Identifying a Training-Set Attack's Target Using Renormalized Influence Estimation

Zayd Hammoudeh
University of Oregon
Eugene, Oregon, USA
zayd@cs.uoregon.edu

Daniel Lowd
University of Oregon
Eugene, Oregon, USA
lowd@cs.uoregon.edu

ABSTRACT

Targeted training-set attacks inject malicious instances into the training set to cause a trained model to mislabel one or more specific test instances. This work proposes the task of *target identification*, which determines whether a specific test instance is the target of a training-set attack. Target identification can be combined with *adversarial-instance identification* to find (and remove) the attack instances, mitigating the attack with minimal impact on other predictions. Rather than focusing on a single attack method or data modality, we build on influence estimation, which quantifies each training instance's contribution to a model's prediction. We show that existing influence estimators' poor practical performance often derives from their over-reliance on training instances and iterations with large losses. Our *renormalized* influence estimators fix this weakness; they far outperform the original estimators at identifying influential groups of training examples in both adversarial and non-adversarial settings, even finding up to 100% of adversarial training instances with no clean-data false positives. Target identification then simplifies to detecting test instances with anomalous influence values. We demonstrate our method's effectiveness on backdoor and poisoning attacks across various data domains, including text, vision, and speech, as well as against a gray-box, adaptive attacker that specifically optimizes the adversarial instances to evade our method. Our source code is available at https://github.com/ZaydH/target_identification.

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

Target Identification; Backdoor Attack; Data Poisoning; Influence Estimation; TracIn; GAS; Influence Functions; Representer Point

ACM Reference Format:
Zayd Hammoudeh and Daniel Lowd. 2022. Identifying a Training-Set Attack's Target Using Renormalized Influence Estimation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3548606.3559335

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3559335

1 INTRODUCTION

Targeted training-set attacks manipulate an ML system's prediction on *target* test instances by modifying the training data [2, 19, 26, 44, 54, 55, 63]. For example, a retailer may try to trick a filter into mislabeling its competitor's emails as spam [55]. Targeted attacks require very few corrupted instances [63], and their effect on test error is small, making them harder to detect [10] than *availability training-set attacks* [10], which are indiscriminate/untargeted and degrade an ML system's overall performance [6, 17, 68].

Kumar et al.'s [33] survey of business and government organizations found that training-set attacks were the top ML security concern, due to prior successful attacks [34]. Kumar et al. specifically identify defenses against such attacks as a significant, practical gap. Existing training-set defenses [18, 47, 69] mitigate the impact of the attack but provide little information about the attacker's goals, methods, or identity. Learning more about the attacker is essential for anticipating their attacks [48], designing targeted defenses [1], and even building defenses outside the ML system, such as stopping spammers through their payment processors [35].

This paper's defense against training-set attacks focuses on the related goals of learning more about an attacker and stopping their attacks. We achieve this through a pair of related tasks:

(1) *target identification*: identifying the target of a training-set attack, which may provide insight into the attacker's goals and how to defend against them; and
(2) *adversarial-instance identification*: identifying the malicious instances that constitute the training-set attack.

We are not aware of any work that studies training-set attack target identification beyond very simple settings.

Our *key insight* is the synergistic interplay between the two tasks above. If attackers can add only a few training instances (as is often the case) [10, 55, 63], then these malicious instances must be highly influential to change target predictions. Thus, targets are those test instances with an unusual number of highly influential training examples. In contrast, non-targets tend to have many weak influences and few very strong ones. Thus, if we can (1) determine which training instances influence which predictions and (2) detect anomalies in this distribution, then we can jointly solve both tasks.

Unfortunately, determining which training instances are responsible for which model behaviors remains a challenge, especially for complex, black-box models such as neural networks. *Influence estimators* [8, 11, 16, 30, 49, 70] attempt to quantify how much each training example contributes to a particular prediction. However, current influence analysis methods often perform poorly [5, 72].

This paper identifies a weakness common to many influence estimators [11, 30, 49, 70]: they induce a *low-loss penalty* that implicitly ranks confidently-predicted training instances as uninfluential. As

a result, they systematically overlook (groups of) highly influential, low-loss instances. We remedy this via a simple *renormalization* that removes the low-loss penalty. Our new *renormalized influence estimators* consistently outperform the originals in both adversarial and non-adversarial settings. The most effective of these, *gradient aggregated similarity* (GAS), often detects 100% of malicious training instances with no clean-data false positives.

Our framework for identifying targets of training-set attacks, FIT, compares the distribution of influence values across test instances checking for anomalies. More concretely, FIT marks as potential targets those test instances with an unusual number of highly influential instances as explained above. Next, FIT mitigates the attack's effect by removing exceptionally influential training instances associated with the target(s). Since mitigation considers only targets, training instance outliers that are "helpful" to non-targets are unaffected. This *target-driven mitigation* has a positive or neutral effect on clean data yet is highly effective on adversarial data where finding even a single target suffices to disable the attack on almost all other targets.

By relying on the concepts of influence estimation and not the properties of a particular attack or domain, GAS and FIT are *attack agnostic* [53]. They can apply equally well to different attack types, including *data poisoning* attacks, which target unperturbed test data, and *backdoor* attacks on test instances activating a specific trigger. Our approach works across data domains from CNN image classifiers to speech recognition to even text transformers.

In addition to learning more about the attack and attacker, *target identification enables targeted mitigation.* Certified training-set defenses [23, 28, 36, 59, 66, 67] (which do not identify targets) implement countermeasures (e.g., smoothing [64]) that affect predictions on *all* instances – not just the very few targets. These methods can substantially degrade performance, in some cases causing up to 10× more errors on clean data [17]. A strength of deep neural networks is that they can "memorize" instances to learn rare cases from one or two examples [16]; certified training prevents this by limiting any single training instance's influence.

Our work's contributions are enumerated below. Note that additional experiments and the proof are in the supplemental materials.

(1) We identify a weakness common to all gradient-based influence estimators and provide a simple renormalization correction that addresses this weakness.
(2) Inspired by influence estimation, we propose GAS – a renormalized influence estimator that is highly adept at identifying influential groups of training instances.
(3) Leveraging techniques from anomaly detection and *robust statistics*, we extend GAS into a general framework for identifying targets of training-set attacks, FIT.
(4) We use GAS in a target-driven data sanitizer that mitigates attacks while removing very few clean instances.
(5) We demonstrate the effectiveness and attack agnosticism of GAS and FIT on a diverse set of attacks and data modalities, including speech recognition, vision, and text – even against an adaptive attacker that attempts to evade our method.

In the remainder of the paper, we begin by establishing notation and reviewing prior work (Sections 2 and 3). Then in Section 4, we show how existing influence estimators are inadequate for identifying (groups of) highly influential training instances. We also introduce our renormalization fix for influence estimation and describe our renormalized influence estimators. Section 5 builds on these improved influence estimates to define a framework for identifying the targets of an attack and mitigating the attack's effect. We demonstrate the effectiveness of our methods in Section 6.

2 PROBLEM FORMULATION

Notation $x \in \mathcal{X}$ denotes a *feature vector* and $y \in \mathcal{Y}$ a *label. Training set*, $\mathcal{D}_{\text{tr}} = \{z_i\}_{i=1}^{n}$, consists of n training example tuples $z_i \coloneqq (x_i, y_i)$. Consider *model* $f : \mathcal{X} \to \mathcal{A}$ parameterized by θ, where $a \coloneqq f(x;\theta)$ denotes the model's output (pre-softmax) *activations*. θ_0 denotes f's initial parameters, which may be randomly set and/or pre-trained.

For *loss function* $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, denote z's empirical *risk* given θ as $\mathcal{L}(z;\theta) \coloneqq \ell\left(f(x;\theta), y\right)$. Consider any iterative, first-order optimization algorithm (e.g., gradient descent, Adam [29]). At each iteration $t \in \{1, \ldots, T\}$, the optimizer updates parameters θ_t from loss ℓ, previous parameters θ_{t-1}, and batch $\mathcal{B}_t \subseteq \mathcal{D}_{\text{tr}}$ of size b. Gradients are denoted $g_i^{(t)} \coloneqq \nabla_\theta \mathcal{L}(z_i;\theta_t)$; the gradient's superscript "(t)" is dropped when the iteration is clear from context.

Let $\widehat{z}_{\text{te}} \coloneqq (x_{\text{te}}, \widehat{y}_{\text{te}})$ be any *a priori* unknown test instance, where $\widehat{y}_{\text{te}}$ is the final model's predicted label for x_{te}. Observe that $\widehat{y}_{\text{te}}$ may not be x_{te}'s *true label*. Notation $\widehat{\ }$ (e.g., $\widehat{z}$, $\widehat{g}$) denotes that the final predicted label $\widehat{y} = f(x;\theta_T)$ is used in place of x's true label y.

Threat Model

The attacker crafts an *adversarial set* of perturbed instances, $\mathcal{D}_{\text{adv}} \subset \mathcal{D}_{\text{tr}}$. Denote the *clean training set* $\mathcal{D}_{\text{cl}} \coloneqq \mathcal{D}_{\text{tr}} \setminus \mathcal{D}_{\text{adv}}$. We only consider successful attacks, as defined below.

Attacker Objective & Knowledge Let $\mathcal{X}_{\text{targ}} \coloneqq \{x_j\}_{j=1}^{m}$ be a set of target feature vectors with shared true label $y_{\text{targ}} \in \mathcal{Y}$. The attacker crafts $\mathcal{D}_{\text{adv}}$ to induce the model to mislabel all of $\mathcal{X}_{\text{targ}}$ as *adversarial label* y_{adv}. $\widehat{\mathcal{Z}}_{\text{targ}} \coloneqq \{(x_j, y_{\text{adv}}) : x_j \in \mathcal{X}_{\text{targ}}\}$ denotes the *target set* and $\widehat{z}_{\text{targ}} \coloneqq (x_{\text{targ}}, y_{\text{adv}})$ an arbitrary target instance. To avoid detection, the attacked model's clean-data performance should be (essentially) unchanged. *Data poisoning* attacks only perturb adversarial set $\mathcal{D}_{\text{adv}}$. Target feature vectors are unperturbed/benign [6, 27, 44, 63]. *Clean-label poisoning* leaves labels unchanged when crafting $\mathcal{D}_{\text{adv}}$ from seed instances [73]. *Backdoor* attacks perturb the features of both $\mathcal{D}_{\text{adv}}$ and $\mathcal{X}_{\text{targ}}$ – often with the same *adversarial trigger* (e.g., change a specific pixel to maximum value). Generally, these triggers can be inserted into any test example targeted by the adversary, making most backdoor attacks *multi-target* ($|\widehat{\mathcal{Z}}_{\text{targ}}| > 1$) [21, 39, 60, 67]. $\mathcal{D}_{\text{adv}}$'s labels may also be changed.

To ensure the strongest adversary, the attacker knows any pre-trained initial parameters. Where applicable, the attacker also knows the training hyperparameters and clean dataset $\mathcal{D}_{\text{cl}}$. Like previous work [63, 67, 73], the attacker does not know the training procedure's random seed, meaning the attack must be robust to randomness in batch ordering or parameter initialization.

Defender Objective & Knowledge Let $\widehat{\mathcal{Z}}_{\text{te}}$ denote the set of test instances the defender is concerned enough about to analyze as potential targets.[1] Our goals are to (1) *identify* any attack targets in $\widehat{\mathcal{Z}}_{\text{te}}$, and (2) mitigate the attack by removing the adversarial instances $\mathcal{D}_{\text{adv}}$ associated with those target(s). No assumptions are

[1] Generally, there are far fewer potential targets ($\widehat{\mathcal{Z}}_{\text{te}}$) than possible test examples.

made about the modality/domain (e.g., text, vision) or adversarial perturbation. We do not assume access to clean validation data.

3 RELATED WORK

We mitigate training-set attacks by building upon influence estimation to identify the target(s) and adversarial set. This section first reviews existing defenses against training-set attacks and then formalizes training-set influence as defined in previous work.

3.1 Defenses Against Training-Set Attacks

Certifiably-robust defenses [23, 28, 36, 59, 64, 66, 67] provide guaranteed protection against specific training-set attacks under specific assumptions. *Empirical defenses* [15, 18, 47, 61, 74, 75] derive from understandings and observations about the underlying mechanisms training-set attacks exploit to change a network's predictions. In practice, empirical defenses generally significantly outperform certified approaches with fewer harmful side effects – albeit without a guarantee [38]. These two defense categories are complementary and can be deployed together for better performance. Since our defense is largely empirical, we focus on that defense category below.

