue) and poison (red) data. In each iteration, the left figure shows the collection of unmarked training data to be clustered and the resulting cluster boundary that divides them into two clusters; the right figure shows the result of pruning where the innocent cluster is removed. At the end of iteration 2, only the set of poison data responsible for the attack is left as unmarked. Upon detecting that the unmarked data cannot be further divided or pruned, the process ends and the unmarked data are declared as the training data responsible for the misclassification event.

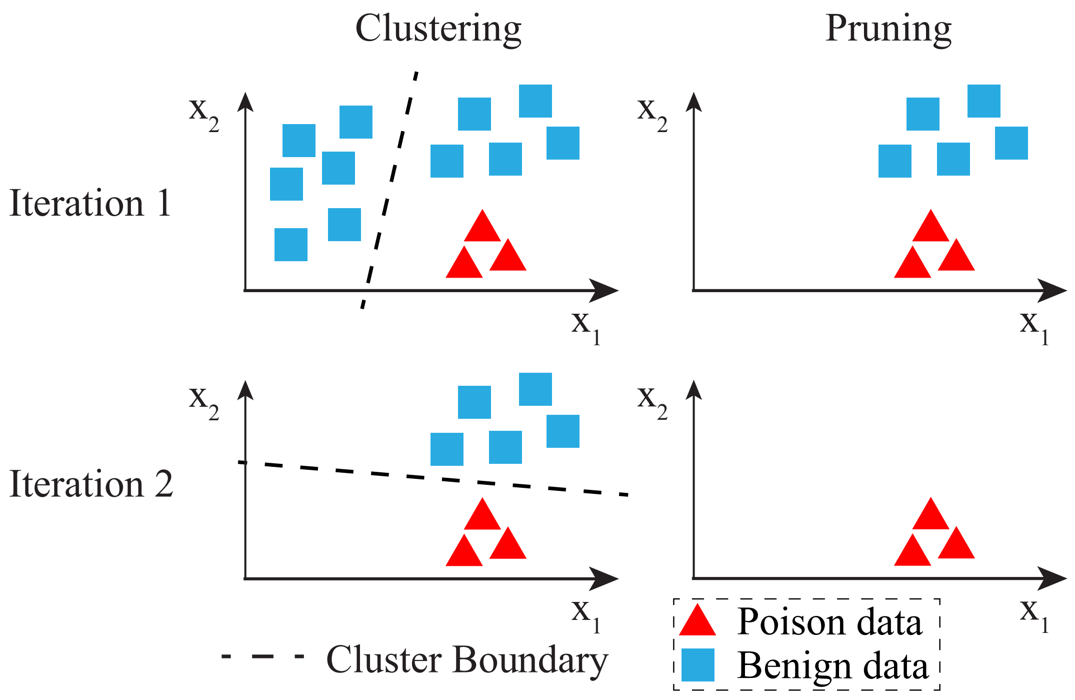


Figure 4.1: An illustration of our poison traceback process that completes in two iterations, visualized on a simplified 2D space representing the training data.

4.0.2 Clustering Unmarked Training Data

We now describe the detailed clustering design, which seeks to progressively separate benign and poison training data into different clusters. The key challenge here is that while innocent and poison data are different by design (in order to inject different behaviors into the model), it is highly challenging to accurately characterize and measure such differences. This is also why it has been hard to design defense mechanisms that can effectively identify and remove poison training data.

Instead, our clustering design first maps these training data into a new space, focusing on "amplifying" the separation between innocent and poison data rather than identifying them. Operating on this new space, each clustering operation will produce two clusters, ideally one containing only innocent data and the other containing a mixture of innocent and poison data. Given these two clusters, our pruning component (§4.0.3) will replay the misclassification event to identify the innocent cluster. In the next iteration, we will run clustering only on the mixed cluster to "extract" more innocent data. After a few iterations, the innocent and poison data become fully separated.

Therefore, our clustering design includes 1) data mapping to "amplify" the distance between innocent and poison data, and 2) a high performance clustering method to generate the clusters, which we discuss below.

Data mapping. We map the data by estimating how a training sample x affects the final model parameters. This is measured by the change of model parameters when x is absent from the training dataset, *i.e.,* comparing the final model parameters when trained on D and $D \backslash x$, where D is the full training dataset. Unlearning benign or poison data results in *different* effect on model parameters. Unlearning of poison samples shifts the model closer to an optimal location in parameter space, where poisoning is ineffective, while unlearning benign samples shifts the model towards its initial randomly initialized state, since if all benign samples are effectively unlearned, the model will have no predictive power. A naive implementation would retrain the model on $D \setminus x$, leading to unnecessary computational overhead and stochasticity from training. Instead, inspired by the concept of *unlearning*, we propose estimating the parameter change using a gradient computation. The gradient of the parameters with respect to a given data point with a specified loss function is a well-known method to characterize its impact on the model [40]. Intuitively, data with similar gradients will have a similar impact on the model. Thus, our data mapping for training data point x is:

$$\nabla_\theta \, \ell(\mathcal{F}(x), NULL) \tag{4.1}$$

where $\mathcal{F}$ is the original model (trained on D), θ is the parameter set of $\mathcal{F}$'s classification layer, ℓ is the cross-entropy loss, and $NULL$ is a new "no knowledge learned" label to represent the effect of not learning from x. We implement $NULL$ as an equal probability output, which has been used by existing works to label out-of-distribution (OOD) samples [77].

We note that our data mapping method is one of the many ways to represent data for analysis in poison setting. Other mapping methods exist in the literature [15, 10]. We leave a systematic study of the optimaility of data mappings and designing better performing mapping to future work.

Clustering heuristics. To handle large training data sets, we use Mini-Batch K-means [64], a scalable variant of the K-means clustering algorithm. As the name suggests, it first runs K-means on multiple smaller batches of the dataset and then aggregates their results. This allows us to distribute the computation across multiple servers, achieving orders of magnitude speedup without degrading the clustering quality [64]. As discussed earlier, we configure the clustering system to produce two clusters per iteration.

4.0.3 Pruning the Innocent Cluster

Given the two clusters, the pruning component will identify which of the two clusters is less responsible, if any, for the misclassification event (x_a, y_a). This is done by using (x_a, y_a) to evaluate each cluster's responsibility to the event – if a cluster meets the condition in Eq. (3.1), it is marked as innocent and excluded in the next clustering iteration. In practice, we find that the less responsible cluster always contains only benign data, due to the clear separation induced by our mapping in §4.0.2, resulting in rapid identification of the poison data.

The design challenge facing this component is how to efficiently evaluate the condition defined by Eq. (3.1), especially $\ell(\mathcal{F}^{-}(x_a), y_a)$. Given a cluster ($D_1$), $\mathcal{F}^{-}$ refers to the DNN model trained on $D \setminus D_1$. One can certainly compute $\ell(\mathcal{F}^{-}(x_a), y_a)$ by training $\mathcal{F}^{-}$ from scratch, which is often expensive in practice. Again, inspired by the concept of 'unlearning', we propose producing $\mathcal{F}^{-}$ by unlearning D_1 from the original model $\mathcal{F}$.

Existing unlearning for provable privacy protection. Unlearning has been explored in the context of privacy (*e.g.,* [8, 30, 53]) where a person protected by privacy laws can request their data be removed from a trained ML model. Thus, existing unlearning methods focus on achieving a strong, provable privacy guarantee at the cost of high computation complexity. Furthermore, their performance degrades rapidly as the number of data points to be unlearned increases [8, 30], making them unsuitable for our traceback system.

Proposed: functional unlearning for traceback. Instead, we propose approximating $\mathcal{F}^-$ by *functionally unlearning* a cluster D_1 from the original model $\mathcal{F}$. Specifically, we fine-tune $\mathcal{F}$ to minimize the cross-entropy loss between D_1 and the NULL label (*i.e.,* no knowledge learned) discussed in §4.0.2 while maintaining a low cross-entropy loss on the rest of the training data ($D \setminus D_1$) like the original model. This fine-tuning operation is guided by

$$\min_{\boldsymbol{\theta}} \left(\sum_{(x,y)\in D_1} \ell(\mathcal{F}(x), NULL) + \sum_{(x,y)\in D\setminus D_1} \ell(\mathcal{F}(x), y) \right) \tag{4.2}$$

where $(x, y) \in D \setminus D_1$ represents the training data instance (input x and its label y). We solve the above optimization using stochastic gradient descent (SGD) with the same hyperparameters as the original model training, and use the fine-tuned version of $\mathcal{F}$ as $\mathcal{F}^-$. We then compute $\ell(\mathcal{F}^-(x_a), y_a)$ using the misclassification event (x_a, y_a) and verify the condition defined by Eq. (3.1).

