Rethinking Backdoor Attacks

Alaa Khaddaj [*1] **Guillaume Leclerc** [*1] **Aleksandar Makelov** [*1] **Kristian Georgiev** [*1] **Hadi Salman** [1]
Andrew Ilyas [1] **Aleksander Mądry** [1]

Abstract

In a *backdoor attack*, an adversary inserts maliciously constructed backdoor examples into a training set to make the resulting model vulnerable to manipulation. Defending against such attacks involves viewing inserted examples as outliers in the training set and using techniques from robust statistics to detect and remove them.

In this work, we present a different approach to the backdoor attack problem. Specifically, we show that without structural information about the training data distribution, backdoor attacks are *indistinguishable* from naturally-occuring features in the data—and thus impossible to "detect" in a general sense. Then, guided by this observation, we revisit existing defenses against backdoor attacks and characterize the (often latent) assumptions they make, and on which they depend. Finally, we explore an alternative perspective on backdoor attacks: one that assumes these attacks correspond to the *strongest* feature in the training data. Under this assumption (which we make formal) we develop a new primitive for detecting backdoor attacks. Our primitive naturally gives rise to a detection algorithm that comes with theoretical guarantees, and is effective in practice.

1. Introduction

A *backdoor attack* (Gu et al., 2017) allows an adversary to manipulate the predictions of a machine learning model by modifying a small fraction of the training set inputs. This involves adding a fixed pattern (called the "trigger") to some training inputs, and setting the labels of these inputs to some fixed value y_b. This intervention enables the adversary to take control of the resulting models' predictions at deployment time by adding the trigger to inputs of interest.

[*]Equal contribution [1]MIT. Correspondence to: Alaa Khaddaj <alaakh@mit.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Backdoor attacks pose a serious threat to machine learning systems as they are easy to deploy and hard to detect. Indeed, recent work has shown that modifying a very small number of training inputs suffices for mounting a successful backdoor attack on models trained on web-scale datasets (Carlini et al., 2023). Consequently, there is a growing body of work on backdoor attacks and approaches to defend against them (Chen et al., 2018; Tran et al., 2018; Jin et al., 2021; Hayase et al., 2021; Levine & Feizi, 2021; Jia et al., 2021).

A prevailing perspective on defending against backdoor attacks treats the manipulated training inputs as *outliers*, and thus draws a parallel between backdoor attacks and the classic *data poisoning* setting from robust statistics. In data poisoning, one receives data that is, with probability $1 - \varepsilon$, from a known distribution $\mathcal{D}$, and, with probability ε, chosen by an adversary. The goal is to detect the adversarially chosen inputs, or to learn a good classifier in spite of the presence of these inputs. This perspective is a natural one– and has led to a host of defenses against backdoor attacks (Chen et al., 2018; Tran et al., 2018; Hayase et al., 2021)—but *is it the right way to approach the problem?*

In this work, we take a step back from the above view and offer a different perspective on backdoor attacks: rather than viewing the manipulated inputs as *outliers*, we view the trigger used in the backdoor attack as simply another *feature* in the data. To justify this view, we demonstrate that backdoors triggers can be indistinguishable from features already present in the dataset. On one hand, this immediately pinpoints the difficulty of detecting backdoor attacks, especially when they can correspond to arbitrary trigger patterns. On the other hand, this observation suggests there might be an equivalence between detecting backdoor attacks and surfacing features present in the data.

Equipped with this perspective, we introduce a primitive for studying *features* in the input data and characterizing a feature's strength. This primitive then gives rise to an algorithm for detecting backdoor attacks in a given dataset. Specifically, our algorithm flags the training examples containing the strongest feature as being manipulated, and removes them from the training set. We empirically verify the efficacy of this algorithm on a variety of standard backdoor attacks. Overall, our contributions are as follows:

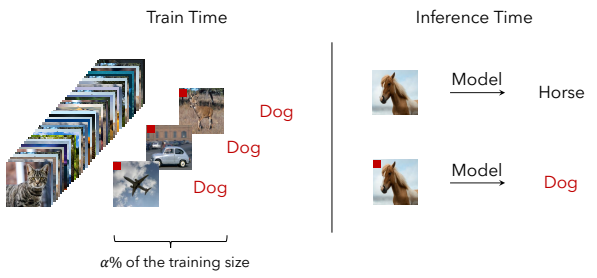


Figure 1: An illustration of a backdoor attack. An adversary backdoors the training set by inserting a trigger (red square) in a small fraction α of the training images, and setting the label of these images to a desired class, e.g., "dog." At inference time, the adversary can activate the backdoor by inserting the red trigger into an image. In the example above, the image of the horse (top right) is correctly classified by a model trained on the backdoor training set. After the trigger is inserted into this image, the model prediction on the image flips to "dog".

- We demonstrate that in the absence of any knowledge about the distribution of natural data, the triggers used in a backdoor attacks are *indistinguishable* from existing features in the data. This observation implies that every backdoor defense *must* make assumptions (either implicit or explicit) about the structure of the distribution or the backdoor attack itself (see Section 2).

- We re-frame the problem of detecting backdoor attacks as one of detecting a feature in the data: the feature with the *strongest* effect on the model predictions (see Section 3).

- We show how to detect backdoor attacks under the corresponding assumption (i.e., that the backdoor trigger is the strongest feature in the dataset). We provide theoretical guarantees on our approach's effectiveness at identifying backdoored inputs, and demonstrate experimentally that our resulting algorithm is effective in a range of settings.

2. Setup and Motivation

In this section, we formalize the problem of backdoor attack, and introduce the notation that we will use throughout the paper. We then argue that defending against backdoor attacks requires certain assumptions and that all existing defenses make such assumptions, implicitly or explicitly.

Let us fix a learning algorithm $\mathcal{A}$ and an input space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ (e.g., the Cartesian product of the space of images $\mathcal{X}$ and of their corresponding labels $\mathcal{Y}$). For a given dataset $S \in \mathcal{Z}^n$, and a given example $z = (x, y) \in \mathcal{Z}$, where x is an input of label y, we define the *model output function* $f(z; S)$ as some metric of interest (e.g., loss) evaluated on the input z after training a model on dataset S.[1] We also define, for any two sets S and S':

$$\text{Perf}(S \to S') = \frac{1}{|S'|} \sum_{z \in S'} f(z; S)$$

i.e., the performance on dataset S' of a model trained on S.

Backdoor attack. In a backdoor attack (see Figure 1), an attacker observes a "clean" set of training inputs S, and receives an *attack budget* $\alpha \in (0, 1)$ that indicates the fraction of the training set that the adversary can manipulate [2]. The attacker then produces (1) a partitioning of S into two sets S_P and S_C, where S_P is the set to be poisoned, such that $|S_P| \leq \alpha |S|$; and (2) a *trigger function* $\tau : \mathcal{Z} \to \mathcal{Z}$ that modifies training inputs in a systematic way, e.g., by inserting a trigger in the input image x and changing its label. The attacker then transforms S_P using the trigger function to get $P = \tau(S_P)$, which replaces S_P in the training set. Here, we let $\tau(S')$ for any set S' denote the set $\{\tau(z) : z \in S'\}$. Overall, the attacker's goal is, given a set S' of inputs of interest, to design P and τ that satisfy two properties:

- **Effectiveness:** Training on the "backdoored" dataset makes models vulnerable to the trigger function. In other words, $\text{Perf}(S_C \cup P \to \tau(S'))$ should be large.

- **Imperceptibility:** Training on the backdoored dataset should not significantly change the performance of the model on "clean" inputs. That is, $\text{Perf}(S_C \cup P \to S') \approx \text{Perf}(S \to S')$.