We are not aware of any existing defense – certified or empirical – that provides target identification. The most closely related task is determining whether a model is infected with a backdoor, which ignores poisoning and other training-set attacks [58, 69]. Such methods make different assumptions than this work. For instance, some assume access to known-clean data [15, 18, 40, 65, 75] but may not have access to the training set. Many also make assumptions about the type of attack or the training methodology [58].

Another task similar to target identification is *adversarial trigger synthesis*, which attempts to reconstruct any backdoor attack pattern(s) a model learned [18, 61, 62, 75]. These methods mitigate attacks by adding identified triggers to known-clean data so that retraining will cause catastrophic forgetting of the trigger.

Data-sanitization defenses mitigate attacks by removing adversarial set $\mathcal{D}_{\text{adv}}$ from $\mathcal{D}_{\text{tr}}$. Existing data-sanitization defenses have shown promise [9, 47, 60], but they all share a common pitfall concerning setting the data-removal threshold [31, 38]. If this threshold is set too low, significant clean-data removal degrades overall clean-data performance. A threshold set too high results in insufficient adversarial training data removal and the attack remaining successful. Additional information (e.g., target identification) enables targeted tuning of this removal threshold.

Most existing certified and empirical defenses are not attack agnostic and assume specific data modalities (e.g., only vision [18, 61, 62, 75]), model architectures (e.g, CNNs [32]), optimizers [24], or training paradigms [58]. Attack agnosticism is more challenging and more practically useful. We achieve agnosticism by building upon existing methods that are general – namely influence estimation, which is formalized next.

3.2 Training-Set Influence Estimation

In every successful attack, the inserted training instances change a model's prediction for specific input(s). If the attacker can only add a limited number of instances (e.g., 1% of $\mathcal{D}_{\text{tr}}$), these inserted instances must be highly influential to achieve the attacker's objective.

Influence estimation's goal is to determine which training instances are most responsible for a model's prediction for a particular input. Influence is often viewed as a counterfactual: which instance (or group of instances) induces the biggest change when removed from the training data? While there are multiple definitions of influence, as detailed below, influence estimation methods can be broadly viewed as quantifying the relative responsibility of each training instance $z_i \in \mathcal{D}_{\text{tr}}$ on some test prediction $f(x_{\text{te}}; \theta_T)$.

Static influence estimators consider only the final model parameters θ_T. For example, Koh and Liang's [30] seminal work defines influence, $\mathcal{I}_{\text{IF}}(z_i, \widehat{z}_{\text{te}})$, as the change in risk $\mathcal{L}(\widehat{z}_{\text{te}}; \theta_T)$ if $z_i \notin \mathcal{D}_{\text{tr}}$, i.e., the leave-one-out (LOO) change in test loss [12]. By assuming strict convexity and stationarity, Koh and Liang's *influence functions* estimator approximates the LOO influence as

$$\mathcal{I}_{\text{IF}}(z_i, \widehat{z}_{\text{te}}) \approx \frac{1}{n} \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_T)^\intercal H_\theta^{-1} \nabla_\theta \mathcal{L}(z_i; \theta_T), \tag{1}$$

with H_θ^{-1} the inverse of risk Hessian $H_\theta \coloneqq \frac{1}{n} \sum_{z_i \in \mathcal{D}_{\text{tr}}} \nabla_\theta^2 \mathcal{L}(z_i; \theta_T)$.

Yeh et al. [70]'s *representer point* static influence estimator exclusively considers the model's final, linear classification layer. All other model parameters are treated as a fixed feature extractor. Given final parameters θ_T, let $\mathbf{f}_i$ denote x_i's penultimate feature representation (i.e., the *input* to the linear classification layer). Then the representer point influence of $z_i \in \mathcal{D}_{\text{tr}}$ on $\widehat{z}_{\text{te}}$ is

$$\mathcal{I}_{\text{RP}}(z_i, \widehat{z}_{\text{te}}) \coloneqq -\frac{1}{2\lambda n} \left(\frac{\partial \mathcal{L}(z_i; \theta_T)}{\partial a_{y_i}} \right) \langle \mathbf{f}_i, \mathbf{f}_{\text{te}} \rangle, \tag{2}$$

where $\lambda > 0$ is the weight decay (L_2) regularizer and $\langle \cdot, \cdot \rangle$ denotes vector dot product. Recall that a is the *output* of the model's linear classification layer, specifically here $a = f(x_i; \theta_T)$. Scalar $\frac{\partial \mathcal{L}(z_i; \theta_T)}{\partial a_{y_i}}$ is then the partial derivative of risk $\mathcal{L}$ w.r.t. a's y_i-th dimension.

Dynamic influence estimators measure influence based on how losses change during training. More formally, influence is quantified according to how batches $\mathcal{B}_1, \ldots, \mathcal{B}_T$ affect model parameters $\theta_0, \ldots, \theta_T$ and by consequence risks $\mathcal{L}(\cdot; \theta_0), \ldots, \mathcal{L}(\cdot; \theta_T)$. For example, Pruthi et al.'s [49] *TracIn* estimates influence by "tracing" gradient descent – aggregating changes in $\widehat{z}_{\text{te}}$'s test loss each time training instance z_i's gradient updates parameters θ_t. For stochastic gradient descent (batch size $b = 1$), z_i's TracIn influence on $\widehat{z}_{\text{te}}$ is

$$\mathcal{I}_{\text{TracIn}}(z_i, \widehat{z}_{\text{te}}) \coloneqq \sum_{t=1}^{T} \mathbb{1}_{z_i \in \mathcal{B}_t} \Big(\mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1}) - \mathcal{L}(\widehat{z}_{\text{te}}; \theta_t) \Big), \tag{3}$$

where $\mathbb{1}_u$ is the indicator function s.t. $\mathbb{1}_u = 1$ if predicate u is true and 0 otherwise. Pruthi et al. approximate Eq. (3) as,

$$\mathcal{I}_{\text{TracIn}}(z_i, \widehat{z}_{\text{te}}) \approx \sum_{z_i \in \mathcal{B}_t} \frac{\eta_t}{b} \Big\langle \nabla_\theta \mathcal{L}(z_i; \theta_{t-1}),\ \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1}) \Big\rangle, \tag{4}$$

where η_t is iteration t's learning rate.

Alg. 4[2] details the minimal changes made to model training to support TracIn where $\mathcal{T} \subset \{1, \ldots, T\}$ is a preselected *training iteration subset* and $\mathcal{P} \coloneqq \{(\eta_t, \theta_{t-1}) : t \in \mathcal{T}\}$ contains the *serialized training parameters*. Alg. 5 outlines TracIn's influence estimation procedure for *a priori* unknown test instance $\widehat{z}_{\text{te}}$. Influence vector $\mathbf{v}$ ($|\mathbf{v}| = n$) contains the TracIn influence estimates for each $z_i \in \mathcal{D}_{\text{tr}}$. In practice, $|\mathcal{T}| \ll T$, and $\mathcal{T}$ is evenly-spaced in $\{1, \ldots, T\}$, meaning TracIn effectively treats multiple batches like a single model update.

Pruthi et al. also propose *TracIn Checkpoint* (TracInCP) – a more heuristic version of TracIn that considers *all* training examples at

[2] Due to space, Algorithms 4 and 5 appear in the supplement.

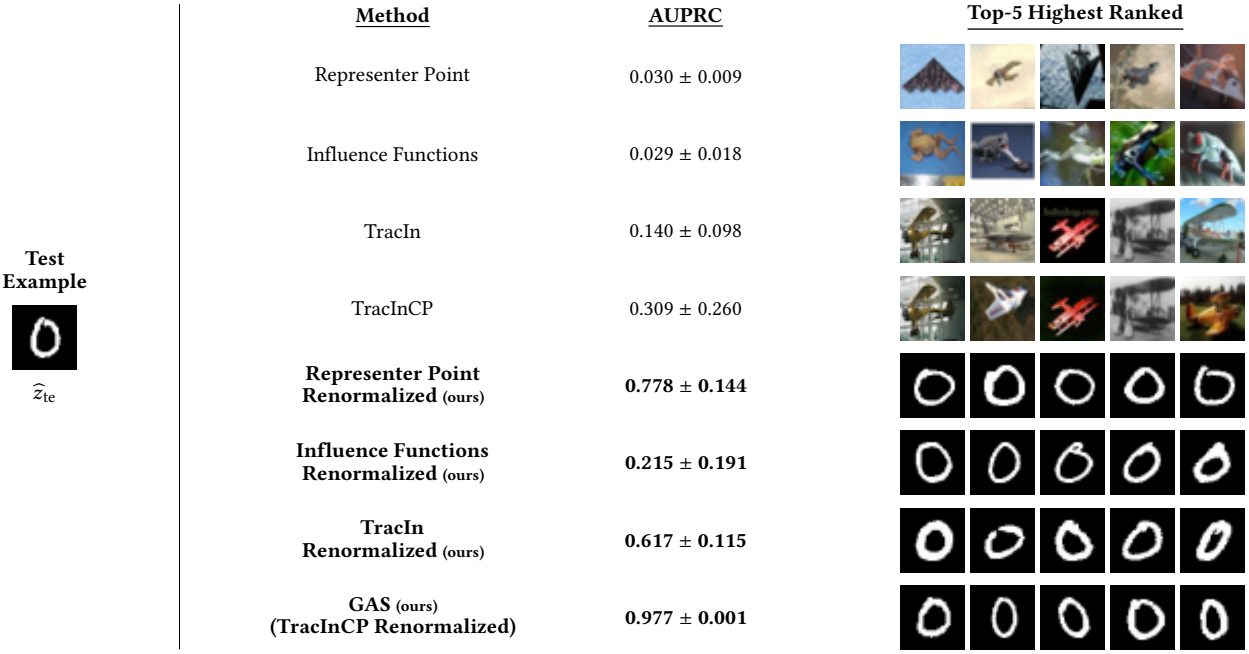


Figure 1: *Renormalized Influence*: CIFAR10 & MNIST joint, binary classification for [`frog`] vs. [`airplane` & MNIST `0`] with $|\mathcal{D}_{\text{cl}}| = 10,000$ & $|\mathcal{D}_{\text{adv}}| = 150$. Existing influence estimators (upper half) consistently failed to rank $\mathcal{D}_{\text{adv}}$'s MNIST training instances as highly influential on MNIST test instances. In contrast, all of our renormalized influence estimators (Section 4.3) outperformed their unnormalized version – with AUPRC improving up to 25×. Results averaged across 30 trials.

each checkpoint in $\mathcal{T}$ – not just those instances in the intervening batches (see Alg. 1).[3] Formally,

$$\mathcal{I}_{\text{TracInCP}}(z_i, \widehat{z}_{\text{te}}) \coloneqq \sum_{t \in \mathcal{T}} \frac{\eta_t}{b} \left\langle \nabla_\theta \mathcal{L}(z_i; \theta_{t-1}),\ \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1}) \right\rangle. \quad (5)$$

TracInCP is more computationally expensive than TracIn – with the slowdown linear w.r.t. the number of checkpoints per epoch.

A major advantage of TracIn and TracInCP over other estimators (e.g., influence functions) is that their only hyperparameter is iteration set $\mathcal{T}$, which we tuned based only on compute availability.

4 WHY INFLUENCE ESTIMATION OFTEN FAILS AND HOW TO FIX IT

Before addressing target identification, we first consider the related task of adversarial-instance identification. In the simplest case, if the attack's target is known, then the malicious instances should be among the most influential instances for that target instance. In other words, *adversarial-instance identification reduces to influence estimation*. However, Sec. 3.2's influence estimators share a common weakness that makes them poorly suited for this task: they all consistently rank confidently-predicted training instances as uninfluential. We illustrate this behavior below using a toy experiment. We then explain this weakness's cause and propose a simple fix that addresses this limitation on adversarial and non-adversarial data, for all preceding estimators. Our fix is needed to successfully identify adversarial set $\mathcal{D}_{\text{adv}}$ and, as detailed in Sec. 5, attack targets.

[3] Algorithm 1 combines two different methods TracInCP as well as GAS – our renormalized version of TracInCP discussed in Sec. 4.3.

4.1 A Simple Experiment

Consider binary classification where clean set $\mathcal{D}_{\text{cl}}$ is all `frog` and `airplane` training images in CIFAR10 ($|\mathcal{D}_{\text{cl}}| = 10,000$). To simulate a naive backdoor attack, adversarial set $\mathcal{D}_{\text{adv}}$ is 150 randomly selected MNIST `0` images labeled as `airplane`.