CHAPTER 5
OVERVIEW OF EVALUATION

Using a variety of tasks/datasets, we evaluate our traceback system against 6 different poison attacks (3 dirty-label and 3 clean-label attacks) and 4 anti-forensic countermeasures. We outline these experiments and a preview of our findings.

I. Traceback of dirty-label poison attacks (§5.1). Using 4 image classification datasets, we test against 3 state-of-the-art dirty-label attacks, including an attack without any known effective defense. Our traceback system achieves $\geq 98.9\%$ precision and $\geq 97.1\%$ recall in identifying poison data.

II. Traceback of clean-label poison attack (§6). For both image classification and malware classification datasets, we test against 3 state-of-the-art clean-label attacks, including an attack without any known effective defense. Our traceback system achieves $\geq 98.4\%$ precision and $\geq 96.8\%$ recall.

III. Robustness against anti-forensic countermeasures (§7). We consider 4 potential countermeasures that a resourceful attacker can deploy to bypass the traceback. Results show that our system is robust against all four. Across all the experiments, the most effective countermeasure reduces the traceback precision and recall by less than 4%.

5.1 Evaluation on Dirty-label Attacks

Our evaluation of the traceback system starts from testing it against dirty-label poisoning attacks. We consider three state-of-the-art dirty-label attacks: BadNet [29], Trojan [48], and Physical Backdoor [80]. We follow the original papers to implement these attacks and vary their attack parameters to produce a rich collection of poisoning attacks and their misclassification events. Overall, our results show that the proposed traceback system can accurately identify the root cause of dirty-label attacks ($\geq 98.9\%$ precision and $\geq 97.1\%$ recall) while maintaining a reasonable traceback time per attack.

Dataset	**Dimensionality**	**# Classes**	**# Training Data**	**Architecture**
CIFAR10 [41]	32×32	10	$50,000$	WideResNet-28 [86]
ImageNet [18]	299×299	$1,000$	$1,281,167$	Inception ResNet [73]
VGGFace [56]	224×224	$2,622$	$2,622,000$	VGG-16 [71]
Wenger Face [80]	224×224	10	762	ResNet-50 [32]
EMBER Malware [2]	2351	2	$600,000$	EmberNN [65]

Table 5.1: Datasets & DNN architectures for our evaluation.

Attack Type	**Attack Name**	**Dataset**	**Traceback Performance**		
			Precision	**Recall**	**Runtime (mins)**
Dirty-label (§5.1)	BadNet	CIFAR10	$99.5 \pm 0.0\%$	$98.9 \pm 0.0\%$	11.2 ± 0.4
	BadNet	ImageNet	$99.1 \pm 0.0\%$	$99.1 \pm 0.0\%$	142.5 ± 4.1
	Trojan	VGGFace	$99.8 \pm 0.0\%$	$99.9 \pm 0.0\%$	208.9 ± 9.2
	Physical Backdoor	Wenger Face	$99.5 \pm 0.1\%$	$97.1 \pm 0.2\%$	2.1 ± 0.0
Clean-label (§6)	BP	CIFAR10	$98.4 \pm 0.1\%$	$96.8 \pm 0.2\%$	19.2 ± 1.2
	BP	ImageNet	$99.3 \pm 0.0\%$	$97.4 \pm 0.1\%$	202.0 ± 7.1
	WitchBrew	CIFAR10	$99.7 \pm 0.0\%$	$96.8 \pm 0.1\%$	21.4 ± 2.1
	WitchBrew	ImageNet	$99.1 \pm 0.1\%$	$97.9 \pm 0.1\%$	194.3 ± 5.9
	Malware Backdoor	Ember Malware	$99.2 \pm 0.0\%$	$98.2 \pm 0.1\%$	57.7 ± 3.0

Table 5.2: Precision, recall, and runtime of the traceback system for each of the four **dirty-label poisoning** attack tasks and the five **clean-label poisoning** attack tasks (averaged over 1000 runs per attack task).

5.1.1 Experiment Setup

We first summarize the configuration of the attacks and our traceback system, and discuss the evaluation metrics.

Attack Setup. Using 4 image classification datasets listed in Table 5.1, we implement the above mentioned three attacks (Table A.4 in Appendix). Their attack success rate and normal classification accuracy match those reported by the original papers. We briefly summarize them below. Further details on the DNN models, datasets, and attacks can be found in Appendix A.0.2.

- **BadNet (CIFAR10, ImageNet):** The BadNet [29] attack builds poison training data by adding a pre-selected backdoor trigger to benign inputs and labeling them with the target label. We run BadNets on image classification tasks trained on CIFAR10 and ImageNet, respectively. The default attack configuration is identical to [29]: 10% injection rate (*i.e.,* 10% of the training data is poison data) and a 'yellow square' as the trigger.
- **Trojan (VGGFace):** The Trojan [48] attack improves upon BadNet by using an optimized trigger to increase attack success. Like the original paper [48], we implement this attack on a face recognition model trained on VGGFace. The default attack configuration uses 10% injection rate and a 59×59 pixel trigger.

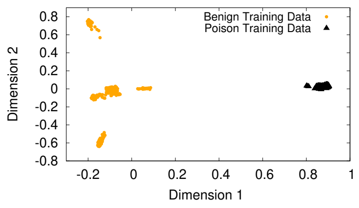
- **Physical Backdoor (WengerFace):** Wenger *et al.* [80] recently proposed a physical backdoor attack against facial recognition models, using everyday physical objects such as eyeglasses and headbands as the trigger. They collected a custom dataset of face images (hereby referred to as WengerFace) of users wearing these accessories. We use the same dataset[1] to implement the attack. Following the original paper, we implement the attack with the default configuration of 10% injection rate and a pair of eyeglasses as the trigger (since it offers the highest success rate among the triggers tested). Note that this backdoor attack is able to bypass 4 state-of-the-art defenses [79, 75, 15, 24]. To the best of our knowledge, there is no known effective defense against this attack.

In the rest of the paper, we often use attack-dataset (*e.g.,* BadNet-CIFAR10) to succinctly identify each attack task. Given an attack configuration (*i.e.,* attack-dataset, injection rate, trigger), we generate 1000 successful attack instances (x_a, y_a) as the misclassification events to test our traceback system. Specifically, we randomly choose 10 target labels to implement 10 versions of the given poisoning attack. Then for each attack version, we randomly select 100 successful attack instances as the misclassification events, producing a total of $10 \times 100 = 1000$ events.

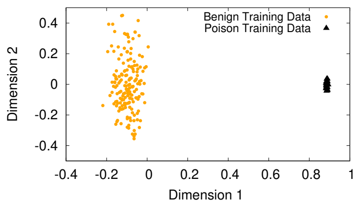
Traceback Setup. Configuring the traceback system is simple as it does not assume prior knowledge of the attacks. The majority of computation comes from the data projection used by the clustering component, *i.e.,* computing eq. (4.1) for each training sample. For large models (like those trained on ImageNet), we speed up the computation of Eq. (4.1) by randomly selecting $\rho\%$ of the model weights in the final classification layer. We empirically find that reducing ρ from 10 to 1 does not lead to visible changes to the traceback accuracy (see Table A.1 in the Appendix), and thus set $\rho = 1$.

Evaluation Metrics. We evaluate the proposed traceback system using three metrics: 1) *precision* of identifying poison training data, 2) *recall* of identifying poison training data, and 3) runtime latency of a successful traceback. We report the average and standard deviation values over 1000

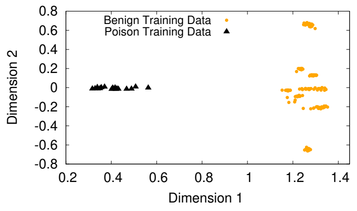
1. We contacted the authors of [80] to obtain the dataset and the required user consent and authorization to use this dataset for our study.



(a) BadNet-CIFAR10



(b) Trojan-VGGFace



(c) Physical-Wenger

Figure 5.1: A simplified, 2-D PCA visualization of the projected training data, where poison and benign data are well-separated.

misclassification events per attack configuration.

5.1.2 Traceback Performance

Precision and Recall. We first report the precision and recall of our traceback system, when tested against dirty-label attacks under the default attack configuration (*i.e.,* those used by the original papers). The top section of Table 5.2 shows the traceback precision and recall for each attack-dataset. Across all these experiments, our system consistently achieves a high precision ($99.1 - 99.8\%$) and a high recall ($97.1 - 99.9\%$).