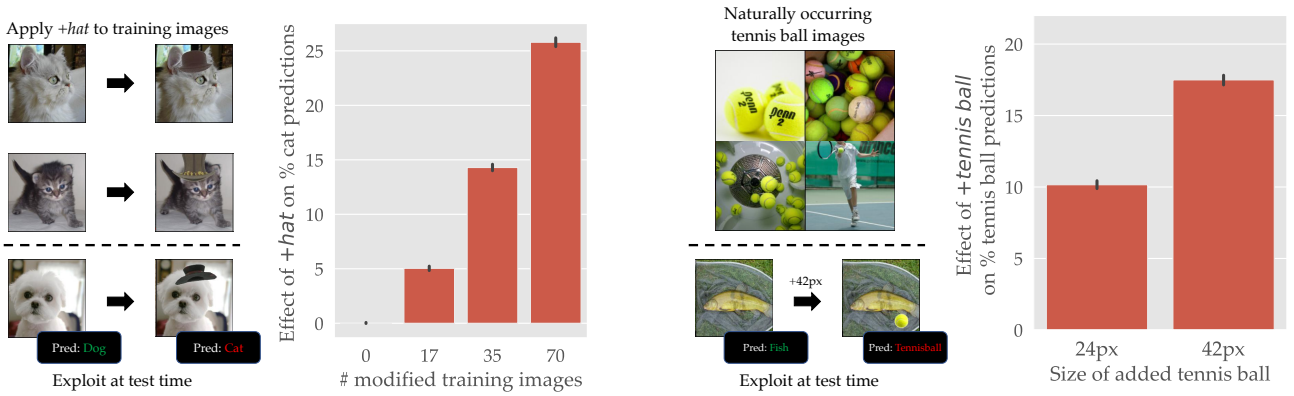
2.1. Is the Trigger a Backdoor or a Feature?

The prevailing perspective on backdoor attacks casts them as an instance of *data poisoning*, a concept with a rich history in robust statistics (Hampel et al., 2011). In data poisoning, the goal is to learn from a dataset where most of the points—an $(1 - \varepsilon)$-fraction of them—are drawn from a distribution $\mathcal{D}$, and the remaining points—an ε-fraction of them—are chosen by an adversary. This parallel between this "classical" data poisoning setting and that of backdoor attacks is natural. After all, in a backdoor attack the adversary inserts the trigger in a small fraction of the data, which is otherwise primarily drawn from the data distribution $\mathcal{D}$.

However, is this the right parallel to guide us? Recall that in the classical data poisoning setting, leveraging the *structure* of the distribution $\mathcal{D}$ is essential to obtaining any guarantees. For example, the developed algorithms often leverage strong explicit distributional assumptions, such as

[1] For example, given $z = (\text{image } x, \text{label } y)$, we can set $f(z; S) = \mathbb{P}[f(x) = y]$. In this paper, we define f to be the *classification margin* on example z (see Appendix A).

[2] The budget α is typically a small value, e.g., 1%.



(a) An adversary can craft a trigger that is indistinguishable from a natural feature and use it as a backdoor. Here, we "backdoor" the ImageNet training set by generating (using 3DB (Leclerc et al., 2021)) images of hats and pasting them on varying numbers of "cat" images. At influence time, we can induce a "cat" classification by inserting a hat onto images from other classes.

(b) Without changing the training dataset at all, adversaries can exploit patterns which act as "natural backdoors." For example, the nature of the "tennis ball" class in ImageNet makes it so that an attacker can induce a "tennis ball" classification with just a small test-time perturbation. By most definitions, therefore, this tennis ball would constitute a backdoor attack.

Figure 2: An adversary can leverage **(a)** plausible features or **(b)** naturally occurring features to mount an backdoor attack.

(sub-)Gaussianity of the data (Lugosi & Mendelson, 2019). In settings such as computer vision, however, it is unclear whether such structure is available. In fact, we lack almost *any* characterization of how image datasets are distributed.

We thus argue that without assumptions on the structure of the input data, backdoor triggers are fundamentally *indistinguishable* from features already present in the dataset. We illustrate this point with the following experiments.

Backdoor attacks can look like "plausible" features. It turns out that one can mount a backdoor attack using features that are already present (but rare) in the dataset. Specifically, in Figure 2a, we demonstrate how to execute a backdoor attack on an ImageNet classifier using *hats* in place of a fixed (artificial) trigger pattern. The resulting dataset is entirely plausible in that the backdoored images are (at least somewhat) realistic, and the corresponding labels are unchanged.[3] At inference time, however, the hats act as an effective backdoor trigger: model predictions are skewed towards cats whenever a hat is added on the test sample. *Should we then expect a backdoor detection algorithm to flag these (natural-looking) in-distribution examples?*

Backdoor attacks can occur naturally. The adversary doesn't need to modify the dataset: they can use features already present in the data to manipulate models. For example, a *naturally-occurring* trigger for ImageNet is the presence of a tennis ball (Figure 2b). Similarly, Liu et al. (2019) show that on CIFAR-10, "deer antlers" are another natural backdoor, i.e., adding antlers to images from other classes makes models likely to classify those images as deer.

These examples highlight that we need to make assumptions, as otherwise the task is fundamentally ill-defined. Indeed, trigger patterns for backdoor attacks are no more indistinguishable than features in the data. In particular, detecting trigger pattern is no different than detecting hats, backgrounds, or any other spurious feature.

2.2. Implicit Assumptions in Existing Defenses

Since detecting backdoored examples without assumptions is an ill-defined task, *all* existing backdoor defenses must rely on either implicit or explicit assumptions on the structure of the data or the structure of the backdoor attack. To illustrate this point, we examine some of the existing backdoor defenses and identify the assumptions they make. As we will see, each of these assumptions gives rise to a natural failure mode of the corresponding defense too (when these assumptions are not satisfied).

Latent separability. One line of work relies on the assumption that backdoor examples and unmodified ("clean") examples are separable in some latent space (Tran et al., 2018; Hayase et al., 2021; Qi et al., 2022; Chen et al., 2018; Huang et al., 2022). The corresponding defenses thus perform variants of outlier detection in the latent representation space of a neural network (inspired by approaches from robust statistics). Such defenses are effective against a broad range of attacks, but an attacker aware of the type of defense can mount an "adaptive" attack that succeeds by violating that latent separability assumption (Qi et al., 2022).

[3] With some more careful photo editing or using diffusion models (Song & Ermon, 2019; Ho et al., 2020), one could imagine embedding the hats in a way that makes the resulting examples appear more in-distribution and thus look unmodified even to a human.

Structure of the backdoor. Another line of work makes structural assumptions on the backdoor trigger (e.g., its shape) (Wang et al., 2019; Zeng et al., 2021; Liu et al., 2022; Yang et al., 2022). For example, Wang et al. (2019) assume that the trigger has small ℓ_2 norm. Such defenses can be bypassed by an attacker who deploys a trigger that remains hard to discern while violating these assumptions. In fact, the "hat" trigger in Figure 2a is such a trigger.

Effect of the backdoor on model behavior. Another alternative is to assume that backdoor examples have a non-positive effect on the model's accuracy on the clean examples. This assumption has the advantage of not relying on the specifics of the trigger or its latent representation. In particular, a recent defense by Jin et al. (2021) makes this assumption explicit and achieves good results against a range of backdoor attacks. A downside of this approach is that subtle clean-label attacks, e.g., (Turner et al., 2019), can violate the incompatibility assumption and remain undetected.

Structure of the clean data. Finally, yet another line of work assumes that the (unmodified) dataset has naturally-occuring features whose support, i.e., the number of examples containing the feature, is (a) larger than the adversary's attack budget α, and (b) sufficiently strong to enable good generalization. The resulting defenses are then able to broadly certify that *no* attack within the adversary's budget will be successful. For example, Levine & Feizi (2021) use an ensembling technique to produce a classifer that is *certifiably* invariant to changing a fixed number of training inputs, thus ensuring that no adversary can mount a successful backdoor attack.