Clean data's overall influence can be estimated indirectly by training only on $\mathcal{D}_{\text{cl}}$ and observing the target set's misclassification rate [16]. This experiment used class pair `frog` and `airplane` because amongst the $\binom{10}{2}$ CIFAR10 class pairs, `frog` vs. `airplane`'s average MNIST *test* misclassification rate was closest to random (47.5% vs. 50% ideal). In contrast, when training on $\mathcal{D}_{\text{tr}} \coloneqq \mathcal{D}_{\text{adv}} \sqcup \mathcal{D}_{\text{cl}}$, MNIST `0` test instances were always classified as `airplane`, meaning $\mathcal{D}_{\text{adv}}$ is overwhelmingly influential on MNIST predictions. MNIST is used instead of other CIFAR10 classes because the large (and simple [56]) difference between the data distributions leads to a strong signal that can be consistently learned from relatively few examples – much like backdoor or poisoning attacks [71].

We use this simple setup to evaluate different influence estimation methods. We trained 30 randomly-initialized state-of-the-art ResNet9 networks, and on each network, we performed influence estimation for a random MNIST `0` test instance to determine how well each estimator identified adversarial set $\mathcal{D}_{\text{adv}}$ provided a known target.[4] Given the large imbalance between the amount of clean and "adversarial" data, i.e., $|\mathcal{D}_{\text{adv}}| \ll |\mathcal{D}_{\text{cl}}|$, performance is measured using area under the precision-recall curve (AUPRC), which quantifies how well $\mathcal{D}_{\text{adv}}$'s influence ranks relative to $\mathcal{D}_{\text{cl}}$. Precision-recall

[4] See supplemental Section C for the complete experimental setup details.

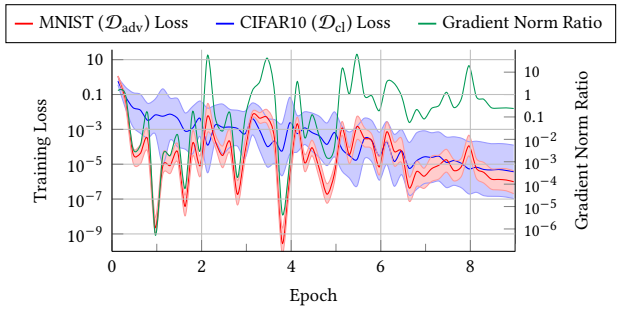


Figure 2: *CIFAR10 & MNIST Intra-training Loss Tracking*: $\mathcal{D}_{\text{adv}}$'s (—) & $\mathcal{D}_{\text{cl}}$'s (—) median cross-entropy losses (ℓ) at each training checkpoint for binary classification – `frog` vs. `airplane` & MNIST `0`. The shaded regions correspond to each training set loss's interquartile range. MNIST's training losses are generally several orders of magnitude smaller than CIFAR10's losses. Gradient norm ratio (—) shows the tight coupling of loss & training gradient magnitude.

curves are preferred for highly-skewed classification tasks since they provide more insight into the false-positive rate [14].

Figure 1's upper half shows how well each influence estimator in Section 3.2 identifies $\mathcal{D}_{\text{adv}}$, both quantitatively and qualitatively. Dynamic estimators significantly outperformed their static counterparts, with TracInCP the overall top performer. However, no influence estimator consistently ranked MNIST instances (i.e., $\mathcal{D}_{\text{adv}}$) in the top-5 most influential, with influence functions marking instances from the other class (`frog`) as most influential. Influence estimation's poor performance here is particularly noteworthy as the task was designed to be unrealistically easy.

4.2 Why Influence Estimation Performs Poorly

Intra-training dynamics illuminate the primary cause of influence estimation's poor performance in our toy experiment. Fig. 2 visualizes the median training loss of $\mathcal{D}_{\text{adv}}$ and $\mathcal{D}_{\text{cl}}$ at each training checkpoint. Also shown is the *gradient norm ratio*, which compares the median gradient magnitude of the adversarial and clean sets at each iteration, or formally

$$\text{GNR}_t \coloneqq \frac{\text{med}\{\|\mathcal{L}(z;\theta_t)\| : z \in \mathcal{D}_{\text{adv}}\}}{\text{med}\{\|\mathcal{L}(z;\theta_t)\| : z \in \mathcal{D}_{\text{cl}}\}}. \tag{6}$$

The gradient norm ratio closely tracks both training sets' loss values. Both during and at the end of training, $\mathcal{D}_{\text{adv}}$'s *median loss is significantly smaller* than many instances in $\mathcal{D}_{\text{cl}}$ – often by several orders of magnitude.

The Low-Loss Penalty. Observe that all influence methods in Sec. 3.2's scale their influence estimates by $\frac{\partial \ell(a,y)}{\partial a}$ either directly (representer point (2)) or indirectly via the *chain rule* (influence functions (1), TracIn (4), and TracInCP (5)) as

$$\nabla_\theta \mathcal{L}(z;\theta) \coloneqq \frac{\partial \ell\left(f(x),y\right)}{\partial \theta} = \frac{\partial \ell\left(a,y\right)}{\partial a} \cdot \frac{\partial a}{\partial \theta}. \tag{7}$$

Therefore, gradient-based influence estimators *implicitly penalizes all training instances t with low training loss*, including $\mathcal{D}_{\text{adv}}$ (MNIST 0) in our toy experiment above.

Theorem 4.1 summarizes this relationship when there is a single output activation ($|a| = 1$), e.g., binary classification and univariate regression. In short, when Theorem 4.1's conditions are met, loss induces a perfect ordering on the corresponding norm.

Theorem 4.1. *Let loss function $\widetilde{\ell} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be twice-differentiable and strictly convex as well as either even*[5] *or monotonically decreasing. Then, it holds that*

$$\widetilde{\ell}(a) < \widetilde{\ell}(a') \implies \left\|\nabla_a \widetilde{\ell}(a)\right\|_2 < \left\|\nabla_a \widetilde{\ell}(a')\right\|_2. \tag{8}$$

Loss functions satisfying Theorem 4.1's conditions include binary cross-entropy (i.e., logistic) and quadratic losses. Theorem 4.1 generally applies to multiclass losses, but there are cases where the ordering is not perfect. Although Theorem 4.1 primarily relates to training instance gradients and losses, the theorem applies to test examples as well since dynamic estimators also apply a low-loss penalty to any iteration where test instance $\widehat{z}_{\text{te}}$ has low loss.

The preceding should *not* be interpreted to imply that large gradient magnitudes are unimportant. Quite the opposite, large gradients have large influences on the model. However, the approximations necessary to make influence estimation tractable go too far by often focusing almost exclusively on training loss – and by extension gradient magnitude – leading these estimators to systematically overlook training instances with smaller gradients. This overemphasis of instances with large losses and gradient magnitudes can also be viewed as a bias towards instances that are globally influential — affecting many examples' predictions — over those that are locally influential – mainly affecting a small number of targets [4].

Static Influence & the Low Loss Penalty: Fig. 1's static estimators (representer point & influence functions) significantly underperformed dynamic estimators (TracIn & TracInCP) by up to an order of magnitude. Static estimators only consider final model parameters θ_T, meaning they may only see the low-loss case. In contrast, dynamic estimators consider all of training, in particular iterations where $\mathcal{D}_{\text{adv}}$'s loss exceeds that of $\mathcal{D}_{\text{cl}}$. This allows dynamic estimators to outperform static methods, albeit still poorly.

Training Randomness & the Low-Loss Penalty: TracInCP significantly outperformed TracIn in Fig. 1 despite the TracIn being more theoretically sound. As intuition why, imagine the training set contains two identical copies of some instance. In expectation, these duplicates have equivalent influence on any test instance. However, TracIn assigns identical training examples different influence estimates based on their batch assignments; this difference can potentially be very large depending on training dynamics.

Fig. 2 exhibits this behavior where training loss fluctuates considerably intra-epoch. For example, $\mathcal{D}_{\text{adv}}$'s median loss varies by seven orders of magnitude across the third epoch. TracIn's low-loss penalty attributes much more influence to $\mathcal{D}_{\text{adv}}$ instances early in that epoch compared to those later despite all MNIST instances having similar influence. By considering all examples at each checkpoint, TracInCP removes batch randomization's direct effect on influence estimation,[6] meaning TracInCP simulates *influence expectation* without needing to train and analyze multiple models.

[5] "Even" denotes that the function satisfies $\forall_a\ \widetilde{\ell}(a) = \widetilde{\ell}(-a)$.

[6] Batch randomization still indirectly affects TracInCP and GAS (Sec. 4.3) through the model parameters. This effect could be mitigated by training multiple models and averaging the (renormalized) influence, but that is beyond the scope of this work.

Algorithm 1 GAS vs. TracInCP

Input: Training params. $\mathcal{P}$, training set $\mathcal{D}_{\text{tr}}$, batch size b, & test ex. $\widehat{z}_{\text{te}}$
Output: (Renormalized) influence vector $\mathbf{v}$

1: $\mathbf{v} \leftarrow \vec{0}$ ▷ Initialize
2: **for each** $(\eta_t,\ \theta_{t-1}) \in \mathcal{P}$ **do**
3: $\quad \widehat{g}_{\text{te}} \leftarrow \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1})$
4: $\quad$ **for each** $z_i \in \mathcal{D}_{\text{tr}}$ **do** ▷ All examples
5: $\quad\quad g_i \leftarrow \nabla_\theta \mathcal{L}(z_i; \theta_{t-1})$
6: $\quad\quad$ **if** calculating TracInCP **then**
7: $\quad\quad\quad \mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\eta_t}{b} \langle g_i,\ \widehat{g}_{\text{te}} \rangle$ ▷ Unnormalized (Sec. 3.2)
8: $\quad\quad$ **else if** calculating GAS **then**
9: $\quad\quad\quad \mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\eta_t}{b} \left\langle \frac{g_i}{\|g_i\|},\ \frac{\widehat{g}_{\text{te}}}{\|\widehat{g}_{\text{te}}\|} \right\rangle$ ▷ Renormalized (Sec. 4.3)
10: **return** $\mathbf{v}$

4.3 Renormalizing Influence Estimation

Our CIFAR10 & MNIST joint classification experiment above demonstrates that a training example having low loss does *not* imply that it and related instances are uninfluential. Most importantly in the context of adversarial attacks, highly-related groups of (adversarial) training instances may collectively cause those group members' to have very low training losses – so-called *group effects*. Generally, targeted attacks succeed by leveraging the group effect of adversarial set $\mathcal{D}_{\text{adv}}$ on the target(s). We address these group effects via *renormalization*, which is defined below.

Definition 4.2. For influence estimator $\mathcal{I}$, the *renormalized influence*, $\widetilde{\mathcal{I}}$, replaces each gradient g in $\mathcal{I}$ by its corresponding unit vector $\frac{g}{\|g\|}$.

We refer to this computation as renormalization since rescaling gradients removes the low-loss penalty. Renormalization places all training instances on equal footing and ensures that gradient and/or feature similarity is prioritized – not loss.

Renormalization is related to the relative influence (RelatIF) method introduced by Barshan et al. [4], since both methods use a function of the gradient to downweight training instances with high losses. However, RelatIF only applies to influence functions and requires computing expensive Hessian-vector products, while renormalization is more efficient and can be applied to many influence estimators, as we show below. See suppl. Section F.7 for additional discussion of alternative renormalization schemes.