An interesting observation is that the recall for Physical Backdoor is lower (97.1%) than the other attacks ($> 98.9\%$). That is, our traceback detects less portion of the poison training data samples used by Physical Backdoor compared to the other attacks. We wonder whether this is because those data samples contributed very little to the injected vulnerability, especially since real photos of physical objects often lead to less precise triggers than those injected digitally. We validate this hypothesis by removing the exact set of poison data missed by our traceback and relaunching the poison attack, and the attack success rate drops by only 0.08% on average. But when removing a random set of poison training data of the same size, the attack success rate drops by 1.02% on average. This confirms our hypothesis.

Detailed Analysis of Clustering. To obtain a deeper understanding of the traceback performance, we perform a detailed analysis of the clustering component. Intuitively, clustering is most effective if the data projection makes the benign and poison data well-separated from each other. Along this line, our analysis starts from visualizing the project result of the model training data (benign and poison), for three poisoning attacks (BadNet-CIFAR10, Trojan-VGGFace, and Physical-Wenger). Figure 5.1 shows the simplified 2D version generated using 2-dimensional Principal Component Analysis (PCA) [57] on the projected data. In this visualization, the benign and poison data appear to be well-separated.

Next, we study the L_2 distance among the projected training data, focusing on measuring the normalized L_2 distance between each poison data sample to the centroid of all the benign data (*i.e.,* the benign centroid), and the centroid of all the poison data (*i.e.,* the poison centroid). Table A.6 lists the average results for four attacks. We calculate the normalized L_2 distances to allow a fair comparison across attacks.

Results in Table A.6 confirm that the poison and benign data are reasonably separated. The poison data in Physical-Wenger are more spread out while those in Trojan-VGGFace are densely packed. These observations align with those of Figure 5.1). Overall, our proposed data projection achieves sufficient separation between the benign and poison data, allowing the subsequent clustering and pruning operation to quickly identify all the benign data. Across all four attacks, the traceback takes no more than 4 clustering/pruning iterations to complete.

Detailed Analysis of Pruning. Our pruning operation is based on the condition defined by eq. (3.1) that compares the cross-entropy loss of the misclassification event on the original model $\mathcal{F}$ and the new model $\mathcal{F}^-$ after removing a cluster D_1 from the training dataset. Using BadNet-CIFAR10 as an example, Table 5.3 lists the mean and standard deviation of $\ell(\mathcal{F}(x_a), y_a)$ for the original model, $\ell(\mathcal{F}^-(x_a), y_a)$ when removing an "innocent" cluster, and $\ell(\mathcal{F}^-(x_a), y_a)$ when removing a poison cluster. The latter two display distinct difference when compared to the first term ($\ell(\mathcal{F}(x_a), y_a)$), confirming that our proposed binary condition offers a clear signal to accurately identify innocent clusters not responsible for the misclassification.

$\ell(\mathcal{F}(x_a), y_a)$	$\ell(\mathcal{F}^{-}(x_a), y_a)$ **when removing**	
	an innocent cluster	**a poison cluster**
0.09 ± 0.02	0.02 ± 0.00	6.91 ± 0.6

Table 5.3: The cross-entropy loss of the misclassification event on the original and modified models, for BadNet-CIFAR10.

5.1.3 Traceback Overhead

Finally, we report in Table 5.2 (the last column) the runtime of traceback against different attack tasks. We run a prototype of our traceback system on a machine with one Nvidia Titan X GPU and 12 Intel Xeon CPUs. The computation time linearly increases with the dimension of the data projection and the number of training data samples. For models with a large training dataset, the bulk of the traceback computation comes from the clustering of training data, which takes up 83% of the computation time for Trojan-VGGFace, the most computational expensive task. On the other hand, simple data parallelism enabled by the use of mini-batch K-mean clustering can significantly speed up the runtime. For example, parallelizing using 5 machines reduces the runtime for Trojan-VGGFace to 49.5 ± 3.2 minutes, a 4.1x speed up.

CHAPTER 6

EVALUATION ON CLEAN-LABEL ATTACKS

We now evaluate our traceback system against clean-label poisoning attacks, and contrast its performance to that on dirty-label attacks. Compared to dirty-label attacks, clean-label attacks follow a different attack methodology, use fewer poison training samples, and these samples appear less separated from the benign data even after data projection. These factors make the clustering and pruning process more challenging. Nevertheless, our traceback system still achieves good performance across all attack tasks ($\geq 98.4\%$ precision and $\geq 96.8\%$ recall). In the following, we present the experiment setup of the five clean-label attacks used by our evaluation, and how our traceback system responds to these attacks.

6.0.1 Experiment Setup

Attack Setup. We consider two state-of-the-art triggerless clean-label attacks on image classification, and a backdoor-based clean-label attack on malware classification (Table A.5 in Appendix). Our implementation of these attacks match the original papers in both attack success rate and benign classification rate. Further implementation details are in Appendix A.0.2.

- **Bullseye Polytope (BP) (CIFAR10 & ImageNet):** The BP attack [1] aims to make the model classify a single attack sample x_a to a target class y_a at test time without modifying the sample (hence, triggerless). This is done by adding imperceptible perturbations to the poison training data so their representations in the feature space form a fixed-radius polytope around the chosen attack sample x_a. The classifier then associates the region around the attack sample with the label of the poison data, which is the desired target label y_a. This leads to misclassification of x_a to y_a at test time.

  It is known that BP only works well when the attacker has access to a pretrained feature extractor used by the victim model, *i.e.,* it is effective in the transfer learning setting. We follow this setup and use a feature extractor pretrained on the benign data as the victim. We use the attack parameters of the BP-5x attack (the strongest variant) from the original paper and test the attack on CIFAR10 and ImageNet.

- **Witches' Brew (CIFAR10 & ImageNet):** Witches' Brew [26] is a clean-label triggerless attack that. It works by adding imperceptible perturbations to the poison data to align its gradient with that of the attack sample. This makes the model misclassify the attack sample to the target class of the poison data. We test the attack on CIFAR10 and ImageNet datasets.
- **Malware Backdoor (Ember Malware):** This is a clean-label, *backdoor* attack on malware classifiers [65]. The attacker uses influence functions to find 128 most important features defining 'goodware' and uses these as a trigger. These features are then modified for poison malware samples, with the target class being 'goodware'. The authors of [65] found all three of the state-of-the-art defenses [15, 46, 75] are ineffective against this attack. Like the original paper, we use the publicly-available Ember malware classification dataset. Since attackers are mostly interested in disguising malware as 'goodware', we only use 'goodware' as the target label of the attack.

Traceback System Setup & Evaluation Metrics. We use the same system setup and evaluation metrics as §5.1.

6.0.2 Traceback Performance

The bottom section of Table 5.2 shows that our traceback system achieves $> 98.4\%$ precision and $> 96.8\%$ recall when going against the five clean-label poisoning attacks. In the following, we discuss these results in detail by contrasting the trackback performance on clean-label attacks to that on dirty-label attacks (discussed in § 5.1.2).

Lower traceback recall due to ineffective poison data. First, we see that the traceback precision remains high even as compared to dirty-label attacks, but the recall is consistently lower and the runtime is higher. In particular, Table 5.2 shows that the lowest traceback recall happens on the two triggerless attacks, BP and Witches' Brew. We hypothesize that it is because these two attacks failed to move the representations of some poison training data to the desired location in the feature space. These "ineffective" poison training data made very little contribution to the misclassification event, and are hard to detect during traceback.

We test and validate this hypothesis by gradually increasing the attack perturbation budget (to increase the effectiveness of the poison training data) and observing the attack success rate. The budget for the BP and WitchBrew attacks determines the magnitude of perturbation the attacker can add to each poison data point (the default budget is set to $L_{inf} = 0.03$). A higher perturbation budget allows the attacker to position the poison training data "closer" to their desired locations, making these data more "effective" and leading to a stronger attack success rate. Table A.3 confirms that the attack success rate increases with the perturbation budget. We also list the traceback precision and recall. While precision remains high, the recall also increases from 94.9% to 99.4% as we increase the budget, confirming our hypothesis that the presence of ineffective poison samples is the reason for a lower recall value.

Less distinct clusters lead to more pruning iterations. We also study the performance of clustering and pruning on the five clean-label attacks. Like Figure 5.1 for dirty-label attacks, we visualize the projected training data using 2-D PAC for clean-label attacks. The visualization for BP-CIFAR10 is shown in Figure A.1. Compared to the wide separation seen on the dirty-label poison data (Figure 5.1), the BP poison data appear much less separated from the benign training data. In fact, they reside in the gap between different benign clusters. This is not surprising, because clean-label attacks work by moving the representation of the poison data close to that of benign data in the target class, so these poison data are inherently closer to the benign data.