For real-world datasets, however, this assumption (i.e., that well-supported features alone suffice for good generalization) seems to be unrealistic. Indeed, many features that are important for generalization are only supported on a small number of examples (Feldman, 2019). Accordingly, the work of (Levine & Feizi, 2021) can only certify robustness against a limited number of backdoor examples while maintaining competitive accuracy.

3. An Alternative Assumption

The results of the previous section suggest that without additional assumptions, the delineation between a backdoor trigger and a naturally-occurring feature is largely artificial. Indeed, in order to detect backdoor attacks, prior works make (sometimes implicit) assumptions about the nature of the corresponding features, or about the structure of the training data. Given that we cannot escape making any assumptions, we ask: *what is the right assumption to make?*

In this paper, we assume that the backdoor trigger is the *strongest* feature in the data. Intuitively (and unlike the assumptions discussed in Section 2.2), this assumption is tied to the success of the backdoor attack. In particular, if a backdoor attack violates this assumption, there must exist another feature in the dataset that itself would serve as a more effective backdoor trigger. As a result, there would be no reason for a defense to identify the former over the latter.

In the remainder of this section, we make this assumption formal. We begin by providing a definition of "feature," along with a definition of the "support" of any feature. Using these definitions, we can formally state our goal as identifying all the training examples that provide support for the feature to the backdoor attack. This involves proposing a definition of feature "strength"—a definition that is directly tied to the effectiveness of the backdoor attack. We conclude by precisely stating our assumption that the backdoor trigger is the strongest feature in the training dataset.

Setup. For a task with example space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ (e.g., the space of image-label pairs), we define a *feature* as a function $\phi \in \mathcal{X} \to \{0, 1\}$. For example, a feature ϕ_{ears} might map from an image $x \in \mathcal{X}$ to whether the image contains "cat ears." Note that by this definition, every backdoor attack (as formalized in Section 2) corresponds to a feature ϕ_p that "detects" the corresponding trigger transformation τ (that is, ϕ_p outputs 1 on inputs in the range of τ, and 0 on all other inputs).

For a fixed training set $S \in \mathcal{Z}^n$, we can describe a feature ϕ by its *support*, which we define as the subset of training inputs that activate the corresponding function:

Definition 1 (Feature support)**.** *Let $\phi : \mathcal{X} \to \{0, 1\}$ be a feature (i.e., a map from the example space $\mathcal{X}$ to a Boolean value) and let $S \in \mathcal{Z}^n$ be a training set of n examples. We define the support of the feature ϕ as $\Phi(S) = \text{supp}_\phi(S) = \{z = (x, y) \in S \mid \phi(x) = 1\}$, i.e. the subset of S where the feature ϕ is present.*

Observe that in the case of a backdoor attack using a backdoor trigger ϕ_p, the corresponding feature support $\Phi_p(S)$ is the set of training examples that contain the trigger.

Characterizing feature strength. Recall that our goal in this section is to place a (formal) assumption on a backdoor attack as corresponding to the strongest feature in a dataset. To accomplish this goal, we first need a way to quantify the "strength" of a given feature ϕ. Intuitively, we would like to call a feature "strong" if adding a single example containing that feature to the training set significantly changes the resulting model. (That is, if the *counterfactual value* of examples containing feature ϕ is high.)

To this end, fix a distribution $\mathcal{D}_S$ over subsets of the training set S. For any feature ϕ and natural number k, let the *k-output* of ϕ be the expected model output (over random draws of the training set) on inputs with feature ϕ, conditioned on having k inputs with feature ϕ in the training set:

Definition 2 (Output function of a feature ϕ)**.** *For a feature*

ϕ, and a distribution $\mathcal{D}_S$ over subsets of the training set S, we define the feature output function g_ϕ as the function that maps any integer k to the expected model output on examples with that feature ϕ when training on exactly k training inputs with that feature ϕ, i.e.,

$$g_\phi(k) = \mathbb{E}_{z\sim\Phi(S)}\left[\mathbb{E}_{S'\sim\mathcal{D}_S}\left[f(z;S')\middle||\Phi(S')|=k, z\notin S'\right]\right] \tag{1}$$

where $z \sim \Phi(S)$ represents a random sample from the support $\Phi(S)$ of the feature ϕ in the set S.

Intuitively, the feature output function $g_\phi(k)$ should grow quickly, as a function of k, for strong features and slowly for weak features. For example, adding an image of a rare dog breed to the training set will rapidly improve accuracy on that dog breed, whereas adding images with a weak feature like "sky" or "grass" will impact the accuracy of the model much less. In the context of backdoor attacks, the "effectiveness" property (Section 2) implies that $g_{\phi_p}(|P|) - g_{\phi_p}(0)$ is *large* where, ϕ_p is the backdoor trigger. Motivated by this observation, we define the *strength* of a feature ϕ as the rate of change of the corresponding output function.

Definition 3 (Strength of a feature ϕ)**.** *We define the k-strength of a feature ϕ as the following function $s_\phi(k)$:*

$$s_\phi(k) = g_\phi(k+1) - g_\phi(k) \tag{2}$$

Note that we can extend Definition 2 and Definition 3 to individual examples too. Specifically, we can define for a feature ϕ the model *k-output* at an example z as:

$$g_\phi(z,k) = \mathbb{E}_{S'\sim\mathcal{D}_S; S'\not\ni z}\left[f(z;S')\middle||\Phi(S')|=k\right].$$

Similarly, we can define the *k-strength* of a feature ϕ at an example z as: $s_\phi(z,k) = g_\phi(z,k+1) - g_\phi(z,k)$.

To provide intuition for Definitions 2 and 3, we instantiate both of them in the context of the trigger feature in a very simple backdoor attack. Specifically, we alter the CIFAR-10 training set, planting a small red square in a random 1% of the training examples, and changing their label to class "0" (so that at inference time, we can add the red square to the input image and make its predicted class be "0").

In this poisoned dataset, the backdoor feature ϕ_p is a detector of the presence of a red square, and the support Φ_p comprises the randomly selected 1% of altered images (i.e., the backdoor images). We train 100,000 models on random 50% fractions of this poisoned dataset, and use them to estimate the k-output and k-strength of the backdoor feature. Specifically, for a variety of examples $z \in S$, we (a) find the models whose training sets had exactly k backdoor images *and* did not contain z; and (b) average the model output on z for each of these models.

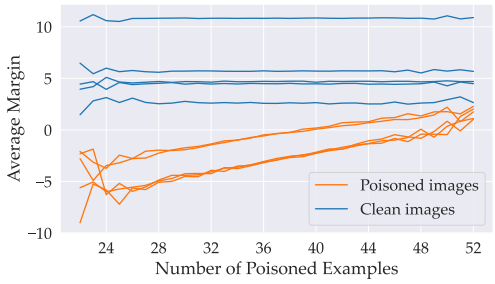


Figure 3: Backdoored CIFAR10 examples. Each orange (resp. blue) line corresponds to a poisoned (resp. clean) example. The x-value represents the number of backdoored examples present in the training set, while the y-value represents the model output (average margin) at that specific example. The rate of change of the model output represents the feature strength $s_{\phi_p}(k)$. We observe that the model output of backdoored images (orange lines) increases as more backdoored examples are included in the training set. In contrast, the model output for clean images (blue lines) is not affected by the number of poisoned training examples.