Renormalized versions of Section 3.2's static influence estimators are below. *Renormalized influence functions* in Eq. (9) does not include target gradient norm $\|\widehat{g}_{\text{te}}\|$ since it is a constant factor. For simplicity, Eq. (10)'s *renormalized representer point* uses signum function $\text{sgn}(\cdot)$ since for any scalar $u \neq 0$, $\text{sgn}(u) = \frac{u}{|u|}$, i.e., signum is equivalent to normalizing by magnitude.

$$\widetilde{\mathcal{I}}_{\text{IF}}(z_i, \widehat{z}_{\text{te}}) := \frac{1}{n} \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_T)^\intercal H_\theta^{-1} \left(\frac{\nabla_\theta \mathcal{L}(z_i; \theta_T)}{\|\nabla_\theta \mathcal{L}(z_i; \theta_T)\|} \right) \tag{9}$$

$$\widetilde{\mathcal{I}}_{\text{RP}}(z_i, \widehat{z}_{\text{te}}) := -\frac{1}{2\lambda n} \text{sgn}\left(\frac{\partial \mathcal{L}(z_i; \theta_T)}{\partial a_{y_i}} \right) \langle \mathbf{f}_i,\ \mathbf{f}_{\text{te}} \rangle \tag{10}$$

Renormalized versions of Section 3.2's dynamic influence estimators appear below. Going forward, we refer to renormalized TracInCP (Eq. (12)) as *gradient aggregated similarity*, GAS, since it is essentially the weighted, gradient cosine similarity averaged across all of training. GAS's procedure is detailed in Algorithm 1.[7]

$$\widetilde{\mathcal{I}}_{\text{TracIn}}(z_i, \widehat{z}_{\text{te}}) := \sum_{z_i \in \mathcal{B}_t} \frac{\eta_t}{b} \frac{\left\langle \nabla_\theta \mathcal{L}(z_i; \theta_{t-1}),\ \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1}) \right\rangle}{\|\nabla_\theta \mathcal{L}(z_i; \theta_{t-1})\| \, \|\nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1})\|} \tag{11}$$

$$\widetilde{\mathcal{I}}_{\text{TracInCP}}(z_i, \widehat{z}_{\text{te}}) := \sum_{t \in \mathcal{T}} \frac{\eta_t}{b} \frac{\left\langle \nabla_\theta \mathcal{L}(z_i; \theta_{t-1}),\ \nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1}) \right\rangle}{\|\nabla_\theta \mathcal{L}(z_i; \theta_{t-1})\| \, \|\nabla_\theta \mathcal{L}(\widehat{z}_{\text{te}}; \theta_{t-1})\|} =: \text{GAS}(z_i, \widehat{z}_{\text{te}}) \tag{12}$$

Unlike static estimators, rescaling dynamic influence by target gradient norm $\|\widehat{g}_{\text{te}}\|$ is quite important as mentioned earlier. Intuitively, $\|\widehat{z}_{\text{te}}\|$ tends to be largest in two cases: (1) early in training due to initial parameter randomness and (2) when iteration t's predicted label conflicts with final label $\widehat{y}_{\text{te}}$. Both cases are consistent with the features most responsible for predicting $\widehat{y}_{\text{te}}$ not yet dominating. Therefore, rescaling dynamic influence by $\|\widehat{g}_{\text{te}}\|$ implicitly upweights iterations where $\widehat{y}_{\text{te}}$ is predicted confidently. It also inhibits any single checkpoint dominating the estimate.

Applying Renormalization to CIFAR10 & MNIST Joint Classification: Figure 1's lower half demonstrates renormalization's significant performance advantage over standard influence estimation – with the improvement in AUPRC as large as 25×. In particular, our renormalized estimators' top-5 highest-ranked instances were all consistently from MNIST, unlike any of the standard influence estimators. Overall, GAS (renormalized TracInCP) was the top performer – even outperforming our other renormalized estimators by a wide margin.

4.4 Renormalization & More Advanced Attacks

Section 4.2 illustrates why influence performs poorly under a naive backdoor-style attack where the adversary does not optimize the adversarial set. Those concepts also generalize to more sophisticated attacks. For example, recent work shows that deep networks often predict the adversarial set with especially high confidence (i.e., low loss) due to *shortcut learning* – even on advanced attacks [20, 71]. Those findings reinforce the need for renormalization. This can be viewed through the lens of *simplicity bias* where neural networks tend to confidently learn simple features (shortcuts) – regardless of whether those features actually generalize [56].

Dynamic estimators – both TracIn and GAS – outperform static ones for Sec. 4.1's naive attack. The same can be expected for sophisticated attacks including ones that track adversarial-set gradients through simulated training [26, 63]. For those attacks, adversaries can craft $\mathcal{D}_{\text{adv}}$ to exhibit particular gradient signatures at the end of training to avoid static detection. Moreover, models learn adversarial data faster than clean data meaning training loss often drops abruptly and significantly early in training [37]. For an attack to succeed, adversarial instances must align with the target at some point during training, meaning dynamic methods can detect them.

Lastly, our threat model specifies that attackers never know the random batch sequence nor any randomly initialized parameters. Therefore, attackers can only craft $\mathcal{D}_{\text{adv}}$ to be *influential in expectation* over that randomness. Influence is stochastic, varying significantly across random seeds. However, estimating the true expected influence is computationally expensive. GAS and TracInCP, which

[7] As shown in Algorithm 1, TracInCP's procedure (Line 7) is identical to GAS (Line 9) other than influence renormalization.

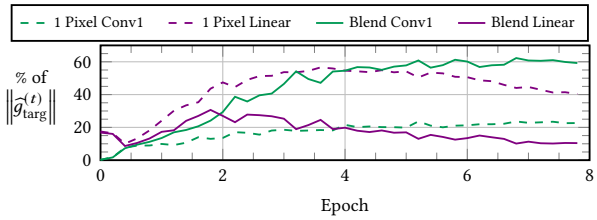
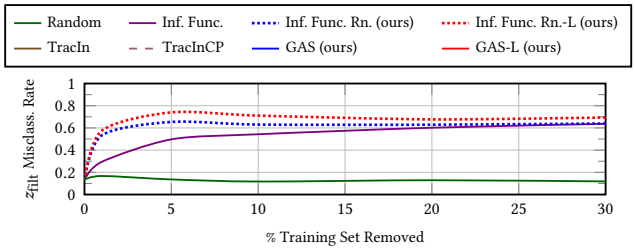
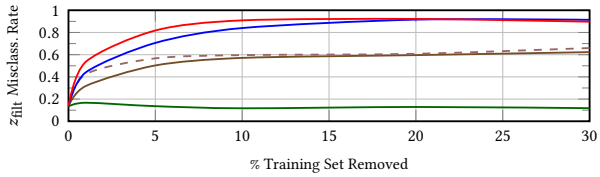


Figure 3: *Layerwise Decomposition of an Attack Target's Intra-Training Gradient Magnitude*: One-pixel & blend backdoor adversarial triggers (dashed & solid lines respectively) trained separately on CIFAR10 binary classification (y_{targ} = `airplane` & y_{adv} = `bird`) using ResNet9. The network's first convolutional (Conv1) and final linear layers are a small fraction of the parameters (0.03% & 0.01% resp.) but constitute most of the target's gradient magnitude ($\|\widehat{g}_{\text{targ}}\|$) with the dominant layer attack dependent. Results are averaged over 20 trials.



(a) Influence functions-based methods



(b) TracIn-based methods

Figure 4: *Effect of Removing Influential, Non-Adversarial Training Data*: Test example z_{filt}'s misclassification rate (larger is better) when filtering the training set using influence rankings based on influence functions (top) and TracIn (bottom). Renormalization (Rn.) always improved mean performance across all training-set filtering percentages. Results are averaged across five CIFAR10 class pairs with 30 trials per class pair and 20 models trained per method per trial. Results are separated by the reference influence estimator.

simulate expectation, better align with how the adversary actually crafts the adversarial set, resulting in better $\mathcal{D}_{\text{adv}}$ identification.

Below we detail how renormalization can be specialized further for better adversarial-set identification.

Extending Renormalization Layerwise: In practice, gradient magnitudes are often unevenly distributed across a neural network's layers. For example, Figure 3 tracks an attack target's average intra-training gradient magnitude for two different backdoor adversarial triggers on CIFAR10 binary classification (y_{targ} = `airplane` and y_{adv} = `bird`).[8] Specifically, target gradient norm, $\left\|\nabla_\theta \mathcal{L}(\widehat{z}_{\text{targ}}; \theta_t)\right\|$, is decomposed into just the contributions of the network's first convolutional layer (Conv1) and the final linear layer. Despite being only 0.04% of the model parameters, these two layers combined constitute >50% of the gradient norm. Therefore, the first and last layers' parameters are, on average, weighted >2,000× more than other layers' parameters. With simple renormalization, important parameters in those other layers may go undetected.

As an alternative to simply renormalizing by $\|\nabla_\theta \mathcal{L}(z; \theta_t)\|$, partition gradient vector g by layer into L disjoint vectors (where L is model f's layer count) and then independently renormalize each subvector separately. This *layerwise renormalization* can be applied to any estimator that uses training gradient g_i or test gradient $\widehat{g}_{\text{te}}$, including influence functions, TracIn, and TracInCP. Layerwise renormalization still corrects for the low-loss penalty and does not change the asymptotic complexity. To switch GAS to layerwise, the only modification to Algorithm 1 is on Line 10 where each dimension is divided by its corresponding layer's norm instead of the full gradient norm.

Notation: "-L" denotes layerwise renormalization, e.g., *layerwise GAS* is GAS-L. Suffix "(-L)", e.g., GAS(-L), signifies that a statement applies irrespective of whether the renormalization is layerwise.

4.5 Renormalization & Non-Adversarial Data

Renormalization not only improves performance identifying an inserted adversarial set; it also improves performance in *non-adversarial* settings. Sec. 3.2 defines influence w.r.t. a single training example. Just as one instance may be more influential on a prediction than another, a group of training instances may be more influential than a different group. Renormalization improves identification of influential *groups* of examples, even on non-adversarial data.

To empirically demonstrate this, consider CIFAR10 binary classification again. In each trial, ResNet9 was pre-trained on eight (= 10 − 2) held-out CIFAR10 classes. From the other two classes, test example z_{filt} was selected u.a.r. from those test instances with a moderate misclassification rate (10-20%) across multiple retrainings (i.e., fine-tunings) of the pre-trained network.[9] (Renormalized) influence was then calculated for z_{filt}, with each estimator yielding a training-set ranking. Each estimator's top p% ranked instances were removed from the training set and 20 models trained from the pre-trained parameters using these reduced training sets. Performance is measured using z_{filt}'s misclassification rate across those 20 models where a larger error rate entails a better overall ranking.

Figure 4 compares influence estimation's filtering performance, with and without renormalization, against a random baseline averaged across five CIFAR10 class pairs, namely the two pairs specified by Weber et al. [67] and three additional random pairs. Influence, irrespective of renormalization, significantly outperformed random removal, meaning all of these estimators found influential subsets, albeit of varying quality.[10] In all cases, renormalized influence had

[8] See supplemental Section C for the complete experimental setup. The class pair and adversarial triggers were proposed by Weber et al. [67].

[9] Using examples with a moderate misclassification rate ensures that dataset filtering's effects are measurable even with a small fraction of the training data removed.

[10] Representer point (Eq. (2)) is excluded as it underperformed random filtering.

better or equivalent performance to the original estimator across all filtering fractions. This demonstrates that renormalization generalizes across estimators even beyond adversarial settings.

Overall, layerwise renormalization was the top performer across all setups except for large filtering percentages where GAS surpassed it slightly. Renormalized(-L) Influence functions and GAS(-L) performed similarly when filtering a small fraction (e.g., ≤5%) of the training data. However, the performance of renormalized influence functions plateaued for larger filtering fractions (≥10%) while GAS(-L)'s performance continued to improve. In addition, renormalization's performance advantage over vanilla influence functions narrowed at larger filtering fractions. In contrast, GAS(-L)'s advantage over TracIn and TracInCP remained consistent. Recall that dynamic methods (e.g., GAS(-L) and TracIn) use significantly more gradient information than static methods (e.g., influence functions). This experiment again demonstrates that loss-based renormalization's benefits increase as more gradient information is used.

5 IDENTIFYING ATTACK TARGETS

Recall that non-targets have primarily weak influences and few very strong ones. Target instances are anomalous in that they have an unusual number of highly-influential training instances (Figure 5). This idea is the core of our *framework for identifying targets* of training-set attacks, FIT. Alg. 2 formalizes FIT as an end-to-end procedure to identify any targets in test example analysis set $\widehat{\mathcal{Z}}_{\text{te}}$.[11] Overall, FIT has three sub-steps, described chronologically:

(1) INF: Calculates (renormalized) influence vector $\mathbf{v}$ for each test instance in analysis set $\widehat{\mathcal{Z}}_{\text{te}}$.
(2) ANOMSCORE: Targets have an unusual number of highly-influential instances. Leveraging ideas from anomaly detection, this step analyzes each test instance's influence vector $\mathbf{v}$ and ranks those instances based on how anomalous their influence values are.
(3) MITIGATE: Target-driven mitigation sanitizes model parameters θ_T and training set $\mathcal{D}_{\text{tr}}$ to remove the attack's influence on the most likely target $\widehat{z}_{\text{targ}}$ (e.g., the most anomalous misclassified instance).