We also analyze the pruning operation when tracing back clean-label attacks. Table 6.1 shows the cross-entropy loss caused by training data removal, which is used to determine whether a cluster is benign or not. Again, we observe a clear pattern that separates benign clusters from poison (or mixed) clusters, *i.e.,* $\ell(\mathcal{F}^{-}(x_a), y_a)$ for removing an innocent cluster is smaller than $\ell(\mathcal{F}(x_a)$, and $\ell(\mathcal{F}^{-}(x_a), y_a)$ for removing a poison cluster is significantly larger. As such, pruning can effectively identify benign clusters.

$\ell(\mathcal{F}(x_a), y_a)$	$\ell(\mathcal{F}^{-}(x_a), y_a)$ **when removing**	
	an innocent cluster	**a poison cluster**
0.61 ± 0.07	0.39 ± 0.04	8.81 ± 0.81

Table 6.1: The cross-entropy loss of the misclassification event on the original and modified models, for BP-CIFAR10.

Attack-Dataset	**Traceback Method**		
	Spectral Signature	Neural Cleanse	**Ours**
BadNet-CIFAR10	95.3% / 92.4%	98.7% / 96.6%	**99.5% / 98.9%**
BadNet-ImageNet	96.0% / 93.7%	91.1% / 97.2%	**99.1% / 99.1%**
Trojan-VGGFace	93.1% / 89.8%	94.8% / 97.4%	**99.8% / 99.9%**
Physical-Wenger	43.2% / 67.4%	0% / 0%	**99.5% / 97.1%**
Malware Backdoor	2.1% / 15.1%	0% / 0%	**99.2% / 98.2%**

Table 6.2: Comparing our traceback system against forensic tools adapted from existing backdoor defenses. We present the results as "Precision / Recall".

Despite the reduced separation, our traceback system can still accurately identify the benign (and thus poison) data clusters while using more clustering/pruning iterations. For example, an average of 10.7 iterations are needed for BP-CIFAR10 compared to 3.2 iterations for the dirty-label attack on the same model (BadNet-CIFAR10).

6.1 Comparing against Adapted Defenses

While forensics tools are solving a different problem as defenses, it is reasonable to ask, can existing defenses be adapted to become tools in forensic analysis. Here, we adapt four state-of-the-art poison defenses (two backdoor and two triggerless defenses) to perform post-attack traceback analysis. We compare our system against adapted backdoor defenses (Spectral Signature [75] and Neural Cleanse [79]) in Table 6.2, and our system against triggerless defenses (Deep K-NN and L_2-Norm Outliers [59]) in Table 6.3.

Spectral Signature. Spectral signature [75] extracts signatures for backdoor training data using the spectrum of the covariance of the feature representation. Spectral signature identifies the malicious training samples post-training (before attack). We adapt spectral signature simply by using it to identify the malicious training data post-attack.

Table 6.2 shows that spectral signature performs well on BadNet and Trojan attacks, but performs quite poorly when tracing back attacks for physical backdoors and malware backdoors. The

Attack-Dataset	**Traceback Method**		
	Deep K-NN	L_2-Norm	**Ours**
BP-CIFAR10	36.1% / 74.3%	34.5% / 78.0%	**98.4% / 96.8%**
BP-ImageNet	57.9% / 79.6%	53.4% / 72.4%	**99.3% / 97.4%**
WitchBrew-CIFAR10	49.3% / 53.9%	52.1% / 42.8%	**99.7% / 96.8%**
WitchBrew-ImageNet	53.5% / 47.2%	51.3% / 44.3%	**99.1% / 97.9%**

Table 6.3: Comparing our traceback system against forensic tools adapted from existing clean-label defenses. We present the results as "Precision / Recall".

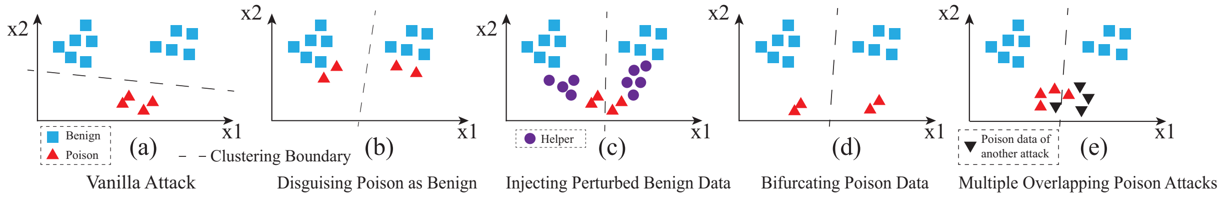

x2
x1
Benign
Poison
(a)
Clustering Boundary
(b)
Helper
(c)
(d)
Poison data of another attack
(e)
Vanilla Attack
Disguising Poison as Benign
Injecting Perturbed Benign Data
Bifurcating Poison Data
Multiple Overlapping Poison Attacks

Figure 6.1: An illustration of four countermeasures where attacker can manipulate the data layout in data projection space in order to disrupt traceback.

poor performance against Malware Backdoors is consistent with the malware paper itself [65], where the authors showed that spectral signature defense is ineffective against their proposed malware attack.

Neural Cleanse (NC). NC [79] recovers backdoor triggers in a poisoned model by reverse-engineering the backdoor trigger. For traceback, we apply NC and search for the recovered trigger in the training dataset to identify poison data. Since NC does not recover the exact input trigger, we perform the matching in neuron activation space, *i.e.,* flagging the training data if its neuron activation is close to that of the recovered trigger. We use cosine similarity to measure the activation distance and use a small set of benign data to calculate a cutoff threshold.

In our tests, NC performs well on BadNet and Trojan backdoors, where it successfully recovers backdoor triggers. Against physical and malware backdoor attack, however, NC fails to identify any triggers, and thus traceback fails.

Deep K-NN and L_2-Norm Outliers. Both defenses are proposed by [59], where clean-label poison data are detected using anomaly detection in the feature space. In our tests, both defenses perform poorly on all four clean-label attack tasks (Table 6.3), achieving similar performance as random guessing. This result is also consistent with results from the BP and WitchBrew attack papers, where they show existing defenses are not effective [1, 26].

CHAPTER 7

ANTI-FORENSIC COUNTERMEASURES

As with other security mechanisms, we expect attackers with knowledge of our system to deploy adaptive countermeasures. In this section, we explore potential anti-forensics techniques and evaluate them for their impact on our system.

Our experiments make the strongest possible assumption about the attacker: that they know full details of the target model's training pipeline, including read access to all training data, model architecture, and training hyperparameters. They also know details of the traceback system, including the function to calculate the data projection. We assume that attacker can add additional perturbations to its poison data in order to evade traceback, and we assume a generous perturbation budget of $L_{inf} = 0.1$ (attacker can change each input value by 10%). For trigger-based attacks, we further allow the attacker to arbitrarily change the location and value of the triggers.

We consider five different countermeasures, each leveraging in different ways the attacker's ability to inject manipulated training data into the data pipeline. The countermeasures are shown in Figure 6.1, and include: a) disguising poison data as benign, b) injecting perturbed benign data, c) bifurcating poison data, and d) multiple overlapping poison attacks. We evaluate the countermeasures on a total of 7 attacks, all of our attacks except for physical backdoors (cannot easily modify physical triggers) and malware backdoors (fixed triggers based on heuristic). We also test the scenarios where attacker increases the attack's cost in order to evade traceback.

Disguising Poison as Benign. The first countermeasure tries to confuse the clustering algorithm, by perturbing the poison data and pushing it closer to benign data in the data projection space (Figure 6.1b). If the clustering heuristic groups poison data into different benign clusters, it will terminate the pruning and either reduce recall or confuse the system into identifying it as a non-poison misclassification.