In Figure 3, we plot the resulting model output for examples $z \in \Phi_p(S)$ that have the feature ϕ_p (orange lines) and also for examples $z \notin \Phi_p(S)$ that do not contain the backdoor feature. Note that by Definition 2, the average k-output of the backdoor feature is the average of the orange lines, and by Definition 3, the average k-strength of the backdoor feature is the average (instantaneous) slope of the orange line. We observe that for the poisoned examples, the k-strength is consistently positive (i.e., the output monotonically increases). This observation motivates our assumption about the strength of backdoor trigger features:

Assumption 1. *Let ϕ_p be the backdoor feature, and $\Phi_p(S)$ be its support (i.e., the backdoored training examples) and let $p := |\Phi_p(S)|$. Then, for some $\delta > 0$, $\alpha \in (0,1)$ and all other features ϕ with $|\Phi(S)| = p$, we assume that*

$$s_{\phi_p}(\alpha \cdot p) \geq \delta + s_\phi(\alpha \cdot p)$$

Justifying the assumption. As we already discussed, Assumption 1 has the advantage of being directly tied to the effectiveness of the backdoor attack. In particular, we know that in the absence of backdoored training examples, the model should do poorly on the inputs with the backdoor trigger (otherwise, we would consider the model to have already been compromised). Thus, $g_{\phi_p}(0)$ is small. On the other hand, for the backdoor attack to be effective, we must have that $g_{\phi_p}(p)$ is large, i.e., models trained on the backdoor training set should perform "well" on backdoored inputs. The mean value theorem[4] thus implies that there must one point $0 \leq k \leq p$ at which $s_{\phi_p}(k)$ is large.

[4]Informally, the mean value theorem says that for any continuous function f and any interval $[a,b]$, there must exist $c \in [a,b]$

4. Detecting Backdoored Examples

The perspective from the previous sections suggests that we need to be able to analyze the strength of features present in a dataset to understand the effect of these features on a model's predictions. Particularly, such an analysis would allow us to translate Assumption 1 into an algorithm for detecting backdoor training examples. Specifically, we would be able to estimate the feature strength $s_\phi(k)$ for a given feature ϕ. If we had a specific feature ϕ in mind, we could compute the feature strength $s_\phi(k)$ using Equations (1) and (2) directly. In our case, however, identifying the feature of interest (i.e., backdoor feature) is essentially our goal.

To this end, in this section we first show how to estimate the strength of all viable features ϕ *simultaneously*. We then demonstrate how we can leverage this estimate to detect the strongest one among them. Our key tool here will be the *datamodeling* framework (Ilyas et al., 2022). In particular, Ilyas et al. (2022) have shown that, for every example z, and for a model output function f corresponding to training a deep neural network and evaluating it on that example, there exists a weight vector $w_z \in \mathbb{R}^{|S|}$ such that: $\mathbb{E}[f(z; S')] \approx \mathbf{1}_{S'}^\top w_z$ for subsets $S' \sim \mathcal{D}_S$, where $\mathbf{1}_{S'} \in \{0,1\}^{|S|}$ is the *indicator vector* of S'[5]. In other words, we can approximate the specific outcome of training a deep neural network on a given subset $S' \subset S$ as a linear function of the presence of each training data example. As the ability of the datamodeling framework to capture the model output function will be critical to our method, we state it as an explicit assumption.

Assumption 2 (Datamodel accuracy)**.** *For any example z, with a corresponding datamodel weight w_z, we have that*

$$\mathbb{E}_{S' \sim \mathcal{D}_S}\left[\left(\mathbb{E}[f(z; S')] - \mathbf{1}_{S'}^\top w_z\right)^2\right] \leq \varepsilon \tag{3}$$

where $\epsilon > 0$ represents a bound on the error of estimating the model output function using datamodels.

Assumption 2 essentially guarantees that datamodels provide an accurate estimate of the model output function for any example z and for any random subset $S' \sim \mathcal{D}_S$. Also, we can in fact verify this assumption by sampling sets S' and computing the error from Assumption 2 directly (replacing the inner expectation with an empirical average).

It turns out that this property alone—captured as a formal lemma below—suffices to estimate the feature strength $s_\phi(k)$ of any feature ϕ.

Lemma 1. *For a feature ϕ, let $\mathbf{1}_{\phi(S)}$ be the indicator vector of its support $\Phi(S)$, $\mathbb{1}_n$ be the n-dimensional vector of ones, and let $h : \mathbb{R}^n \to \mathbb{R}^n$ be defined as*

$$h(v) = \frac{1}{\|v\|_1} v - \frac{1}{n - \|v\|_1}(\mathbb{1}_n - v).$$

Then, under Assumption 2, there exists $C > 0$ such that

$$\left| s_\phi(\alpha \cdot |\Phi(S)|) - \frac{1}{|\Phi(S)|} \sum_{z \in \Phi(S)} w_z^\top h(\mathbf{1}_{\phi(S)}) \right| \leq C\varepsilon^{1/2} n^{1/4}. \tag{4}$$

where ϵ is as defined in Assumption 2.

So, Lemma 1 provides a closed-form expression—involving only the datamodel weight vectors $\{w_z\}$—for the (approximate) feature strength $s_\phi(k)$ of feature ϕ. We provide a proof of this lemma in Appendix B.

4.1. Poisoned examples as a maximum-sum submatrix

In the previous section, we have shown how we can leverage datamodels to estimate any given feature's strength. In this section, we combine Lemma 1 and Assumption 1 (i.e., that the backdoor trigger constitutes the strongest feature in the dataset) into an algorithm that *provably* finds backdoor training examples (provided that Assumptions 1 and 2 hold).

To this end, recall that $n = |S|$ and $p = |\Phi_p(S)|$. Assumption 1 then implies that $s_{\phi_p}(\alpha \cdot p)$ (i.e., the strength of a backdoor feature ϕ_p) is large. So, guided by Lemma 1, we consider the following optimization problem:

$$\arg \max_{v \in \{0,1\}^n} h(v)^\top \mathbf{W} v \qquad \text{s.t.} \qquad \|v_i\|_1 = p, \tag{5}$$

where h is as in Lemma 1. The following lemma (proved in Appendix C) shows that under Assumption 1, the solution to (5) is the indicator vector of the backdoor examples.

Lemma 2. *Suppose Assumption 1 holds for some δ and Lemma 1 for some C. Then if $\delta > 2pC\varepsilon^{1/2}n^{1/4}$, the unique maximizer of (5) is the vector $\mathbf{1}_{\phi_p(S)}$, i.e., the indicator of the backdoored examples, where ϵ is as in Assumption 2.*

Now, the fact that for $v \in \{0,1\}^n$ we have $\mathbb{1}_n^\top \mathbf{W} v = v^\top(\text{diag}(\mathbb{1}_n^\top \mathbf{W}))v$ allows us to express (5) as a submatrix-sum maximization problem. In particular, we have that

$$\begin{aligned} &\operatorname*{argmax}_{v \in \{0,1\}^n : \|v\|_1 = p} \left(\frac{1}{p} \cdot v - \frac{1}{n-p} \cdot (\mathbb{1}_n - v)\right)^\top \mathbf{W} v \\ = &\operatorname*{argmax}_{v \in \{0,1\}^n : \|v\|_1 = p} v^\top \left(\mathbf{W} - \text{diag}\left(\frac{p}{n} \cdot \mathbb{1}_n^\top \mathbf{W}\right)\right) v. \end{aligned}$$

4.2. Detecting backdoored examples

The formulation presented in (5) is difficult to solve directly, for multiple reasons. First, the optimization problem requires knowledge of the number of poisoned examples

such that the rate of change of f at c is equal to $\frac{f(b)-f(a)}{b-a}$.

[5]The indicator vector $\mathbf{1}_{S'}$ takes a value of 1 at index i, if training example $z_i \in S'$, and 0 otherwise.

$|\Phi_p(S)|$, which is unknown in practice. Second, even if we did know the number of poisoned examples, the problem is still NP-hard in general (Branders et al., 2017). In fact, even linearizing (5) and using the commercial-grade mixed-integer linear program solver Gurobi (Gurobi Optimization, LLC, 2021) takes several days to solve (per problem instance) due to the sheer number of optimization variables.