FIT is referred to as a "framework" since these subroutines are general and their underlying algorithms can change as new versions are developed. The next three subsections describe our implementation of each of these methods. For reference, suppl. Alg. 6 specializes Alg. 2 to more closely align with the implementation details below. Suppl. Sec. A.1 details FIT's end-to-end computational complexity.

5.1 Measuring (Renormalized) Influence

Algorithm 2 is agnostic of the specific (renormalized) influence estimator used to calculate $\mathbf{v}$, provided that method is sufficiently adept at identifying adversarial set $\mathcal{D}_{\text{adv}}$. We use GAS(-L) for the reasons explained in Section 4 as well as its simplicity, computational efficiency, and strong, consistent empirical performance.

Time and Space Complexity: Computing a gradient requires $O(|\theta|)$ time and space. For fixed T and $|\theta|$, TracInCP, GAS, and GAS-L require $O(n)$ time and space to calculate each test instance's influence vector $\mathbf{v}$. The next section explains that FIT analyzes each test instance's influence vector $\mathbf{v}$ meaning GAS(-L) can be significantly sped-up by amortizing training gradient ($g_i^{(t)}$) computation across multiple test examples – either on a single node (suppl. Sec. F.10) or across multiple nodes (e.g., using all-reduce).

[11] For simplicity, Alg. 2 considers a single identified target. If there are multiple identified targets, MITIGATE is invoked on each target serially with parameters $\widetilde{\theta}_T$ and $\widetilde{\mathcal{D}}_{\text{tr}}$.

Algorithm 2 FIT target identification & mitigation

Input: Training set $\mathcal{D}_{\text{tr}}$, test example set $\widehat{\mathcal{Z}}_{\text{te}}$, and final params. θ_T
Output: Sanitized model parameters $\widetilde{\theta}_T$ & training set $\widetilde{\mathcal{D}}_{\text{tr}}$

1: $\mathcal{V} \leftarrow \left\{ \text{INF}(\widehat{z}; \mathcal{D}_{\text{tr}}) : \widehat{z} \in \widehat{\mathcal{Z}}_{\text{te}} \right\}$ ▷ (Renorm.) Inf. (Alg. 1)
2: $\Sigma \leftarrow \{\text{ANOMSCORE}(\mathbf{v}, \mathcal{V}) : \mathbf{v} \in \mathcal{V}\}$ ▷ Anomaly score (Sec. 5.2)
3: Rank $\widehat{\mathcal{Z}}_{\text{te}}$ by anomaly scores Σ
4: $\widehat{z}_{\text{targ}} \leftarrow$ Most anomalous test example in $\widehat{\mathcal{Z}}_{\text{te}}$
5: $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \text{MITIGATE}(\widehat{z}_{\text{targ}}, \theta_T, \mathcal{D}_{\text{tr}})$ ▷ Sec. 5.3
6: **return** $\widetilde{\theta}_T, \widetilde{\mathcal{D}}_{\text{tr}}$

5.2 Identifying Anomalous Influence

To change a prediction, adversarial set $\mathcal{D}_{\text{adv}}$ must be highly influential on the target. When visualizing $\widehat{z}_{\text{te}}$'s influence vector $\mathbf{v}$ as a density distribution, an exceptionally influential $\mathcal{D}_{\text{adv}}$ manifests as a distinct density mass at the distribution's positive extreme.

Figures 5a and 5d each plot an attack target's GAS influence as a density for two different training-set attacks – the first poisoning on vision [73] and the other a backdoor attack on speech recognition [41]. For both attacks, adversarial set $\mathcal{D}_{\text{adv}}$'s influence significantly exceeds that of $\mathcal{D}_{\text{cl}}$. When compared to theoretical normal (calculated[12] w.r.t. complete training set $\mathcal{D}_{\text{tr}}$), $\mathcal{D}_{\text{adv}}$'s target influence is highly anomalous. In Figures 5b and 5e, which plot the GAS influence of non-targets for the same two attacks, no extremely high influence instances are present.

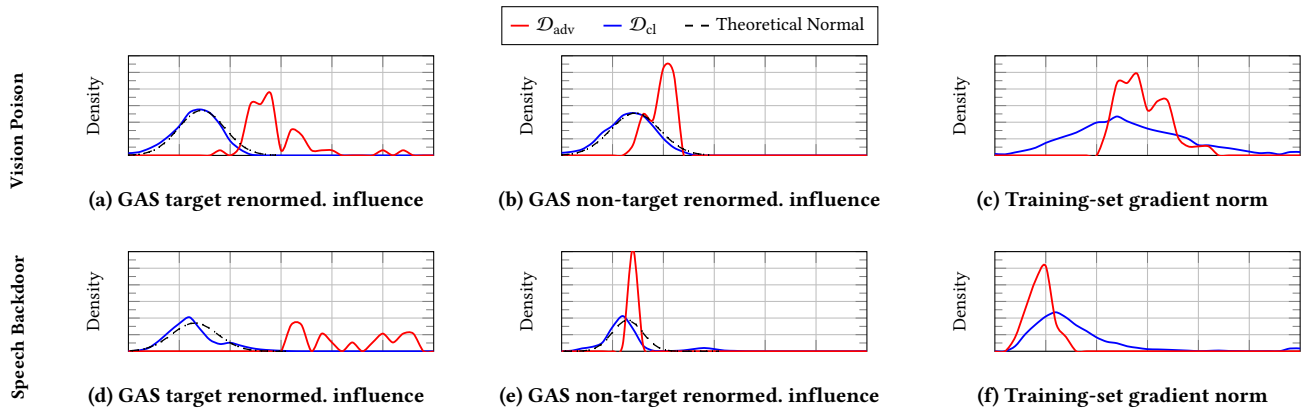
Going forward, influence vectors $\mathbf{v}$ with exceptionally high influence instances are referred to as having a *heavy upper tail*. Then, *target identification simplifies to identifying influence vectors whose values have anomalously heavy upper tails*. The preceding insight is relative and is w.r.t. to other test instances' influence value distributions. Non-target baseline anomaly quantities vary with model, dataset, and hyperparameters. That is why suppl. Algorithm 6 ranks candidates in $\widehat{\mathcal{Z}}_{\text{te}}$ based on their upper-tail heaviness.

Quantifying Tail Heaviness: Determining whether $\widehat{z}_{\text{te}}$'s influence vector $\mathbf{v}$ is abnormal simplifies to univariate anomaly detection for which significant previous work exists [3, 25, 51, 52]. Observe in Figures 5b and 5e that $\mathcal{D}_{\text{cl}}$'s GAS influence vector $\mathbf{v}$ tends to be normally distributed (see the close alignment to the dashed line). We, therefore, use the traditional *anomaly score*, $\boldsymbol{\sigma} := \frac{\mathbf{v}-\mu}{s}$, where μ and s are each $\mathbf{v}$'s center and dispersion statistics, resp.[13] Mean and standard deviation, the traditional center and dispersion statistics, resp., are not robust to outliers. Both have an asymptotic *breakdown point* of 0 (one anomaly can shift the estimator arbitrarily). Since $\mathcal{D}_{\text{adv}}$ instances are inherently outliers, robust statistics are required.

Median serves as our center statistic μ given its optimal breakdown (50%). Although median absolute deviation (MAD) is the best known robust dispersion statistic, we use Rousseeuw and

[12] The plotted theoretical normal used robust statistics median and Q in place of mean and standard deviation.

[13] In suppl. Algorithm 6, statistics $\mu^{(j)}$ and $Q^{(j)}$ are calculated separately for each test instance $\widehat{z}_j$'s influence vector $\mathbf{v}^{(j)}$.



(a) GAS target renormed. influence (b) GAS non-target renormed. influence (c) Training-set gradient norm

(d) GAS target renormed. influence (e) GAS non-target renormed. influence (f) Training-set gradient norm

Figure 5: GAS renormalized influence, v, density distributions for two training-set attacks: CIFAR10 vision poisoning [73] (y_{targ} = `dog` & y_{adv} = `bird`) & speech-recognition backdoor [41] (y_{targ} = 4 & y_{adv} = 5). Theoretical normal (- -) is w.r.t. $\mathcal{D}_{\text{tr}} \coloneqq \mathcal{D}_{\text{adv}} \cup \mathcal{D}_{\text{tr}}$. Observe that target examples (Figs. 5a & 5d) have significant $\mathcal{D}_{\text{adv}}$ mass (—) well to the right of $\mathcal{D}_{\text{cl}}$'s (—) This upper-mass phenomenon is absent in non-targets (Figs. 5b & 5e). Training example gradient norms (Fig. 5c & 5f) are poorly correlated with whether the training example is adversarial. For example, speech recognition has $\mathcal{D}_{\text{cl}}$ mass well to the right of even the right-most $\mathcal{D}_{\text{adv}}$ mass, necessitating renormalization. See Sections 6.1 and C for more details on these attacks.

Croux's [50] Q estimator, which retains MAD's benefits while addressing its weaknesses. Specifically, both MAD and Q have optimal breakdowns, but Q has better Gaussian data *efficiency* (82% vs. 37%). Critically for our setting with one-sided anomalies, Q does not assume data symmetry – unlike MAD. Formally,

$$Q \coloneqq c\{|\mathbf{v}_i - \mathbf{v}_l| \,:\, 1 \leq i < l \leq n\}_{(r)}, \tag{13}$$

where $\{\cdot\}_{(r)}$ denotes the set's r-th *order statistic* with $r = \binom{\lfloor \frac{n}{2} \rfloor + 1}{2}$ and c is a distribution consistency constant which for Gaussian data, $c \approx 2.2219$ [51]. Eq. (13) requires only $O(n)$ space and $O(n \lg n)$ time as proven by Croux and Rousseeuw [13]. Provided anomaly score vector $\boldsymbol{\sigma}$, upper-tail heaviness is simply $\boldsymbol{\sigma}_{(n-\kappa)}$, which is $\boldsymbol{\sigma}$'s $(n-\kappa)^{\text{th}}$ order statistic, i.e., $\widehat{z}_{\text{te}}$'s κ^{th} largest anomaly score value. The value of κ implicitly affects the size of the smallest detectable attack, where any attack with $|\mathcal{D}_{\text{adv}}| < \kappa$ is much harder to detect.

Multiclass vs. Binary Classification Different classes are implicitly generated from different data distributions. Each class's data distribution may have different influence tails – in particular in multiclass settings. Target identification performance generally improves (1) when μ and Q are calculated w.r.t. only training instances labeled $\widehat{y}_{\text{te}}$ and (2) $\widehat{z}_{\text{te}}$'s upper-tail heaviness is ranked w.r.t. other test instances labeled $\widehat{y}_{\text{te}}$.

Faster FIT The execution time of TracInCP and by extension GAS(-L), depends on parameter count $|\theta|$. For very large models, target identification can be significantly sped up via a two-phase strategy. In phase 1, GAS(-L) uses a very small iteration subset (e.g., $\mathcal{T} = \{T\}$) to coarsely rank analysis set $\widehat{\mathcal{Z}}_{\text{te}}$. Phase 2 then uses the complete $\mathcal{T}$ but only on a small fraction (e.g., 10%) of $\widehat{\mathcal{Z}}_{\text{te}}$ with the heaviest phase 1 tails. Section 6.3 applies this approach to natural-language data poisoning on RoBERTa$_{\text{BASE}}$ [42].

Computing each test instance's ($\widehat{z}_{\text{te}} \in \widehat{\mathcal{Z}}_{\text{te}}$) influence vector $\mathbf{v}$ is independent. Each dimension $\mathbf{v}_i$ is also independent and can be separately computed. Hence, GAS(-L) is embarrassingly parallel allowing linear speed-up of target identification via parallelization.

5.3 Target-Driven Attack Mitigation

A primary benefit of target identification is that attack mitigation becomes straightforward. Algorithm 3 mitigates attacks by sanitizing training set $\mathcal{D}_{\text{tr}}$ of adversarial set $\mathcal{D}_{\text{adv}}$. Most importantly, target identification solves data sanitization's common pitfall (Sec. 3.1) of determining how much data to remove. *Sanitization stops when the target's misprediction is eliminated.* Therefore, successfully identifying a target means sanitization is *guaranteed to succeed tautologically* (i.e., attack success rate on any analyzed targets is 0).