We test the case where an attacker minimizes the distance between each poison data and a benign data point closest to it in the data projection space, while optimizing for the original attack

L_2 **Distance**	**Attack Success Rate**	**Precision**	**Recall**
434.5 ± 8.2	$99.5 \pm 0.0\%$	$99.5 \pm 0.0\%$	$98.9 \pm 0.0\%$
290.4 ± 8.9	$89.5 \pm 0.6\%$	$98.4 \pm 0.1\%$	$98.1 \pm 0.1\%$
184.8 ± 5.7	$64.3 \pm 1.7\%$	$96.9 \pm 0.3\%$	$97.2 \pm 0.1\%$
110.3 ± 3.1	$28.3 \pm 3.2\%$	$95.9 \pm 0.3\%$	$96.7 \pm 0.2\%$
59.0 ± 1.9	$0.0 \pm 0.0\%$	N/A	N/A
31.4 ± 1.2	$0.0 \pm 0.0\%$	N/A	N/A

Table 7.1: For disguising Trojan-VGGFace, attack success rate drops as the L_2 distance between poison and benign projections decreases, while traceback precision and recall drop slightly.

L_2 **Distance**	**Attack Success Rate**	**Precision**	**Recall**
28.2 ± 0.9	$86.1 \pm 1.4\%$	$98.4 \pm 0.1\%$	$96.8 \pm 0.2\%$
25.4 ± 1.3	$59.4 \pm 3.4\%$	$97.2 \pm 0.2\%$	$95.0 \pm 0.3\%$
19.0 ± 1.2	$19.7 \pm 2.0\%$	$96.1 \pm 0.2\%$	$94.8 \pm 0.3\%$
12.8 ± 0.8	$8.7 \pm 0.4\%$	$95.9 \pm 0.1\%$	$95.1 \pm 0.3\%$
9.4 ± 0.5	$0.0 \pm 0.0\%$	N/A	N/A
4.4 ± 0.3	$0.0 \pm 0.0\%$	N/A	N/A

Table 7.2: For disguising BP-CIFAR10, attack success rate decreases as L_2 distance between poison and benign projections decreases, while traceback precision and recall drop slightly.

objectives, using a λ parameter to balance the two objectives. The attacker leverages a *bi-level* optimization [34] to optimize the attack objective, since the modifications of poison data also change the result model and thus, the projection calculation.

Results are consistent across all 7 attack tasks. For brevity, we show detailed results for Trojan-VGGFace (Table 7.1) and BP-CIFAR10 (Table 7.2). As the L_2 distances between poison and benign data decrease, it becomes harder for the model to learn the attack, and attack success rate drops to zero. Impact on traceback is minimal: precision and recall drop less than 3.6% across all tasks.

Targeted Contamination Attack (TaCT). Another method to disguise poison data is the TaCT attack [74]. TaCT is designed to obscures the difference between poison and benign representations in order to evade existing defenses. TaCT leverages source-specific backdoors, triggers *only* work when applied to specific images. The authors found that TaCT causes poison and benign data to be inseparable in neuron representation space.

We test our traceback system against TaCT. We find that while poison and benign data are inseparable in neuron representation space, we can still separate them in our data projection space. We believe the separation is likely because our data projections measure each data's *impact* on model parameters, which better characterize training data than neurons activation values in poison settings. We test TaCT attack on all 4 backdoor attack tasks from §5.1. Overall, TaCT attack is ineffective against our traceback system, reducing precision and recall by less than 4% across the five backdoor attacks.

Injecting Perturbed Benign Data. Next, we consider the countermeasure where the attacker seeks to confuse the clustering algorithm by injecting benign data whose projection fills in the space between benign and poison data. We refer to this additional set of modified data as "helper data". We note that the helper data do not directly contribute to the success of the misclassification event.

As shown in Figure 6.1(c), an attacker positions helper data between poison and benign data to mislead the clustering heuristic. The attacker first identifies the *last* set of benign data pruned out by the traceback system. Using the clustering algorithm, the attacker separates the benign cluster into two clusters, and the poison data into two groups based on proximity to each benign cluster's centroid. The attacker optimizes helper data to uniformly position them in between each benign cluster and its closeby poison cluster. The attacker uses a similar bi-level optimization (§7) to optimize the attack objective.

We apply this countermeasure on the 7 attack tasks. For brevity, we show detailed results for Trojan-VGGFace in Table 7.3. As the number of helper data samples increases to 10000 (25% of training data), attack success rate reduces gradually to zero, while traceback precision drops to 94.1% and recall remains the high ($> 98.7\%$). Table 7.4 shows results for BP-CIFAR10, where the attack success rate drops much faster when injecting merely 25 helper data. The faster drop in attack success is likely due to the fewer clean-label poison samples and their proximity to benign data. When Trojan and BP attacks reach zero attack success rate (15000 and 20 helper data respectively), our traceback can still separate poison data with $> 91.4\%$ precision and recall.

Bifurcating Poison Data. Next, we explore techniques to separate poison data into multiple (two) separate distributions both of which contribute to the attack incident while residing in differ-

Number of Helper Data	Attack Success Rate	Precision	Recall
0	$99.8 \pm 0.0\%$	$99.8 \pm 0.0\%$	$99.9 \pm 0.0\%$
1000	$64.3 \pm 3.8\%$	$96.3 \pm 0.2\%$	$99.1 \pm 0.0\%$
10000	$21.0 \pm 3.9\%$	$94.1 \pm 0.3\%$	$98.7 \pm 0.0\%$
15000	$0.0 \pm 0.0\%$	N/A	N/A

Table 7.3: For adding helper data to Trojan-VGGFace, the attack success rate decreases as the number of helper data increases, while the precision and recall of the traceback system drop slightly.

Number of Helper Data	Attack Success Rate	Precision	Recall
0	$86.1 \pm 1.4\%$	$98.4 \pm 0.1\%$	$96.8 \pm 0.2\%$
5	$35.5 \pm 3.4\%$	$96.6 \pm 0.3\%$	$96.6 \pm 0.1\%$
10	$13.1 \pm 1.1\%$	$94.3 \pm 0.5\%$	$96.6 \pm 0.2\%$
20	$0.0 \pm 0.0\%$	N/A	N/A

Table 7.4: For adding helper data to BP-CIFAR10, the attack success rate decreases as the number of helper data increases, while the recall of the traceback system remains the same and precision drops slightly.

ent parts of data projection space, in order to evade clustering (Figure 6.1(d)). The attacker first identifies the two strongest clusters in the poison data, then maximize the distance between the cluster centroids to separate them. We follow the same bi-level optimization process to optimize the poison data and use a λ term to balance the objective of cluster distance and the original attack objective.

We apply this countermeasure to all 7 attack tasks. For BP attacks and WitchBrew attacks, attack success drops quickly because the two triggerless attacks rely on clever positioning of the poison data (*e.g.,* a fixed radius polytope around target data). For BP and Witches' Brew, this countermeasure has no impact on traceback performance ($> 97.0\%$ precision and recall). For trigger-based attacks, we show results on Trojan-VGGFace in Table 7.5. We found that as we increase λ to push for better separation between the two clusters, the centroid distances fail to increase beyond a certain value. We believe the failure to separate poison data is because these poison samples have the same attack objective and trigger, and naturally cluster together in the data projection space. Overall, the traceback system achieves $\leq 96.7\%$ precision and recall across all 7 attack tasks.

Multiple Overlapping Poison Attacks. Finally, an attacker can try to combine two dirty-label

L_2 **Distance**	**Attack Success Rate**	**Precision**	**Recall**
2.2 ± 0.2	$99.8 \pm 0.0\%$	$99.8 \pm 0.0\%$	$99.9 \pm 0.0\%$
17.9 ± 2.7	$98.3 \pm 0.2\%$	$98.2 \pm 0.0\%$	$97.9 \pm 0.1\%$
25.3 ± 4.1	$97.1 \pm 3.7\%$	$97.3 \pm 0.2\%$	$96.9 \pm 0.2\%$
23.6 ± 5.8	$97.4 \pm 0.0\%$	$97.5 \pm 0.1\%$	$97.3 \pm 0.1\%$
24.0 ± 6.1	$98.1 \pm 0.0\%$	$97.6 \pm 0.2\%$	$97.0 \pm 0.2\%$

Table 7.5: For separate one Trojan attack into two, the attack success rate decreases as the L_2 distance between centroids decreases, while the precision of the traceback system remains the same and recall drops slightly.

attacks in one misclassification event, by training two different triggers with the same misclassification label into the model, then including both triggers into a single attack input. This attack does not work for triggerless attacks, since each attack has its own specific target data.

Our experiments show this countermeasure is *ineffective* against our traceback system. We achieve $> 98.4\%$ precision and recall across all attack tasks. While the two poison attacks leverage different triggers, they have the same objective of misclassifying any inputs to the same target label, and our data projection directly correlates to the objective of each training data. Thus poison data from two separate attacks appear in the same region in projection space, enabling us to cluster them together as part of the same attack.