We thus resort to a different approximation. For each k in a pre-defined set of "candidate sizes" K, we set p equal to k (in (5)). We then solve the resulting maximization problem using a greedy local search algorithm inspired by the Kernighan-Lin heuristic for graph partitioning (Kernighan & Lin, 1970). That is, starting from a random assignment for $v \in \{0,1\}^n$, the algorithm considers all pairs of indices $i, j \in [n]$ such that $v_i \neq v_j$, and such that swapping the values of v_i and v_j would improve the submatrix sum objective. The algorithm greedily selects the pair that would most improve the objective and terminates when no such pair exists. We run this local search algorithm $T = 1000$ times for each value of k and collect the *candidate solutions* $\{\boldsymbol{v}^{k,l} : k \in K, l \in [T]\}$.

Rather than using any one of these solutions, we combine them to yield a final score. We define the score of example z_i as the weighted sum of the number of times it was included in the solution of local search, that is,

$$s_i = \sum_{k \in K} \frac{1}{k} \sum_{l=1}^{T} \boldsymbol{v}_i^{k,l}. \tag{6}$$

Intuitively, we expect that backdoored training examples will end up in many of the greedy local search solutions (due to Assumption 1) and thus have a high score s_i. We translate the scores (6) into a concrete defense by flagging (and removing) the examples with the highest score.

5. Experiments

In the previous section, we developed an algorithm that provably detects backdoored examples in a dataset whenever Assumptions 1 and 2 hold. We now consider several settings, and two common types of backdoor attacks: dirty-label attacks (Gu et al., 2017) and clean-label attacks (Turner et al., 2019). For each setting, we verify whether our assumptions hold, and then validate the effectiveness of our proposed detection algorithm.

Experimental setup. In Table 1, we present a summary of our experiments. For all of these experiments, we use the CIFAR-10 dataset (Krizhevsky, 2009), and the ResNet-9 architecture (He et al., 2015), and compute the datamodels using the framework presented in (Ilyas et al., 2022). Specifically, for each experiment and setup, we train a total of 100,000 models, each on a random subset containing 50%[6] of CIFAR-10[7], and chosen uniformly at random.

Exp.	Type	α	Clean Acc.	Backdoor Acc.
1	DL	1.5%	86.64	19.90
2	DL	5%	86.67	12.92
3	DL	1.5%	86.39	49.57
4	DL	5%	86.23	10.67
5	CL	1.5%	86.89	75.58
6	CL	5%	87.11	41.89
7	CL (no adv.)	5%	86.94	71.68
8	CL (no adv.)	10%	87.02	52.08

Table 1: A summary of the different backdoor attacks we consider. "DL" and "CL" stand for dirty- and clean-label attacks respectively, "CL (no adv.)" is the non-adversarial clean label attack from Turner et al. (2019).

5.1. Verifying our assumptions

In Section 3, we presented two assumptions for our proposed defense to (provably) work. We now verify whether these assumptions hold in the experimental settings we consider, then validate the effectiveness of our detection algorithm.

Datamodel accuracy. Lemma 1 states that datamodels are good approximators of a feature strength (provided Assumption 2 holds). To validate whether this is the case, we estimate the "ground-truth" feature strength $s_{\phi_p}(k)$ of the backdoor triggr feature ϕ_p as described in (1) and (2). More precisely, we train 100,000 models on random subsets of the training set, each containing 50% of the training examples. We then compute how the model outputs change with the inclusion/exclusion of the backdoored examples. We then compute the model's outputs for the backdoored examples as a function of the number of backdoored training examples. We then estimate the feature strength $s_\phi(k)$ as the rate of change of the model outputs for backdoored examples.

Afterwards, we estimate the feature strength using datamodels, as given by Equation (4). In particular, we compute the datamodels matrix $\mathbf{W}$, the indicator vector $h\left(\mathbf{1}_{\phi_p(S)}\right)$ from Lemma 1, and their product $h\left(\mathbf{1}_{\phi_p(S)}\right)^\top \mathbf{W} \in \mathbb{R}^{|S|}$ Each entry of this product is an estimate of the backdoor feature strength at every training example. As Figure 4 shows, datamodels are indeed good approximators of the backdoor trigger feature's strength.

Backdoor trigger as the strongest feature. Recall that Assumption 1 states that the backdoor trigger is the strongest among all the features present in the dataset. To validate this assumption in our settings, we leverage our approximation

[6] We train, in Exp. 2 from Table 1, each model on 30% of the dataset. More details in Section 5.1.

[7] The chosen value of α from Assumption 1 is hence $1/2$.

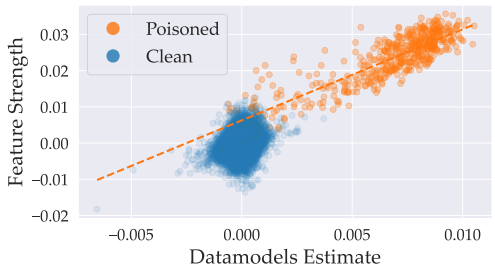


Figure 4: Estimating feature strength using datamodels. Each orange (resp. blue) data point in the scatter plot above represents a poisoned (resp. clean) training example. The x-value of each data point represents the feature strength estimated using datamodels (see Equation (4)), and the y-value represents the feature strength as estimated using Equation (2). We see a strong linear correlation between these two quantities for poisoned examples, which indicates that datamodels provide a good estimate of feature strength.

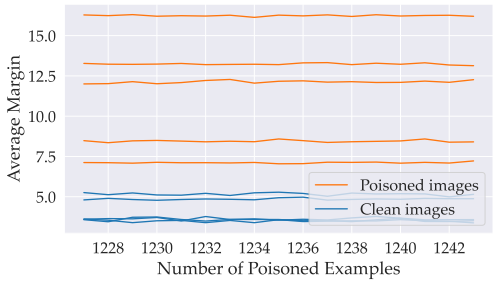


Figure 5: Model output for different number of backdoor training examples. Each orange (resp. blue) line corresponds to a poisoned (resp. clean) example. The x-value represents the number of backdoored examples present in the training set, while the y-value represents the model output (average margin) at that specific example. The rate of change of the model output represents the feature strength. We observe that for backdoored examples (orange lines) from Exp. 2 (see Table 1), the model output does not change as more training examples are poisoned. Consequently, the backdoor feature strength is 0.

of feature strength using datamodels. Specifically, our result from Section 4 suggest;s that the obtained product should be highly correlated with the ground-truth backdoor trigger indicator vector $\mathbf{1}_{\phi_p(S)}$. We thus measure this correlation by computing the area under the ROC curve (AUROC) between the product $h(\mathbf{1}_{\phi_p(S)})^\top \mathbf{W}$ and the indicator vector $\mathbf{1}_{\phi_p(S)}$. As we can see in Table 2, the AUROC score is very high in seven out of the eight settings, which suggests that Assumption 1 indeed holds in these cases.

E1	E2	E3	E4	E5	E6	E7	E8
99.9	60.9	98.0	97.7	99.9	99.9	97.0	98.3

Table 2: AUROC of the backdoor feature strength and the backdoor examples indicator vector for our setups from Table 1.

Interestingly, we observe that we get a low AUROC in the Exp. 2 from Table 1 (the one with a very large number of backdoored examples), which indicates that one of our assumptions does not hold in that case. To investigate the reason, we inspect the backdoor feature strength. Figure 5 shows that, for subsets of the training set containing 50% of the training examples, the model output does not change as the number of poisoned samples increases, i.e., for these subsets, the backdoor feature strength is essentially 0. Consequently, Assumption 1 does not hold. To fix this problem, we use smaller random subsets, i.e., ones containing 30% of the training examples, when estimating the datamodels. In this new setting, the backdoor feature strength is significantly higher, and the AUROC between the poison indicator vector and the feature strenght product jumps to 99.34%. For the remainder of the paper, we will use this setup.