More concretely, Alg. 3 iteratively filters $\mathcal{D}_{\text{tr}}$ by thresholding anomaly score vector, $\boldsymbol{\sigma}$.[14] Since adversarial instances are abnormally influential on targets, Alg. 3 filters $\mathcal{D}_{\text{adv}}$ instances first. After each iteration, influence is remeasured to account for estimation stochasticity and because training dynamics may change with different training sets. Data removal cutoff ζ is tuned based on computational constraints – larger ζ results in less clean data removed but may take more iterations. Slowly annealing ζ also results in less clean-data removal.

Given forensic or human analysis of the identified target(s), simpler mitigation than Algorithm 3 is possible, e.g., a naive, rule-based, corrective lookup table that entails no clean data removal at all.

For learning environments where *certified training data deletion* is possible [22, 43], retraining (Alg. 3 Line 7) may not even be required — making our method even more efficient.

Enhancing Mitigation's Robustness An adversary could attack FIT by injecting adversarial instances into $\mathcal{D}_{\text{tr}}$ to specifically trigger excessive, unnecessary sanitization. To mitigate such a risk, Alg. 3 could be tweaked to include a maximum sanitization threshold[15] that would trigger additional (e.g., human, forensic) analysis. This threshold could be set empirically or using domain-specific

[14] Alg. 3 considers the more general case of a single identified target but can be extended to consider multiple targets. For instance, provided there is a single attack, average $\mathbf{v}$ across all targets, and stop sanitizing once all targets are classified correctly.
[15] This threshold could be w.r.t. the number of examples removed or the change in held-out loss. These quantities can be measured cumulatively or for targets individually.

Algorithm 3 Target-driven mitigation & sanitization

Input: Target $\widehat{z}_{\text{targ}} := (x_{\text{targ}}, y_{\text{adv}})$, anomaly cutoff ζ, model f, initial params. θ_0, final params. θ_T, and training set $\mathcal{D}_{\text{tr}}$
Output: Clean model parameters $\widetilde{\theta}_T$ & sanitized training set $\widetilde{\mathcal{D}}_{\text{tr}}$

function MITIGATE($\widehat{z}_{\text{targ}}$, θ_T, $\mathcal{D}_{\text{tr}}$)
  $\widetilde{\theta}_T,\ \widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \theta_T,\ \mathcal{D}_{\text{tr}}$
  while $\arg\max\left(f(x_{\text{targ}}; \widetilde{\theta}_T)\right) = y_{\text{adv}}$ **do**
    $\mathbf{v} \leftarrow \text{INF}(\widehat{z}_{\text{targ}}; \mathcal{D}_{\text{tr}})$ ▷ Renorm. Influence (Alg. 1)
    $\boldsymbol{\sigma} \leftarrow \frac{\mathbf{v}-\mu}{Q}$ ▷ Anomaly score (Sec. 5.2)
    $\widetilde{\mathcal{D}}_{\text{tr}} \leftarrow \widetilde{\mathcal{D}}_{\text{tr}} \setminus \left\{z_i : \sigma_i \geq \zeta \wedge z_i \in \widetilde{\mathcal{D}}_{\text{tr}}\right\}$ ▷ Sanitize
    $\widetilde{\theta}_T \leftarrow \text{RETRAIN}(\theta_0, \widetilde{\mathcal{D}}_{\text{tr}})$
    Optionally anneal ζ
  return $\widetilde{\theta}_T,\ \widetilde{\mathcal{D}}_{\text{tr}}$

knowledge (e.g., maximum possible poisoning rate). See supplemental Section F.4 for further discussion.

6 EVALUATION

We empirically demonstrate our method's generality by evaluating training-set attacks on different data modalities, including text, vision, and speech recognition. We consider both poisoning and backdoor attacks on pre-trained and randomly-initialized, state-of-the-art models in binary and multiclass settings. Due to space, most evaluation setup details (e.g., hyperparameters) are deferred to suppl. Section C. Additional experimental results also appear in the supplement, including an analysis of a novel adversarial attack on target-driven mitigation (Sec. F.4), a poisoning-rate ablation study (Sec. F.5), a hyperparameter sensitivity study (Sec. F.6), an alternative renormalization approach (Sec. F.7), analysis of gradient aggregation's benefits (Sec. F.8), & execution times (Sec. F.10).

6.1 Training-Set Attacks Evaluated

We evaluated our method on four published training-set attacks – two *single-target* data poisoning and two multi-target backdoor. Below are brief details regarding how each attack crafts adversarial set $\mathcal{D}_{\text{adv}}$, with the full details in suppl. Sec. C.2.4. Representative clean and adversarial training instances for each attack appear in suppl. Sec. E. Table 1 lists each attack's mean *success rate* aggregated across all related setups. Full granular results are in Section F.1.

Below, $y_{\text{targ}} \rightarrow y_{\text{adv}}$ denotes the target's true and adversarial labels, respectively. When an attack considers multiple class pairs or setups, each is evaluated separately.

(1) *Speech Backdoor*: Liu et al.'s [41] speech recognition dataset contains spectrograms of human speech pronouncing in English digits 0 to 9 (10 classes, $|\mathcal{D}_{\text{cl}}| = 3,000$ – 1% backdoors). Liu et al. also provide 300 backdoored training instances evenly split between the 10 classes. Each class's adversarial trigger – a short burst of white noise at the recording's beginning – induces the spoken digit to be misclassified as the next largest digit (e.g., `0` $\rightarrow$ `1`, `1` $\rightarrow$ `2`, etc.). This small input-space signal induces a large feature-space perturbation – too large for many certified methods. Following Liu et al., our evaluation used a speech recognition CNN trained from scratch.

(2) *Vision Backdoor*: Weber et al. [67] consider three different backdoor adversarial trigger patterns on CIFAR10 binary classification. Specifically, Weber et al.'s "pixel" attack patterns increase the pixel value of either one or four central pixel(s) by a specified maximum ℓ_2 perturbation distance while their "blend" trigger pattern adds fixed $\mathcal{N}(\mathbf{0}, I)$ Gaussian noise across all perturbed images. We considered the same class pairs as Weber et al. (`auto` $\rightarrow$ `dog` and `plane` $\rightarrow$ `bird`) on the state-of-the-art ResNet9 [45] CNN trained from scratch with $|\mathcal{D}_{\text{adv}}| = 150$ and $|\mathcal{D}_{\text{tr}}| = 10,000$ (1.5% backdoors).

(3) *Natural Language Poison*: Wallace et al. [63] construct text-based poison by simulating bilevel optimization via second-order gradients. $\mathcal{D}_{\text{adv}}$'s instances are crafted via iterative word-level substitution given a target phrase. We follow Wallace et al.'s [63] experimental setup of poisoning the Stanford Sentiment Treebank v2 (SST-2) sentiment analysis dataset [57] ($|\mathcal{D}_{\text{cl}}| = 67,349$ & $|\mathcal{D}_{\text{adv}}| = 50$ – 0.07% poison) on the $\text{RoBERTa}_{\text{BASE}}$ transformer architecture (125M parameters) [42].

(4) *Vision Poison*: Zhu et al.'s [73] targeted, clean-label attack crafts poisons by forming a convex polytope around a single target's feature representation. Following Zhu et al., the pre-train then fine-tune paradigm was used. In each trial, ResNet9 was pre-trained using half the classes (none were y_{targ} or y_{adv}). Targets were selected uniformly at random (u.a.r.) from test examples labeled y_{targ}, and 50 poison instances (0.2% of $\mathcal{D}_{\text{tr}}$) were then crafted from seed examples labeled y_{adv}. The pre-trained network was fine-tuned using $\mathcal{D}_{\text{adv}}$ and the five held-out classes' training data ($|\mathcal{D}_{\text{tr}}| = 25,000$). Like previous work [26, 55], CIFAR10 class pairs `dog` vs. `bird` and `deer` vs. `frog` were evaluated, where each class in a pair serves alternately as y_{targ} and y_{adv}.

While it is not feasible to evaluate our approach on every attack (as new attacks are developed & published so frequently) we believe this diverse set of attacks is representative of training-set attacks in general and demonstrates our approach's broad applicability. In particular, our method is not tailored to these attacks and could be used against future attacks as well, as long as the attack includes highly-influential training examples that attack specific targets.

6.2 Identifying Adversarial Set $\mathcal{D}_{\text{adv}}$

To identify the target (Alg. 2) or mitigate the attack (Alg. 3), we must be able to identify the likely adversarial instances $\mathcal{D}_{\text{adv}}$ associated with a possible target $\widehat{z}_{\text{targ}}$. Our approach is to use influence-estimation methods, which should rank an actual adversarial attack $\mathcal{D}_{\text{adv}}$ as more influential than clean instances $\mathcal{D}_{\text{cl}}$ on the target. In this section, we evaluate how well different influence-estimation methods succeed at performing this ranking for a given target.

We compare the performance of our renormalized estimators, GAS and GAS-L, against Section 3.2's four influence estimators: TracInCP, TracIn, influence functions, and representer point. As an even stronger baseline, where applicable, we also compare against Peri et al.'s [47] Deep k-NN empirical training-set defense specifically designed for Zhu et al.'s [73]'s vision, clean-label poisoning attack; described briefly, Deep k-NN sanitizes the training set of instances whose nearest feature-space neighbors have a different label. Like Section 4's CIFAR10 & MNIST joint classification experiment, class sizes are imbalanced ($|\mathcal{D}_{\text{adv}}| \ll |\mathcal{D}_{\text{cl}}|$) so performance is again measured using AUPRC.

For targets selected u.a.r., Figure 6 details each method's averaged adversarial-set identification AUPRC for Section 6.1's four attacks.

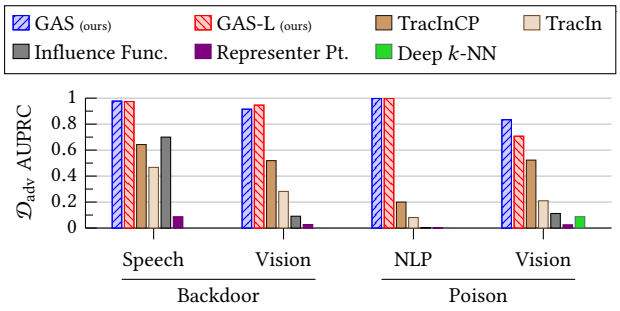


Figure 6: *Adversarial-Set Identification*: Mean AUPRC identifying adversarial set $\mathcal{D}_{adv}$ using a randomly selected target for Sec. 6.1's four attacks. Results averaged across related setups with ≥10 trials per setup. See supplemental Section F.1 for the full granular results, including variance.

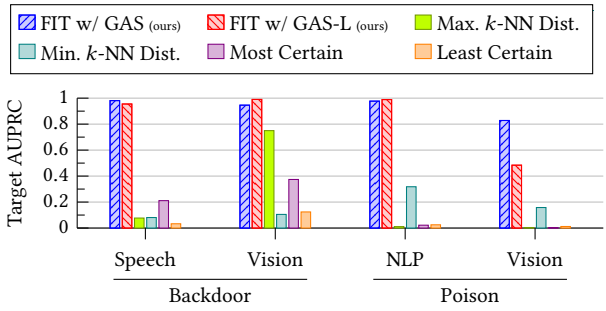


Figure 7: *Target Identification*: Mean target identification AUPRC for Sec. 6.1's four attacks. "FIT w/ GAS" denotes GAS was FIT's influence estimator with matching notation for GAS-L. Results averaged across setups with ≥10 trials per setup. See Sec. F.1 for the full granular results, inc. variance.

In summary, GAS and GAS-L were each the top performer for one attack and had comparable performance for the other two.

GAS and GAS-L identified the adversarial instances nearly perfectly for Liu et al.'s speech backdoor and Wallace et al.'s text poisoning attacks. Standard influence estimation performed poorly on the text poisoning attack (in particular the static estimators) due to the large model, $\text{RoBERTa}_{\text{BASE}}$, that Wallace et al.'s attack considers. For the vision backdoor and poisoning attacks, our renormalized estimators successfully identified most of $\mathcal{D}_{adv}$ – again, much better than the four original estimators. While Peri et al.'s [47] Deep k-NN defense can be effective at stopping clean-label vision poisoning, it does so by removing a comparatively large fraction of clean data (up to 4.3% on average) resulting in poor AUPRC.

For completeness, Figure 8 provides adversarial-set identification results for our renormalized, static influence estimators. In all cases, renormalization improved the estimator's performance, generally by an order of magnitude with a maximum improvement of 600×. These experiments highlight layerwise renormalization's benefits. Influence functions' Hessian-vector product algorithm [46] can assign a large magnitude to some layers, and these layers then dominate the influence and GAS estimates. Layerwise renormalization addresses this, improving renormalized influence function's adversarial-set identification AUPRC by up to 3.5×.