Higher Cost Attacks. So far, we have focused on attacks with a similar cost, *i.e.,* same number of poison data. Now, we explore the impact of attacks with higher cost on our traceback system. We allow an adaptive attacker to poison an increasing number of poison data and test our traceback effectiveness against these higher cost attacks. We found that increasing injection rate has an surprisingly low impact on our traceback system. As attacker increases injection rate to 50%, our traceback system maintains $> 95\%$ precision and $> 92\%$ recall, across all 5 countermeasures discussed in this section and all 9 attack tasks.

We believe the weak impact of increasing injection rate is because our traceback system views poison as a group effect (§3.0.3), and poison data with the same attack objective are clustered together regardless of the number of poison data. As a result, increasing injection rate has limited effectiveness against our traceback system.

CHAPTER 8
DISCUSSION: IDENTIFYING NON-POISON EVENTS

Our work addresses the question of post-attack analysis for poison attacks on neural networks. In practice, however, a system administrator must first identify if a misclassification event was caused by a poisoning attack, or from an *evasion attack* or *benign misclassification*. The former are test-time attacks that leverage existing vulnerabilities in trained models to cause misclassification with perturbed data, while the latter simply arise since models do not classify perfectly.

Attack Identification. We note that our system can double as a tool for the first step towards attack identification. Given a model and a misclassification event (misclassified input and output), one iteration of our forensic system would be able to identify if the attack was a poison attack or caused by other means. Once we separate training data into clusters, and apply the same unlearning techniques, we can observe if removal of either subset of training data will alter misclassification behavior.

Intuitively, both evasion by adversarial perturbation and benign misclassification rely on specifics of the model's loss landscape. In either case, removal or "unlearning" of any significant portion of training data will change the loss landscape and should alter the misclassification behavior. In our system, we would observe that unlearning either of the clusters would alter the misclassification event. So if the first iteration fails to prune away either cluster, then we consider the misclassification event as non-poison and end traceback.

Limitation. Our attack identification system can be vulnerable to "false flag" attack [60] where an attacker carefully crafts a misclassification event that triggers our traceback system to blame an innocent data provider. This is a threat that the deployer of traceback system needs to keep in mind when perform any forms of prosecution based on traceback results. In practice, a system like ours must be understood in context: the assignment of blame to any parties involved will only be possible with the availability of an effective data provenance tool, and there should be a human-in-the-loop confirmation before any lasting decisions are made.

Further, any attacker trying to carry out an inference time attack that also has a false flag component will have to solve a more challenging optimization problem needing access to the training data. The exploration of techniques to do this effectively is beyond the scope of this paper but is an interesting direction for future work.

Empirical Results. We test four representative *evasion attacks*, including two white-box evasion attacks (PGD [52], CW [13]) and two black-box evasion attacks (Boundary [9] and HSJA [16]) against all 5 of our evaluation datasets. We follow the default attack parameters [68] (Appendix A.2). We test 100 evasion attack samples for each attack and classification task pair. For *benign misclassification*, we randomly select 100 misclassified benign test data as the misclassification events for each task. For all these misclassification events, our forensic tool correctly determined that they were not caused by data poisoning attacks, *i.e.,* our tool produced no false positives.

Future Work

We believe the study of post-attack forensic techniques remains an open area with numerous open questions. First, while our method is effective in our tests, more effort is needed to understand its robustness when extended to additional types of attacks and application domains. Second, in order to assign responsibility to a provider for the poisoned data, the model administrator must be confident that metadata associated with training data is accurate and tamper-resistant. The study of data provenance [22, 49] in this context remains an open problem.

REFERENCES

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

[2] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of ICML*. PMLR, 2018.

[4] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. *arXiv preprint arXiv:2005.03823*, 2020.

[5] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

[6] Adam Bates, Dave Jing Tian, Kevin RB Butler, and Thomas Moyer. Trustworthy whole-system provenance for the linux kernel. In *Proc. of USENIX Security*, 2015.

[7] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *Proc. of ICASSP*. IEEE, 2021.

[8] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proc. of IEEE S&P*. IEEE, 2021.

[9] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[10] Nicholas Carlini. Poisoning the unlabeled dataset of semi-supervised learning. In *Proc. of USENIX Security*, 2021.

[11] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[12] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. of AISec*, 2017.

[13] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P*. IEEE, 2017.

[14] Eoghan Casey. *Handbook of digital forensics and investigation*. Academic Press, 2009.

[15] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[16] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *Proc. of IEEE S&P*. IEEE, 2020.

[17] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to ip traceback. *Proc. of TISSEC*, (2), 2002.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of CVPR*. IEEE, 2009.

[19] Hailun Ding, Shenao Yan, Juan Zhai, and Shiqing Ma. Elise: A storage efficient logging system powered by redundancy reduction and representation learning. In *Proc. of USENIX Security*, 2021.

[20] Tino Feri Efendi. The management of physical evidence and chain of custody (coc) in digital forensic laboratory storage. *International Journal of Seocology*, 2019.

[21] Soowoong Eo, Wooyeon Jo, Seokjun Lee, and Taeshik Shon. A phase of deleted file recovery for digital forensics research in tizen. In *Proc. of ICITCS*. IEEE, 2015.

[22] Ru Fang, Hui-I Hsiao, Bin He, C Mohan, and Yun Wang. High performance database logging using storage class memory. In *Proc. of ICDE*. IEEE, 2011.

[23] Peng Fei, Zhou Li, Zhiying Wang, Xiao Yu, Ding Li, and Kangkook Jee. {SEAL}: Storage-efficient causality analysis on enterprise logs with query-friendly compression. In *Proc. of USENIX Security*, 2021.

[24] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proc. of ACSAC*, 2019.

[25] Daniel B Garrie. Digital forensic evidence in the courtroom: understanding content and quality. *Nw. J. Tech. & Intell. Prop.*, 2014.

[26] Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

[27] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. What doesn't kill you makes you robust (er): Adversarial training against poisons and backdoors. *arXiv preprint arXiv:2102.13624*, 2021.

[28] Greg Gogolin. *Digital forensics explained*. CRC Press, 2021.

[29] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *Proc. of MLCS Workshop*, 2017.

[30] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

[31] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Defense against backdoor attacks via robust covariance estimation. In *Proc. of ICML*. PMLR, 2021.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

[33] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *arXiv preprint arXiv:2106.04690*, 2021.

[34] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *arXiv preprint arXiv:2004.00225*, 2020.

[35] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning attacks. *arXiv preprint arXiv:2012.03765*, 2020.

[36] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. *arXiv preprint arXiv:2008.04495*, 2020.

[37] Fabian Kaczmarczyck, Bernhard Grill, Luca Invernizzi, Jennifer Pullman, Cecilia M Procopiuc, David Tao, Borbala Benko, and Elie Bursztein. Spotlight: Malware lead generation at scale. In *Proc. of ACSAC*, 2020.

[38] Vishal Karande, Erick Bauman, Zhiqiang Lin, and Latifur Khan. Sgx-log: Securing system logs with sgx. In *Proc. of AsiaCCS*, 2017.

[39] Samuel T King and Peter M Chen. Backtracking intrusions. In *Proc. of SOSP*, 2003.

[40] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proc. of ICML*. PMLR, 2017.

[41] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[42] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *Proc. of SPW*. IEEE, 2020.

[43] Ioannis Lazaridis, Theodoros Arampatzis, and Sotirios Pouros. Evaluation of digital forensics tools on data recovery and analysis. In *Proc. of CSCESM*, 2016.

[44] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. Loggc: garbage collecting audit log. In *Proc. of CCS*, 2013.

[45] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307*, 2018.

[46] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proc. of ICDM*. IEEE, 2008.

[47] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Proc. of RAID*. Springer, 2018.

[48] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proc. of NDSS*, 2018.

[49] Rongxing Lu, Xiaodong Lin, Xiaohui Liang, and Xuemin Shen. Secure provenance: the essential of bread and butter of data forensics in cloud computing. In *Proc. of CCS*, 2010.

[50] Shiqing Ma, Juan Zhai, Yonghwi Kwon, Kyu Hyung Lee, Xiangyu Zhang, Gabriela Ciocarlie, Ashish Gehani, Vinod Yegneswaran, Dongyan Xu, and Somesh Jha. Kernel-supported cost-effective audit logging for causality tracking. In *Proc. of USENIX Security*, 2018.

[51] Shiqing Ma, Xiangyu Zhang, and Dongyan Xu. Protracer: Towards practical provenance tracing by alternating between logging and tainting. In *Proc. of NDSS*, 2016.

[52] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[53] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proc. of ALT*. PMLR, 2021.