5.2. Evaluating the effectiveness of our defense

Evaluating our score. As a first step, we measure how well our scores predict the backdoored examples in our eight settings. Specifically, we compute our scores by running our local search algorithm from Section 4.2 on the datamodels matrix $\mathbf{W}$, then aggregating the results from the different runs. Following that, we check how well these scores correlate with the backdoor examples indicator vector $\mathbf{1}_{\phi_p(S)}$. As Table 3 shows, there is a high correlation between these two quantities in all setups (cf. Section 5.1). This high correlation suggests that our local search algorithm generates a score that is predictive of the backdoored examples.

E1	E2	E3	E4	E5	E6	E7	E8
94.3	92.25	74.4	80.2	93.4	93.2	91.1	95.5

Table 3: AUROC for our scores (see Section 4.2) and the backdoor indicator vector for our setups from Table 1.

Evaluating the effectiveness of our proposed defense. Given that our scores are predictive of the backdoor examples indicator vector, we expect that removing the examples with the highest scores will be an effective defense against the backdoor attack. To test this claim, for each of the backdoor attacks settings, we train a model on the backdoored dataset, and compute the accuracy of this model on (a) the clean validation set, (b) and on the backdoored validation set[8]. We then remove from the training set the examples cor-

[8]By adding the trigger to all images of the clean validation set.

Exp.	No Defense		AC		ISPL		SPECTRE		SS		Ours	
	Clean	*Poisoned*	*Clean*	*Poisoned*	*Clean*	*Poisoned*	*Clean*	*Poisoned*	*Clean*	*Poisoned*	*Clean*	*Poisoned*
1	86.64	19.90	86.76	19.68	86.13	**86.15**	86.71	20.17	85.52	30.99	85.05	**85.06**
2	86.67	12.92	85.41	12.93	85.88	**85.82**	-	-	85.33	13.63	83.39	**83.13**
3	86.39	49.57	86.25	48.85	86.32	**85.57**	86.28	45.32	85.22	78.22	84.82	**84.11**
4	86.23	10.67	84.75	10.82	85.86	**85.18**	-	-	84.85	13.33	84.64	**83.72**
5	86.89	75.58	86.73	82.83	86.04	**85.89**	86.82	80.65	85.67	**85.41**	83.82	**83.72**
6	87.11	41.89	86.85	51.05	86.18	**86.11**	86.97	51.18	85.68	**85.60**	84.88	**84.79**
7	87.02	71.68	86.90	73.28	86.50	82.31	86.72	76.97	85.70	82.70	84.19	**84.02**
8	86.94	52.08	86.81	56.78	86.04	71.27	86.63	52.27	85.87	71.93	84.81	**84.66**

Table 4: A summary of the model performances on a "clean" and "poisoned" validation sets after applying our method, as well as a number of baselines in all the settings we consider. The high accuracy on both the clean and poisoned validation sets indicates the effectiveness of our defense against the backdoor attacks we consider.

responding to the top 10% of the scores[9], train a new model on the resulting dataset, and then check the performance of this new model on the clean and the fully-backdoored validation sets. We also compare our detection algorithm with several baselines, including Inverse Self-Paced Learning(ISPL) (Jin et al., 2021), Spectral Signatures (SS) (Tran et al., 2018), SPECTRE (Hayase et al., 2021) and Activation Clustering (AC) (Chen et al., 2018). Table 4 shows that there is no substantial drop in accuracy when evaluating the models trained on the curated training set.

6. Related Work

Developing backdoor attacks and defenses in the context of deep learning is a very active area of research (Gu et al., 2017; Tran et al., 2018; Chen et al., 2018; Turner et al., 2019; Saha et al., 2020; Shokri et al., 2020; Hayase et al., 2021; Qi et al., 2022; Goldblum et al., 2022; Goldwasser et al., 2022) (see e.g. (Li et al., 2022) for a survey). One popular approach to defending against backdoor attacks revolves around outlier detection in the latent space of neural networks (Tran et al., 2018; Chen et al., 2018; Hayase et al., 2021). Such defenses inherently fail in defending against adaptive attacks that leave no trace in the latent space of backdoored models (Shokri et al., 2020).

Another line of work investigates certified defenses against backdoor attacks (Levine & Feizi, 2021; Wang et al., 2022). To accomplish that, the proposed methods provide certificates by training separate models on different partitions of the training set, and dropping the models trained on data containing backdoored examples. This approach, however, significantly degrades the accuracy of the trained model, and is only able to certify accuracy when the number of backdoored examples is very small.

A number of prior works explore the applicability of influence-based methods as defenses against different attacks in deep learning (Koh & Liang, 2017). To the best of our knowledge, only Lin et al. (2022) discuss using such methods for defending against backdoor attacks. However, their defense requires knowledge of the attack parameters that are typically unknown. Closest to our work is that of Jin et al. (2021), who consider a defense based on model behavior rather than properties of any latent space.

7. Conclusion

In this paper, we proposed a new perspective on backdoor attacks. Specifically, we argued that backdoor triggers are fundamentally indistinguishable from existing features in the dataset. Consequently, the task of detecting backdoored training examples becomes equivalent to that of detecting strong features. Based on this observation, we propose a primitive—and a corresponding algorithm—for identifying and removing backdoored examples. Through a wide range of backdoor attacks, we demonstrated the effectiveness of our approach.

8. Acknowledgments

Work supported in part by the NSF grants CNS-1815221 and DMS-2134108, and Open Philanthropy. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0015. We thank the MIT Supercloud cluster (Reuther et al., 2018) for providing computational resources that supported part of this work.

[9]We remove 20% in Exp. 2 from Table 1 since the number of poisoned examples is larger.

References

Branders, V., Schaus, P., and Dupont, P. Mining a submatrix of maximal sum. In *International Workshop on New Frontiers in Mining Complex Patterns (NFMCP)*, 2017.

Carlini, N., Jagielski, M., Choquette-Choo, C. A., Paleka, D., Pearce, W., Anderson, H., Terzis, A., Thomas, K., and Tramèr, F. Poisoning web-scale training datasets is practical. In *arXiv preprint arXiv:2302.10149*, 2023.

Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. In *arXiv preprint arXiv:1708.04552*, 2017.

Feldman, V. Does learning require memorization? a short tale about a long tail. In *Symposium on Theory of Computing (STOC)*, 2019.

Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Madry, A., Li, B., and Goldstein, T. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Goldwasser, S., Kim, M. P., Vaikuntanathan, V., and Zamir, O. Planting undetectable backdoors in machine learning models. *arXiv preprint arXiv:2204.06974*, 2022.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011.

Hayase, J., Kong, W., Somani, R., and Oh, S. Spectre: defending against backdoor attacks using robust statistics. *arXiv preprint arXiv:2104.11315*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Huang, K., Li, Y., Wu, B., Qin, Z., and Ren, K. Backdoor defense via decoupling the training process. In *International Conference on Learning Representations (ICLR)*, 2022.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

Jia, J., Cao, X., and Gong, N. Z. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI*, 2021.

Jin, C., Sun, M., and Rinard, M. Provable guarantees against data poisoning using self-expansion and compatibility. 2021.

Kernighan, B. W. and Lin, S. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 1970.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. In *Technical report*, 2009.

Leclerc, G., Salman, H., Ilyas, A., Vemprala, S., Engstrom, L., Vineet, V., Xiao, K., Zhang, P., Santurkar, S., Yang, G., et al. 3db: A framework for debugging computer vision models. In *arXiv preprint arXiv:2106.03805*, 2021.

Leclerc, G., Ilyas, A., Engstrom, L., Park, S. M., Salman, H., and Madry, A. ffcv. https://github.com/libffcv/ffcv/, 2022.

Levine, A. and Feizi, S. Deep partition aggregation: Provable defenses against general poisoning attacks. In *International Conference on Learning Representations*, 2021.