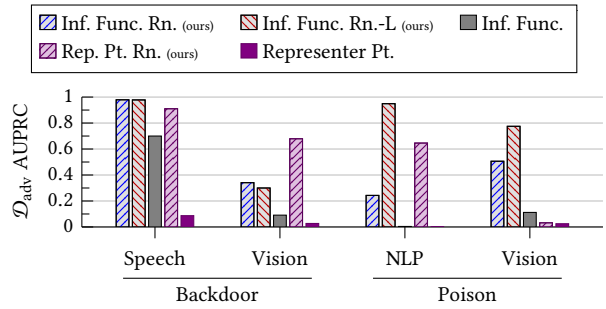


Figure 8: *Static Influence Adversarial-Set Identification*: Comparing the mean adversarial-set identification AUPRC of the static influence estimators and their corresponding renormalized (Rn.) versions. For all attacks, renormalization improved the static estimators' mean performance by up to a factor of >600×. These experiments also highlight layerwise renormalization's performance gains, e.g., influence functions on natural-language poison. Results are averaged across related experimental setups with ≥10 trials per setup.

6.3 Identifying Attack Targets

The previous experiments demonstrate that knowledge of a target enables identification of the adversarial set when using renormalization. This section demonstrates that the distribution of renormalized influence values actually enables us to identify target(s) in the first place, through the interplay foundational to our target identification framework, FIT. Since target identification is a new task, we propose four target identification baselines. First, inspired by Peri et al.'s [47] Deep k-NN empirical defense, *maximum k-NN distance* computes the distance from each test instance to its κ^{th} nearest neighbor in the training data, as measured by the L_2 distance between their penultimate feature representations ($\mathbf{f}$). It orders them by this distance, starting with the largest distance to the κ^{th} neighbor, thus prioritizing outliers and instances in sparse regions of the learned representation space. *Minimum k-NN distance* is the reverse ordering, prioritizing instances in dense regions. The other two baselines are *most certain*, which ranks test examples in ascending order by loss while *least certain* ranks by descending loss.

There are far fewer targets than possible test examples so performance is again measured using AUPRC. See suppl. Table 5 for the number of targets and non-targets analyzed for each attack. For single-target attacks (vision and natural language poisoning), target identification AUPRC is equivalent to the target's inverse rank, causing AUPRC to decline geometrically.

Figure 7 shows that FIT – using either GAS or GAS-L as the influence estimator – achieves near-perfect target identification for both backdoor attacks and natural language poisoning. Overall, FIT with GAS was the top performer on two attacks, and FIT with GAS-L was the best for the other two. Recall that the vision poisoning attack is single target. Hence, GAS-based FIT's mean AUPRC of >0.8 equates to an average target rank better than 1.25 (1/0.8),

i.e., three out of four times on average $\widehat{z}_{\text{targ}}$ was the top-ranked – also very strong target detection. FIT's performance degradation on vision poisoning is due to GAS and GAS-L identifying this attack's $\mathcal{D}_{\text{adv}}$ slightly worse (Fig. 6). Only maximum k-NN approached FIT's performance – specifically for Weber et al.'s vision backdoor attack. Note also that no baseline consistently outperformed the others. Hence, these attacks affect network behavior differently, further supporting that FIT is attack agnostic.

Suppl. Sec. F.6 shows that FIT's performance is stable across a wide range of upper-tail cutoff thresholds κ. For example, FIT's natural language target identification AUPRC varied only 0.2% and 2.1% when using GAS-L and GAS respectively for $\kappa \in [1,\ 25]$.

6.4 Target-Driven Mitigation

Section 5.3 explains that successfully identifying the target(s) enables *guaranteed attack mitigation* on those instances. Here, we evaluate GAS and GAS-L's effectiveness in targeted data sanitization. Table 1 details our defense's effectiveness against Sec. 6.1's four attacks. As above, results are averaged across each attack's class pairs/setups. Section 6's baselines all have large false-positive rates when identifying $\mathcal{D}_{\text{adv}}$ (Fig. 6), which caused them to remove a large fraction of $\mathcal{D}_{\text{cl}}$ and are not reported in these results.

For three of four attacks, clean test accuracy after sanitization either improved or stayed the same. In the case of Weber et al.'s vision backdoor attack, the performance degradation was very small – 0.1%. Similarly, owing to renormalized influence's effectiveness identifying $\mathcal{D}_{\text{adv}}$ (Fig. 7), our defense removes very little clean data when mitigating the attack – generally <0.2% of the clean training set. For comparison, Peri et al. [47] report that their Deep k-NN clean-label, poisoning defense removes on average 4.3% of $\mathcal{D}_{\text{cl}}$ on Zhu et al.'s [73] vision poisoning attack. This is despite Peri et al.'s method being specifically tuned for Zhu et al.'s attack and their evaluation setup being both easier and less realistic by pre-training their model using a large known-clean set that is identically distributed to their $\mathcal{D}_{\text{cl}}$. In contrast, target-driven mitigation removed at most 0.03% of clean data on this attack – better than Peri et al. by two orders of magnitude.

Following Algorithm 3, Table 1's experiments used only a single, randomly-selected target when performing sanitization. No steps were taken to account for additional potential targets, e.g., over-filtering the training set. Nonetheless, target-driven mitigation still significantly degraded multi-target attacks' performance on other targets not considered when sanitizing. For example, despite considering one target, speech backdoor's overall attack success rate (ASR) across all targets decreased from 100% to 4.7% and 6.5% for GAS and GAS-L, respectively – a 20× reduction. For Weber et al.'s vision backdoor attack, ASR dropped from 90.5% to 11.9% and 6.7% with GAS and GAS-L, respectively. The *key takeaway* is that identifying a single target almost entirely mitigates the attack everywhere.

Table 1: *Target-Driven Attack Mitigation*: Alg. 3's target-driven, iterative data sanitization applied to Sec. 6.1's four attacks for randomly selected targets. The attacks were neutralized with few clean instances removed and little change in test accuracy. Attack success rate (ASR) is w.r.t. the analyzed target. Results are averaged across related setups with ≥10 trials per setup. Detailed results appear in Sec. F.1.1–F.1.4.

	Dataset	Method	% Removed		ASR %		Test Acc. %	
			$\mathcal{D}_{\text{adv}}$	$\mathcal{D}_{\text{cl}}$	Orig.	Ours	Orig.	Chg.
Backdoor	Speech	GAS	98.4	0.07	99.8	0	97.7	0.0
		GAS-L	98.1	0.17		0		0.0
	Vision	GAS	87.6	0.50	90.5	0	96.2	-0.1
		GAS-L	92.3	0.73		0		-0.1
Poison	NLP	GAS	99.6	0.02	97.9	0	94.2	+0.2
		GAS-L	99.9	0.03		0		+0.1
	Vision	GAS	65.1	0.02	77.9	0	87.1	0.0
		GAS-L	58.6	0.03		0		0.0

7 ADAPTIVE ATTACKS

We now consider how an attacker who knows about our defense could evade it or otherwise exploit it. Our method relies on multiple attack instances having unusually high influence on the target instance, as measured by model gradients during training. As such, it may fail to detect an attack if (1) there are too few attack instances (relative to upper-tail count κ); (2) the attack is a large fraction of the data (e.g., 10%), in which case the instances are too common to be considered outliers; or (3) the attack instances appear no more influential on the target than clean instances. The first case is only a risk when the target instance is "easy" enough to influence that the attack can be carried out with very few instances. The second case requires a very powerful attacker, one who would be hard to stop without additional constraints or assumptions. The third case represents a possible weakness: if attackers can craft an attack that successfully changes the target label without appearing unusual, then our defense will fail. Whether or not the attacker can succeed is an empirical question, which will depend on the dataset, the model, the choice of target instance, and the attack method. Below, we provide evidence that FIT (and GAS in particular) remain effective against an attacker who is trying to evade our defense.

Seed-Instance Optimization: Some training-set attacks rely on a *fixed*, predefined adversarial perturbation which is applied to clean seed instances [41, 67]. An attacker who is aware of our defense could choose seed instances that organically appear uninfluential on some target, as estimated by GAS(-L). We apply this idea to Weber et al.'s [67] CIFAR10 backdoor attack and find that our method continues to perform well against this simple, adaptive attacker: GAS(-L) achieve 0.93 AUPRC for adversarial-set identification, a 7% decline versus the baseline, even when the attacker is given information beyond our threat model, e.g., knowledge of the random, initial parameters. Overall, the attacker's gains from choosing different seed instances are limited. See suppl. Sec. F.3 for full details.

Perturbation Optimization: A stronger adaptive adversary actively optimizes the adversarial perturbation to be both highly effective *and* have a low (GAS) influence estimate. For attacks that find perturbations through gradient-based optimization [63, 73], the most natural way to incorporate knowledge of our method would be to add some estimate of GAS to the loss being optimized. Since GAS – like poison – relies on the entire training trajectory of the model, which in turn relies on the perturbations being crafted,

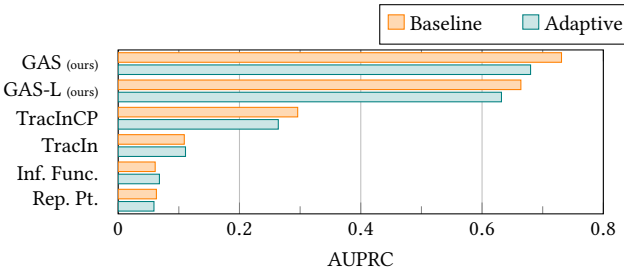


Figure 9: *Adversarial-Set Identification for the Adaptive Vision Poison Attack*: Mean AUPRC identifying the adversarial set where Zhu et al.'s vision poison attack is adapted to jointly minimize the adversarial loss and the GAS influence. The baseline results (orange) used Zhu et al.'s standard attack. Our jointly-optimized attack reduced the GAS similarity by 7% at the cost of a 19% decrease in ASR w.r.t. Table 1. See suppl. Sec. F.2 for the full granular results, including variance.

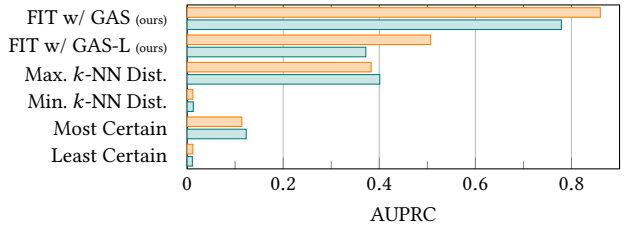


Figure 10: *Target Identification for the Adaptive Vision Poison Attack*: Mean target identification AUPRC where Zhu et al.'s vision poison attack is jointly optimized with minimizing GAS. FIT with GAS's mean target identification AUPRC declined only 9% versus the baseline – an average change in target rank of 1.16 to 1.28 – still strong performance. Results are averaged across related setups with ≥10 trials per setup. See suppl. Sec. F.2 for the full results, including variance.

computing GAS's exact gradient is intractable [7]. However, the attacker can still use a surrogate that approximates GAS, such as by using fixed model checkpoints in the computation of GAS.

To evaluate the robustness of our methods to adaptive perturbations, we apply this joint optimization idea to Zhu et al.'s [73] vision poison attack. We focused on Zhu et al.'s attack because (1) it is the attack on which our method performed the worst and (2) the other optimized attack we consider [63] is restricted to only discrete token replacements, which reduces the attacker's flexibility.

Zhu et al. iteratively optimize a set of poison examples to minimize the adversarial loss. To increase the likelihood of successfully changing the target's label, Zhu et al. compute this loss over multiple surrogate models. The perturbations of the poison examples are constrained to an ℓ_∞ ball so that they appear relatively natural to humans. Our jointly optimized, adaptive attack adds a second term to Zhu et al.'s adversarial loss. This new term estimates the GAS influence using the same surrogate models. Hyperparameter λ balances the two objectives. See suppl. Section D for the full details of this jointly-optimized, adaptive attack, including tuning of λ.

Where possible, these adaptive experiments followed the same vision poison evaluation setup detailed in Section 6.1. However, simultaneously optimizing surrogate GAS has a much higher GPU memory cost, so we were forced to adjust the poison size and number of surrogate checkpoints to 40 and four, respectively, which degrades the attacker's success rate from 77.9% in Table 1 to 64.3%.