[54] James Newsome and Dawn Xiaodong Song. Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software. In *Proc. of NDSS*. Citeseer, 2005.

[55] Nicolas Papernot. A marauder's map of security and privacy in machine learning. *arXiv preprint arXiv:1811.01134*, 2018.

[56] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. *Proc. of BMVC*, 2015.

[57] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The Philosophical Magazine*, (11), 1901.

[58] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proc. of NSDI*, 2010.

[59] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. Deep k-nn defense against clean-label data poisoning attacks. In *Proc. of ECCV*. Springer, 2020.

[60] Thomas Rid and Ben Buchanan. Attributing cyber attacks. *Journal of Strategic Studies*, (1-2), 2015.

[61] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proc. of AAAI*, number 07, 2020.

[62] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In *Proc. of SIGCOMM*, 2000.

[63] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *Proc. of USENIX Security*, 2021.

[64] David Sculley. Web-scale k-means clustering. In *Proc. of WWW*, 2010.

[65] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Explanation-guided backdoor poisoning attacks against malware classifiers. In *Proc. of USENIX Security*, 2021.

[66] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018.

[67] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[68] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y Zhao. Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proc. of CCS*, 2020.

[69] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting privacy against unauthorized deep learning models. In *Proc. of USENIX Security*, 2020.

[70] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross Anderson. Manipulating sgd with data ordering attacks. *arXiv preprint arXiv:2104.09667*, 2021.

[71] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[72] Dawn Xiaodong Song and Adrian Perrig. Advanced and authenticated marking schemes for ip traceback. In *Proc. of IEEE INFOCOM*. IEEE, 2001.

[73] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. of AAAI*, 2017.

[74] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *Proc. of USENIX Security*, 2021.

[75] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *arXiv preprint arXiv:1811.00636*, 2018.

[76] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[77] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proc. of ECCV*, 2018.

[78] Binghui Wang, Xiaoyu Cao, Neil Zhenqiang Gong, et al. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020.

[79] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. of IEEE S&P*. IEEE, 2019.

[80] Emily Wenger, Josephine Passananti, Arjun Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y. Zhao. Backdoor attacks against deep learning systems in the physical world. In *Proc. of CVPR*, 2021.

[81] Yulai Xie, Dan Feng, Zhipeng Tan, Lei Chen, Kiran-Kumar Muniswamy-Reddy, Yan Li, and Darrell DE Long. A hybrid approach for efficient provenance storage. In *Proc. of CIKM*, 2012.

[82] Zhang Xu, Zhenyu Wu, Zhichun Li, Kangkook Jee, Junghwan Rhee, Xusheng Xiao, Fengyuan Xu, Haining Wang, and Guofei Jiang. High fidelity data reduction for big data security dependency analyses. In *Proc. of CCS*, 2016.

[83] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proc. of CCS*, 2019.

[84] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proc. of CCS*, 2007.

[85] Le Yu, Shiqing Ma, Zhuo Zhang, Guanhong Tao, Xiangyu Zhang, Dongyan Xu, Vincent E Urias, Han Wei Lin, Gabriela Ciocarlie, Vinod Yegneswaran, et al. Alchemist: Fusing application and audit logs for precise attack provenance without instrumentation. In *Proc. of NDSS*, 2021.

[86] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[87] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *Proc. of ICML*. PMLR, 2019.

APPENDIX A

APPENDIX

A.0.1 Theoretical Analysis of Training Data Removal

As discussed in §3.0.3, our binary measure of event responsibility is inspired by a theoretical analysis on how removing a portion of the training data affects the poisoning attack performance. We now present this theoretical analysis in detail.

To examine the impact of removing a subset of training data (D_1) on the poisoning attack, we seek to analytically quantify its impact on the model loss over the true distribution of the poison test data, which indicates the contribution of D_1 to successful misclassification at test time. Furthermore, our analysis considers the general case where D_1 may contain both benign and poison training data, *i.e.,* D_1 is drawn from a mixed distribution. We note that while our theoretical analysis is driven by the expected value of the loss over the distribution of the poison test data, in practice the traceback system can only measure the impact on a single misclassification event (usually just one data sample). Yet this is empirically sufficient to label and prune clusters (as shown by our experimental results in §5.1 and 6). Finally, since our clustering is able to find clusters that only contain benign data, the pruning component used by our traceback focuses on picking this cluster (D_1). This is a special case under our theorems, which show that clusters can be differentiated even if they are mixed.

Definitions: Let the full training data D be drawn from a distribution $\mathcal{D}$ comprised of the benign distribution $\mathcal{D}_b$ and the poison distribution $\mathcal{D}_p$ in the ratio α, *i.e.,* $\mathcal{D} = \alpha\mathcal{D}_b + (1-\alpha)\mathcal{D}_p$. Let $\mathcal{F}$ denote the original model trained on D, using the loss function ℓ. To measure the impact of removing a group of data D_1 from the training dataset, we consider a new model $\mathcal{F}^-$ trained on $D \setminus D_1$, effectively drawn from a distribution $\mathcal{D}^- = \alpha^-\mathcal{D}_b + (1-\alpha^-)\mathcal{D}_p$. Finally, the expected loss over the true distribution for a classifier $\mathcal{F}$ is $L_{\mathcal{D}}(\mathcal{F})$= $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(\mathcal{F}(x), y)\right]$.

Key Results: If removing D_1 from the model training process either reduces or maintains the loss of the resulting classifier on the poison test data, *e.g.,* $L_{\mathcal{D}_p}(\mathcal{F}) \geq L_{\mathcal{D}_p}(\mathcal{F}^-)$, this action has skewed the ratio towards poison data in the training dataset, implying that $D \setminus D_1$ is more responsible for the success of poison attacks at test time.

Next, we prove this result in two cases: i) learning from the entire distribution and ii) learning from the empirical distribution. We note that the former is not possible in practice but is useful pedagogically.

Theorem 1. *[Learning from true distribution] Consider classifiers $\mathfrak{F}_*$ and $\mathfrak{F}_*^-$ that are trained directly from the true distributions $\mathcal{D}$ and $\mathcal{D}^-$, respectively. We can show that if*

$$L_{\mathcal{D}_p}(\mathfrak{F}_*) \geq L_{\mathcal{D}_p}(\mathfrak{F}_*^-), \tag{A.1}$$

then $\alpha^- \leq \alpha$.

Proof. By definition,

$$\begin{aligned}\mathfrak{F}_* &= \underset{\mathfrak{F}}{\operatorname{argmin}}\, L_{\mathcal{D}}(\mathfrak{F}) \\ &= \underset{\mathfrak{F}}{\operatorname{argmin}}\, \alpha L_{\mathcal{D}_b}(\mathfrak{F}) + (1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F})\end{aligned}$$

and

$$\begin{aligned}\mathfrak{F}_*^- &= \underset{\mathfrak{F}}{\operatorname{argmin}}\, L_{\mathcal{D}^-}(\mathfrak{F}) \\ &= \underset{\mathfrak{F}}{\operatorname{argmin}}\, \alpha^- L_{\mathcal{D}_b}(\mathfrak{F}) + (1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}).\end{aligned}$$

The first implies that

$$\forall \mathfrak{F},\ \alpha L_{\mathcal{D}_b}(\mathfrak{F}_*) + (1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F}_*) \\ \leq \alpha L_{\mathcal{D}_b}(\mathfrak{F}) + (1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F}) \tag{A.2}$$

$$\Rightarrow \alpha L_{\mathcal{D}_b}(\mathfrak{F}_*) + (1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F}_*) \\ \leq \alpha L_{\mathcal{D}_b}(\mathfrak{F}_*^-) + (1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F}_*^-), \tag{A.3}$$

and the second

$$\forall \mathfrak{F},\ \alpha^- L_{\mathcal{D}_b}(\mathfrak{F}_*^-) + (1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}_*^-) \\ \leq \alpha^- L_{\mathcal{D}_b}(\mathfrak{F}) + (1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}) \tag{A.4}$$

$$\Rightarrow \alpha^- L_{\mathcal{D}_b}(\mathfrak{F}_*^-) + (1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}_*^-) \\ \leq \alpha^- L_{\mathcal{D}_b}(\mathfrak{F}_*) + (1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}_*). \tag{A.5}$$

We multiply Eq. A.3 by α^- and Eq. A.5 by α, which gives us

$$\alpha\alpha^- L_{\mathcal{D}_b}(\mathfrak{F}_*^-) + \alpha(1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}_*^-) - \alpha(1-\alpha^-) L_{\mathcal{D}_p}(\mathfrak{F}_*) \\ \leq \alpha\alpha^- L_{\mathcal{D}_b}(\mathfrak{F}_*^-) + \alpha^-(1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F}_*^-) - \alpha^-(1-\alpha) L_{\mathcal{D}_p}(\mathfrak{F}_*) \\ \Rightarrow (\alpha - \alpha^-)(L_{\mathcal{D}_p}(\mathfrak{F}_*) - L_{\mathcal{D}_p}(\mathfrak{F}_*^-)) \geq 0 \tag{A.6}$$

From Eq. A.6, it is clear that if $L_{\mathcal{D}_p}(\mathfrak{F}_*) \geq L_{\mathcal{D}_p}(\mathfrak{F}_*^-)$, $\alpha \geq \alpha^-$, proving our claim. □

Our proof did not make any assumptions about the convexity of the loss function or the type of learning algorithm used. This is due to the assumption that we are able to find classifiers that minimize the loss on the true distribution. This assumption does not hold in practice, and we typically use gradient descent algorithms over sampled data for training [67]. The next theorem deals with this case, but makes the additional assumptions that the set of possible classifiers is convex and that the loss function is convex, Lipschitz and bounded. The learning algorithm used is Stochastic Gradient Descent (SGD).