Li, Y., Jiang, Y., Li, Z., and Xia, S.-T. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Lin, J., Zhang, A., Lecuyer, M., Li, J., Panda, A., and Sen, S. Measuring the effect of training data on deep learning predictions via randomized experiments. *arXiv preprint arXiv:2206.10013*, 2022.

Liu, T. Y., Yang, Y., and Mirzasoleiman, B. Friendly noise against adversarial noise: A powerful defense against data poisoning attacks. In *arXiv preprint arXiv:2208.10224*, 2022.

Liu, Y., Lee, W.-C., Tao, G., Ma, S., Aafer, Y., and Zhang, X. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

Lugosi, G. and Mendelson, S. Sub-gaussian estimators of the mean of a random vector. *The annals of statistics*, 47 (2):783–794, 2019.

Qi, X., Xie, T., Mahloujifar, S., and Mittal, P. Circumventing backdoor defenses that are based on latent separability. *arXiv preprint arXiv:2205.13613*, 2022.

Reuther, A., Kepner, J., Byun, C., Samsi, S., Arcand, W., Bestor, D., Bergeron, B., Gadepally, V., Houle, M., Hubbell, M., Jones, M., Klein, A., Milechin, L., Mullen, J., Prout, A., Rosa, A., Yee, C., and Michaleas, P. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6. IEEE, 2018.

Saha, A., Subramanya, A., and Pirsiavash, H. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11957–11965, 2020.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Shokri, R. et al. Bypassing backdoor detection algorithms in deep learning. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 175–183. IEEE, 2020.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Tran, B., Li, J., and Mądry, A. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Turner, A., Tsipras, D., and Madry, A. Label-consistent backdoor attacks. 2019.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proceedings of 40th IEEE Symposium on Security and Privacy*, 2019.

Wang, W., Levine, A., and Feizi, S. Improved certified defenses against data poisoning with (deterministic) finite aggregation. In *International Conference on Machine Learning*, 2022.

Xie, C., Huang, K., Chen, P.-Y., and Li, B. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

Yang, Y., Liu, T. Y., and Mirzasoleiman, B. Not all poisons are created equal: Robust training against data poisoning. In *International Conference on Machine Learning*, 2022.

Zeng, Y., Park, W., Mao, Z. M., and Jia, R. Rethinking the backdoor attacks triggers: A frequency perspective. In *International Conference on Computer Vision (ICCV)*, 2021.

A. Model Predictions Formulation Used in Our Paper

In this work, we use the *margin function* (defined below) as the *model output function* $f(z; S)$.

Definition 4 (Margin function)**.** *For a dataset $S' \subset S$ and a fixed $z = (x, y) \in \mathcal{X} \times \mathcal{Y}$, the* margin function $f(x; S)$ *is defined as*

$$f(z; S') := \textit{the correct-class margin on } z \textit{ of a model trained on } S',$$

where the correct-class margin is the logit of the correct class minus the largest incorrect logit.

Intuitively, $f(z; S')$ maps from an example z and any subset of the training dataset $S' \subset S$ to the correct-class margin on z after training (using any fixed learning algorithm) on S'.

Here we focus on margins because of their (empirically observed) suitability for ordinary least squares, as observed in (Ilyas et al., 2022, Appendix C).

B. Proof of Lemma 1

Lemma 1. *For a feature ϕ, let $\mathbf{1}_{\phi(S)}$ be the indicator vector of its support $\Phi(S)$, $\mathbb{1}_n$ be the n-dimensional vector of ones, and let $h : \mathbb{R}^n \to \mathbb{R}^n$ be defined as*

$$h(v) = \frac{1}{\|v\|_1} v - \frac{1}{n - \|v\|_1}(\mathbb{1}_n - v).$$

Then, under Assumption 2, there exists $C > 0$ such that

$$\left| s_\phi(\alpha \cdot |\Phi(S)|) - \frac{1}{|\Phi(S)|} \sum_{z \in \Phi(S)} w_z^\top h(\mathbf{1}_{\phi(S)}) \right| \leq C\varepsilon^{1/2} n^{1/4}. \tag{4}$$

where ϵ is as defined in Assumption 2.

We decompose the proof into two parts. In the first part, we show that we can approximate the feature strength $s_\phi(k)$ using datamodel weights (Lemma 3). In the second part, we relate h to the expressions involving datamodel weights in Lemma 3 (Lemma 4).

Lemma 3. *Let $\alpha \in (0, 1)$, and S' be a subset of S, sampled uniformly at random, such that $|S'| = \alpha \cdot |S|$. Let $a := \alpha \cdot |\Phi(S)|$. Suppose α is such that $c \leq \alpha \leq 1 - c$ for some absolute constant $c \in (0, 1)$. Then, there exists a constant $C > 0$ (dependending on c) such that we have:*

$$\left| s_\phi(a) - \mathbb{E}_{z \sim \Phi(S)} \left[\mathbb{E}_{S' \sim \mathcal{D}_S} \left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = a + 1 \right] + \mathbb{E}_{S' \sim \mathcal{D}_S} \left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = a \right] \right] \right| \leq C\varepsilon^{1/2} n^{1/4}. \tag{7}$$

Proof. Recall that the feature strength $s_\phi(k)$ is defined as:

$$\begin{aligned} s_\phi(k) :=& g_\phi(k+1) - g_\phi(k) \\ =& \mathbb{E}_{z \sim \Phi(S)} \left[\mathbb{E}_{S' \sim \mathcal{D}_S, S' \not\ni z} \left[f(z; S') \,\middle|\, |\Phi(S')| = k + 1 \right] - \mathbb{E}_{S' \sim \mathcal{D}_S, S' \not\ni z} \left[f(z; S') \,\middle|\, |\Phi(S')| = k \right] \right] \end{aligned} \tag{8}$$

For convenience, assume that $a = \alpha \cdot |\Phi(S)|$ is an integer. First, by triangle inequality, it is enough to show that:

$$\left| \mathbb{E}_{z \sim \Phi(S)} \left[\mathbb{E}_{S' \sim \mathcal{D}_S} \left[f(z; S') \,\middle|\, |\Phi(S')| = a \right] - \mathbb{E}_{S' \sim \mathcal{D}_S} \left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = a \right] \right] \right| \leq \frac{1}{2} \cdot C\varepsilon^{1/2} n^{1/4} \tag{9}$$

and

$$\left| \mathbb{E}_{z \sim \Phi(S)} \left[\mathbb{E}_{S' \sim \mathcal{D}_S} \left[f(z; S') \,\middle|\, |\Phi(S')| = a + 1 \right] - \mathbb{E}_{S' \sim \mathcal{D}_S} \left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = a + 1 \right] \right] \right| \leq \frac{1}{2} \cdot C\varepsilon^{1/2} n^{1/4}. \tag{10}$$

We address Equation (9), and the bound for Equation (10) follows analogously.

To address Equation (9), we will show a stronger (per-example) statement:

$$\left| \mathbb{E}_{S' \sim \mathcal{D}_S} \left[f(z; S') \,\middle|\, |\Phi(S')| = a \right] - \mathbb{E}_{S' \sim \mathcal{D}_S} \left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = a \right] \right| \leq \frac{1}{2} \cdot C\varepsilon^{1/2} n^{1/4} \tag{11}$$

Trivially, this implies that the expectation over z is also bounded from above by $\frac{1}{2} \cdot C\varepsilon^{1/2} n^{1/4}$. To show that Equation (11) holds, we first compute the probability of a random subset S' containing a poisoned samples, and then show an upper bound for Equation (9) leveraging the derived probability by using the definition of conditional expectation.