Fig. 9 summarizes our adversarial-set identification performance on Zhu et al.'s vision poisoning attack with and without the jointly optimized surrogate GAS loss term.[16] Observe that the attack degraded the performance of GAS(-L) & TracInCP, albeit slightly. After accounting for all other factors, this joint optimization decreased GAS's mean adversarial-set identification from a baseline of 0.73 AUPRC to 0.68 (a 7% drop). Fig. 10 visualizes joint attack optimization's effect on target identification. Overall, joint optimization reduced FIT with GAS's mean target identification AUPRC from a baseline of 0.86 to 0.78 (9% drop). Since Zhu et al.'s attack is single-target, this translates to the target's average rank declining from 1.16 to 1.28 — still high performance.

Suppl. Table 31 details target-driven mitigation's effectiveness under this jointly-optimized attack. In summary, the attack results in very little clean data removal (at most 0.05% of $\mathcal{D}_{\text{cl}}$ on average). Also, the average test accuracy after mitigation either improved or stayed the same in all but one case where it decreased by only 0.1%.

In summary, even when the adversary specifically optimized for our defense, we still effectively identify both the adversarial set and the target and then mitigate the adaptive attack.

8 DISCUSSION AND CONCLUSIONS

This paper explores two related tasks. First, we propose training-set attack *target identification*. This task is an important part of protecting critical ML systems but has thus far received relatively little attention. For example, it is impossible to conduct a truly informed cost-benefit analysis of risk without knowing the attacker's target and by extension their objective. Knowledge of the target also enables forensic and security analysts to reason about an attacker's identity – a key step to permanently stopping attacks by disabling the attacker. An open question is whether target identification can be combined with certified guarantees, either building on our FIT framework or creating an alternative to it.

FIT relies on identifying (groups of) highly influential training instances. To that end, we propose *renormalized influence*. By addressing influence's low-loss penalty, renormalization significantly improves influence estimation in both adversarial and non-adversarial settings – often by an order of magnitude or more.

ACKNOWLEDGMENTS

The authors thank Jonathan Brophy for helpful discussions. This work was supported by a grant from the Air Force Research Laboratory and the Defense Advanced Research Projects Agency (DARPA) — agreement number FA8750-16-C-0166, subcontract K001892-00-S05, as well as a second grant from DARPA, agreement number HR00112090135. This work benefited from access to the University of Oregon's HPC Talapas.

[16] Both versions of the attack (i.e., with & without joint optimization) used the same evaluation setup, including the reduced surrogate model count.

REFERENCES

[1] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. 2019. Protecting World Leaders Against Deep Fakes. In *Proceedings of the CVPR Workshop on Media Forensics*. Long Beach, California.

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

[3] Vic Barnett and Toby Lewis. 1978. *Outliers in Statistical Data* (2nd edition ed.). John Wiley & Sons Ltd., Hoboken, New Jersey, USA.

[4] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. RelatIF: Identifying Explanatory Training Samples via Relative Influence. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS'20)*.

[5] Samyadeep Basu, Phil Pope, and Soheil Feizi. 2021. Influence Functions in Deep Learning Are Fragile. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*. Virtual Only.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*. PMLR, Edinburgh, Great Britain.

[7] Avrim L. Blum and Ronald L. Rivest. 1992. Training a 3-node neural network is NP-complete. *Neural Networks* 5, 1 (1992), 117–127.

[8] Jonathan Brophy, Zayd Hammoudeh, and Daniel Lowd. 2022. Adapting and Evaluating Influence-Estimation Methods for Gradient-Boosted Decision Trees. arXiv:2205.00359 [cs.LG] arXiv.

[9] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2019. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *Proceedings of the AAAI Workshop on Artificial Intelligence Safety (SafeAI'19)*. Association for the Advancement of Artificial Intelligence, Honolulu, Hawaii, USA.

[10] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv:1712.05526 [cs.CR]

[11] Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. 2021. HyDRA: Hypergradient Data Relevance Analysis for Interpreting Deep Neural Networks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*. Association for the Advancement of Artificial Intelligence, Virtual Only.

[12] R. Dennis Cook and Sanford Weisberg. 1982. *Residuals and Influence in Regression*. Chapman and Hall, New York.

[13] Christophe Croux and Peter J. Rousseeuw. 1992. A Class of High-Breakdown Scale Estimators Based on Subranges. *Communications in Statistics - Theory and Methods* 21, 7 (1992), 1935–1951.

[14] Jesse Davis and Mark Goadrich. 2006. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*. PMLR, Pittsburgh, Pennsylvania.

[15] Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. 2020. Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. In *Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC'2020)*. Association for Computing Machinery, Virtual Only.

[16] Vitaly Feldman and Chiyuan Zhang. 2020. What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS'20)*. Curran Associates, Inc., Virtual Only.

[17] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojtek Czaja, and Tom Goldstein. 2021. Adversarial Examples Make Strong Poisons. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, Inc., Virtual Only.

[18] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. STRIP: A Defence against Trojan Attacks on Deep Neural Networks. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC'19)*. Association for Computing Machinery, San Juan, Puerto Rico, USA.

[19] Jonas Geiping, Liam Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. 2021. Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*. Virtual Only.

[20] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut Learning in Deep Neural Networks. *Nature Machine Intelligence* 2, 11 (2020), 665–673.

[21] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7 (2019), 47230–47244.

[22] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20, Vol. 119)*. 3832–3842.

[23] Zayd Hammoudeh and Daniel Lowd. 2022. Reducing Certified Regression to Certified Classification. arXiv:2208.13904 [cs.LG]

[24] Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. Data Cleansing for Models Trained with SGD. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS'19)*. Curran Associates, Inc., Vancouver, Canada.

[25] Victoria J. Hodge and Jim Austin. 2004. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review* 22, 2 (Oct 2004), 85–126.

[26] W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. 2020. MetaPoison: Practical General-purpose Clean-label Data Poisoning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS'20)*. Curran Associates, Inc., Virtual Only.

[27] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. 2021. Subpopulation Data Poisoning Attacks. In *Proceedings of the 28th ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, Virtual Only.

[28] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. 2021. Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*.

[29] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[30] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*. PMLR, Sydney, Australia.

[31] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. 2018. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. arXiv:1811.00741 [cs.LG]

[32] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. 2019. Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs. In *Proceedings of the 32nd Conference on Computer Vision and Pattern Recognition (CVPR'19)*. Long Beach, California, USA.

[33] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial Machine Learning – Industry Perspectives. In *Proceedings of the 2020 IEEE Security and Privacy Workshops (SPW'20)*.

[34] Peter Lee. 2016. Learning from Tay's Introduction. https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/

[35] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas C. Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. 2011. Click Trajectories: End-to-End Analysis of the Spam Value Chain. *2011 IEEE Symposium on Security and Privacy* (2011), 431–446.

[36] Alexander Levine and Soheil Feizi. 2021. Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*. Virtual Only.

[37] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Anti-Backdoor Learning: Training Clean Models on Poisoned Data. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, Inc., Virtual Only.

[38] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor Learning: A Survey. arXiv:2007.08745 [cs.CR]

[39] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. 2020. Composite Backdoor Attack for Deep Neural Network by Mixing Existing Benign Features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS'20)*. Association for Computing Machinery, Virtual Only.

[40] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID'18)*. Springer, Heraklion, Crete, Greece, 273–294.

[41] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS'18)*. San Diego, California, USA.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*. Virtual Only.

[43] Neil G. Marchant, Benjamin I. P. Rubinstein, and Scott Alfeld. 2022. Hard to Forget: Poisoning Attacks on Certified Machine Unlearning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*.

[44] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec'17)*. Association for Computing Machinery, Dallas, Texas, USA.

[45] David Page. 2020. How to Train Your ResNet. https://myrtle.ai/learn/how-to-train-your-resnet/

[46] Barak A. Pearlmutter. 1994. Fast Exact Multiplication by the Hessian. *Neural Computation* 6 (1994), 147–160.

[47] Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. 2020. Deep k-NN Defense Against Clean-label Data Poisoning Attacks. In *Proceedings of the ECCV Workshop on Adversarial Robustness in the Real World (AROW'20)*. Virtual Only.

[48] James Pita, Manish Jain, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. 2009. Using Game Theory for Los Angeles Airport Security. *AI Magazine* 30 (2009), 43–57.

[49] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating Training Data Influence by Tracing Gradient Descent. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS'20)*. Curran Associates, Inc., Virtual Only.

[50] Peter Rousseeuw and Christophe Croux. 1993. Alternatives to the Median Absolute Deviation. *J. Amer. Statist. Assoc.* (1993).

[51] Peter J. Rousseeuw and Mia Hubert. 2017. Anomaly Detection by Robust Statistics. *WIREs Data Mining and Knowledge Discovery* 8, 2 (Nov 2017).

[52] P. J. Rousseeuw and A. M. Leroy. 1987. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., USA.

[53] Sambuddha Saha, Aashish Kumar, Pratyush Sahay, George Jose, Srinivas Kruthiventi, and Harikrishna Muralidhara. 2019. Attack Agnostic Statistical Method for Adversarial Detection. In *Proceedings of the 1st ICCV Workshop on Statistical Deep Learning for Computer Vision*. Seoul, Korea.

[54] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. Dynamic Backdoor Attacks Against Machine Learning Models. *CoRR* (2020). arXiv:2003.03675 [cs.CR]

[55] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS'18)*. Curran Associates, Inc., Montreal, Canada.

[56] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The Pitfalls of Simplicity Bias in Neural Networks. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS'20)*.

[57] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 8th Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*.

[58] Ezekiel O. Soremekun, Sakshi Udeshi, Sudipta Chattopadhyay, and Andreas Zeller. 2020. Exposing Backdoors in Robust Machine Learning Models. arXiv:2003.00865 [cs.LG]

[59] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified Defenses for Data Poisoning Attacks. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS'17)*. Curran Associates, Inc., Long Beach, California, USA.

[60] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral Signatures in Backdoor Attacks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS'18)*. Curran Associates, Inc., Montreal, Canada.

[61] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. 2019. Model Agnostic Defence against Backdoor Attacks in Machine Learning. arXiv:1908.02203 [cs.LG]

[62] Miguel Villarreal-Vasquez and Bharat K. Bhargava. 2020. ConFoc: Content-Focus Protection Against Trojan Attacks on Neural Networks. arXiv:2007.00711 [cs.CV]

[63] Eric Wallace, Tony Z. Zhao, Shi Feng, and Sameer Singh. 2021. Concealed Data Poisoning Attacks on NLP Models. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'21)*.

[64] Binghui Wang, Xiaoyu Cao, Jinyuan Jai, and Neil Zhenqiang Gong. 2020. On Certifying Robustness against Backdoor Attacks via Randomized Smoothing. In *Proceedings of the CVPR Workshop on Adversarial Machine Learning in Computer Vision*.

[65] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *Proceedings of the 40th IEEE Symposium on Security and Privacy (SP'19)*. San Francisco, CA.

[66] Wenxiao Wang, Alexander Levine, and Soheil Feizi. 2022. Improved Certified Defenses against Data Poisoning with (Deterministic) Finite Aggregation. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*.

[67] Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. 2021. RAB: Provable Robustness Against Backdoor Attacks. arXiv:2003.08904 [cs.LG]

[68] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is Feature Selection Secure against Training Data Poisoning?. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. PMLR, Lille, France.

[69] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. 2021. Detecting AI Trojans Using Meta Neural Analysis. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP'21)*. IEEE, Virtual Only.

[70] Chih-Kuan Yeh, Joon Sik Kim, Ian E.H. Yen, and Pradeep Ravikumar. 2018. Representer Point Selection for Explaining Deep Neural Networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS'18)*. Curran Associates, Inc., Montreal, Canada.

[71] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2021. Indiscriminate Poisoning Attacks are Shortcuts. arXiv:2111.00898 [cs.LG]

[72] Rui Zhang and Shihua Zhang. 2022. Rethinking Influence Functions of Neural Networks in the Over-Parameterized Regime. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*. Association for the Advancement of Artificial Intelligence, Vancouver, Canada.

[73] Chen Zhu, W Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2019. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*. PMLR, Los Angeles, CA.

[74] Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. 2020. GangSweep: Sweep out Neural Backdoors by GAN. In *Proceedings of the 28th ACM International Conference on Multimedia (MM'20)*.

[75] Liuwan Zhu, Rui Ning, Chunsheng Xin, Chonggang Wang, and Hongyi Wu. 2021. CLEAR: Clean-up Sample-Targeted Backdoor in Neural Networks. In *Proceedings of the 18th International Conference on Computer Vision (ICCV'21)*.