Theorem 2. *[Learning from empirical distribution] Consider datasets D and D_2 defined as above such that $D_2 \subset D$. The corresponding models $\mathfrak{F}$ and $\mathfrak{F}^-$ are defined over a B-bounded convex set and trained using SGD over a convex, ρ-Lipschitz loss function ℓ. D is drawn from $\mathcal{D} = \alpha\mathcal{D}_b + (1-\alpha)\mathcal{D}_p$, and D_2 from $\mathcal{D}' = \alpha^-\mathcal{D}_b + (1-\alpha^-)\mathcal{D}_p$. Then, we can show that if*

$$L_{\mathcal{D}_p}(\mathcal{F}) - L_{\mathcal{D}_p}(\mathcal{F}^-) \geq \frac{-(\alpha\epsilon^- + \alpha^-\epsilon)}{\alpha - \alpha^-} \tag{A.7}$$

then $\alpha^- < \alpha$.

Proof. In the learning setting of interest, we have, from Corollary 14.12 in Shalev-Shwartz and Ben-David [67],

$$L_{\mathcal{D}}(\mathcal{F}) \leq L_{\mathcal{D}}(\mathfrak{F}_*) + \epsilon, \tag{A.8}$$

where $\epsilon \geq \frac{B^2\rho^2}{|D|}$. Similarly, for learning from D^-, we have

$$L_{\mathcal{D}^-}(\mathcal{F}^-) \leq L_{\mathcal{D}^-}(\mathfrak{F}_*^-) + \epsilon^-, \tag{A.9}$$

where $\epsilon^- \geq \frac{B^2\rho^2}{|D^-|}$.

Now we can i) substitute $\mathcal{F}^-$ in the right hand side of Eq. A.2 and $\mathcal{F}$ in the right hand side of Eq. A.4, ii) use Eqs. A.8 and and iii) perform the appropriate scaling and rearrangement to get

$$(\alpha - \alpha^-)(L_{\mathcal{D}_p}(\mathcal{F}) - L_{\mathcal{D}_p}(\mathcal{F}^-)) \geq -(\alpha\epsilon^- + \alpha^-\epsilon). \tag{A.10}$$

If the condition from Eq. A.7 is true, then we have $\alpha > \alpha^-$. □

We can then determine that D_1 was *less responsible* than the remaining data $D \setminus D_1$ if the difference of losses satisfies the condition from Theorem 2. In other words, this implies that the remaining data $D \setminus D_1$ is more skewed towards the poison distribution $\mathcal{D}_p$, guiding our search for the set of poisoned data. The implication of the theorem above is that set searching is viable since for any identified set, its relative impact on the attack incident can be quantitatively determined. We note that the case when the removed set of data D_1 contains only benign data is a special case in the theorem above.

A.0.2 Further experimental details

Evaluation Dataset. We discuss in details of training datasets we used for the evaluation.

- *Image Recognition (CIFAR10)* - The task is to recognize 10 different objects. The dataset contains 50,000 training images and 10,000 testing images [41]. The model is an Wide Residual Neural Network (RNN) with 50 residual blocks and 1 dense layer [86]. We use this task because of its prevalence in general image classification and security literature.
- *Image Recognition (ImageNet)* - The task is to recognize 1000 different objects. The dataset contains 1,281,167 training images [18]. We include this task because it has been used as a general benchmark for computer vision and the large number of training data poses a challenge for our traceback system.
- *Face Recognition* (VGGFace) – This task is to recognize faces of $2,622$ different people drawn from the Internet. We include this task because it simulates a more complex facial recognition-based security screening scenario. Tracing back poison attack in this setting is important. Furthermore, the large set of labels and training data in this task allows us to explore the scalability of our system.
- *Malware Detection* (EMBER Malware) – Ember is a representative public dataset of malware and goodware samples. The dataset consists of 2,351-dimensional feature vectors extracted from Portable Executable (PE) files for the Microsoft Windows OS. We include the dataset to test traceback performance on malware detection.

Percentage of Weights Kept	Precision	Recall
0.1%	$98.9 \pm 0.0\%$	$98.0 \pm 0.0\%$
1%	$99.1 \pm 0.0\%$	$97.9 \pm 0.1\%$
5%	$99.2 \pm 0.0\%$	$97.9 \pm 0.1\%$

Table A.1: Precision and recall of traceback system remain the same as the precentage of weights kept for clustering increases for ImageNet-BadNet.

Attack Method	**Attack Configuration**
PGD	$\epsilon = 0.05$, step size = 9, max iterations = 1000, learning rate = 0.05
CW	$\epsilon = 0.05$, # of iteration = 100, epsilon of each iteration = 0.005
Boundary	$\epsilon = 0.05$, num_iterations = 10000, $\delta = 0.1$
HSJA	$\epsilon = 0.05$, num_iterations = 10000, $\gamma = 1.0$

Table A.2: Detailed information on evasion attacks.

Perturbation Budget (L_inf)	**Attack Success Rate**	**Precision**	**Recall**
0.01	29.2%	99.4%	**94.9%**
0.03	86.1%	98.4%	96.8%
0.05	93.7%	99.2%	99.2%
0.09	97.6%	99.1%	**99.4%**

Table A.3: For BP-CIFAR10, both attack success rate and traceback recall increase with the attack perturbation budget, because the poison training data becomes more effective.

Attack Name	**Dataset**	**Injection Rate**	**Benign Classification Accuracy**
BadNet	CIFAR10	10%	$92.9 \pm 0.3\%$
BadNet	ImageNet	10%	$78.9 \pm 1.7\%$
Trojan	VGGFace	10%	$76.1 \pm 0.8\%$
Physical Backdoor	Wenger Face	10%	$99.9 \pm 0.0\%$

Table A.4: The default setup of dirty-label poisoning attacks.

Attack Name	**Dataset**	**Injection Rate**	**Benign Classification Accuracy**
BP	CIFAR10	0.01%	$93.0 \pm 0.2\%$
BP	ImageNet	0.01%	$79.1 \pm 0.9\%$
WitchBrew	CIFAR10	1%	$92.2 \pm 0.3\%$
WitchBrew	ImageNet	1%	$79.3 \pm 0.7\%$
Malware Backdoor	EMBER Malware	1%	$99.2 \pm 0.1\%$

Table A.5: The default setup of the five clean-label poisoning attacks used in our evaluation.

Attack-Dataset	**Avg L_2 distance of poison data to**		**Avg # of iterations**
	benign centroid	**poison centroid**	
BadNet-CIFAR10	0.65	0.19	3.2
BadNet-ImageNet	0.76	0.24	3.4
Trojan-VGGFace	0.69	0.09	3.1
Physical-Wenger	0.62	0.39	4.0

Table A.6: The average L_2 distance between individual poison data and the benign (poison) centroid, and the number of pruning iterations needed to complete the traceback.

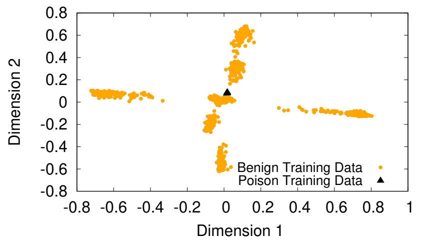


Figure A.1: 2-D PCA visualization of the projection of training data (sampled from BP-CIFAR10). Orange circles are innocent data and red crosses are poison data.