By directly counting, we have that

$$\mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = a] = \frac{\binom{|\Phi(S)|}{a} \binom{|S| - |\Phi(S)|}{\alpha \cdot |S| - a}}{\binom{|S|}{\alpha \cdot |S|}}.$$

To ease notation, let $n := |S|$ and $p := |\Phi(S)|$, thus $a = \alpha p$. Rewriting, we have

$$\mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p] = \frac{\binom{p}{\alpha p}\binom{n-p}{\alpha(n-p)}}{\binom{n}{\alpha n}}.$$

Next,

$$\begin{aligned}\mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p + 1] &= \frac{\binom{p}{\alpha p+1}\binom{n-p}{\alpha(n-p)-1}}{\binom{n}{\alpha n}} \\ &= \left(\frac{p(1-\alpha)}{\alpha p + 1} \cdot \frac{\alpha(n-p)}{(1-\alpha)(n-p)+1}\right) \cdot \mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p]\end{aligned}$$

We first show that the ratio of the two probabilities is bounded by a constant, i.e.,

$$\begin{aligned}&\frac{p(1-\alpha)}{\alpha p + 1} \cdot \frac{\alpha(n-p)}{(1-\alpha)(n-p)+1} \\ &= \frac{\alpha}{\alpha + \frac{1}{p}} \cdot \frac{1-\alpha}{1-\alpha+\frac{1}{n-p}} \\ &\geq \frac{\alpha}{\alpha+\alpha} \cdot \frac{1-\alpha}{2-\alpha} \geq \frac{1}{2} \cdot \frac{c}{2-c}\end{aligned}$$

where we used that $1 \leq \alpha p$ and $c \leq \alpha \leq 1 - c$. Thus

$$\mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p + 1] \geq \frac{c}{2(2-c)} \mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p].$$

Now we proceed with bounding $\mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p]$. Using Stirling's approximation, we have

$$\mathbb{P}_{S' \sim \mathcal{D}_S}[|\Phi(S')| = \alpha p] \asymp \sqrt{\frac{n}{p(n-p)} \frac{1}{\alpha(1-\alpha)}} \geq \frac{2}{C^2 \cdot \sqrt{n}}$$

for some constant $C > 0$. Now from the triangle inequality, Jensen's inequality and Markov's inequality we have that for sufficiently large n

$$\begin{aligned}&\mathbb{E}_{S' \sim \mathcal{D}_S}\left[f(z; S') \,\middle|\, |\Phi(S')| = \alpha p\right] - \mathbb{E}_{S' \sim \mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha p\right] \\ &\leq \mathbb{E}_{S' \sim \mathcal{D}_S}\left[\left|f(z; S') - w_z^\top \mathbf{1}_{S'}\right| \,\middle|\, |\Phi(S')| = \alpha p\right] \\ &\leq \sqrt{\mathbb{E}_{S' \sim \mathcal{D}_S}\left[\left(f(z; S') - w_z^\top \mathbf{1}_{S'}\right)^2 \,\middle|\, |\Phi(S')| = \alpha p\right]} \\ &\leq \frac{1}{2} C \varepsilon^{1/2} n^{1/4}.\end{aligned}$$

The case for Equation (10) is analogous. □

Next, we show that $w_z^\top h(\mathbf{1}_{\Phi(S)})$ corresponds to the desired difference of conditional expectations. In this proof, we let $h_\phi = h(\mathbf{1}_{\Phi(S)})$ for brevity.

Lemma 4. *We have for every $x \in S$ that*

$$\mathbb{E}_{z \sim \Phi(S)}\left[\mathbb{E}_{S' \sim \mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha \cdot |\Phi(S)| + 1\right] - \mathbb{E}_{S' \sim \mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha \cdot |\Phi(S)|\right]\right] = \mathbb{E}_{z \sim \Phi(S)} w_z^\top h_\phi.$$

Proof. Again, let us consider the case for a single example z, and let $n = |S|, p = |\Phi(S)|$. Then we can write

$$\mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha p\right] = \mathbb{E}_{S'\sim\mathcal{D}_S}\left[\sum_{z\in S} \mathbf{1}_{z\in S'} w_{xz} \,\middle|\, |\Phi(S')| = \alpha p\right]$$
$$= \sum_{z\in S} \mathbb{P}_{S'\sim\mathcal{D}_S}\left[z \in S' \,\middle|\, |\Phi(S')| = \alpha p\right] \cdot w_{xz}.$$

There are a total of

$$\binom{p}{\alpha p}\binom{n-p}{\alpha(n-p)}$$

sets satisfying $|\Phi(S')| = \alpha p$. Among these, given that the sample z contains the feature ϕ, i.e., $\phi(z) = 1$, there are

$$\binom{p-1}{\alpha p-1}\binom{n-p}{\alpha(n-p)}$$

random subsets containing z. So for all z containing ϕ we have

$$\mathbb{P}\left[z \in S' \,\middle|\, \phi(z) = 1, |\Phi(S')| = \alpha p\right] = \frac{\alpha p}{p}.$$

Similarly, for all samples z that do not contain ϕ ,i.e, $\phi(z) = 0$, we have that

$$\mathbb{P}\left[z \in S' \,\middle|\, \phi(z) = 0, |\Phi(S')| = \alpha p\right] = \frac{\alpha(n-p)}{n-p}.$$

Thus, overall

$$\mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha p\right] = \frac{\alpha p}{p} w_z^\top \mathbf{1}_{\phi(S)} + \frac{\alpha(n-p)}{n-p} w_z^\top (1 - \mathbf{1}_{\phi(S)})$$
$$= w_z^\top \left(\frac{\alpha p}{p}\mathbf{1}_{\phi(S)} + \frac{\alpha(n-p)}{n-p}(1 - \mathbf{1}_{\phi(S)})\right)$$
$$= w_z^\top \left(\frac{\alpha p}{p}\mathbf{1}_{\phi(S)} + \frac{\alpha(n-p)}{n-p}(1 - \mathbf{1}_{\phi(S)})\right)$$

Analogously,

$$\mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha p + 1\right] = \frac{\alpha p + 1}{p} w_z^\top \mathbf{1}_{\phi(S)} + \frac{\alpha(n-p) - 1}{n-p} w_z^\top (1 - \mathbf{1}_{\phi(S)})$$
$$= w_z^\top \left(\frac{\alpha p + 1}{p}\mathbf{1}_{\phi(S)} + \frac{\alpha(n-p) - 1}{n-p}(1 - \mathbf{1}_{\phi(S)})\right)$$
$$= w_z^\top \left(\frac{\alpha p + 1}{p}\mathbf{1}_{\phi(S)} + \frac{\alpha(n-p) - 1}{n-p}(1 - \mathbf{1}_{\phi(S)})\right)$$

We then subtract the two terms to get:

$$\mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha p + 1\right] - \mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha p\right] = w_z^\top \left(\frac{1}{p}\mathbf{1}_{\phi(S)} - \frac{1}{n-p}(1 - \mathbf{1}_{\phi(S)})\right)$$
$$= w_z^\top h(\mathbf{1}_{\phi(S)})$$

Finally, we get the desired results over all examples $z \in \Phi(S)$ by directly averaging. □

Proof of Lemma 1. The proof of Lemma 1 follows by combining the results of Lemma 3 and Lemma 4:

$$
\begin{aligned}
&\left|s_\phi(\alpha \cdot |\Phi(S)|) - \mathbb{E}_{z\sim\Phi(S)} w_z^\top h(\mathbf{1}_{\phi(S)})\right| \\
&= \left|s_\phi(\alpha \cdot |\Phi(S)|) - \mathbb{E}_{z\sim\Phi(S)}\left[\mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha\cdot|\Phi(S)|+1\right] + \mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha\cdot|\Phi(S)|\right]\right]\right| \\
&\le \left|\mathbb{E}_{z\sim\Phi(S)}\left[\mathbb{E}_{S'\sim\mathcal{D}_S}\left[f(z;S') \,\middle|\, |\Phi(S')| = \alpha\cdot|\Phi(S)|+1\right] - \mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha\cdot|\Phi(S)|+1\right]\right]\right| \\
&+ \left|\mathbb{E}_{z\sim\Phi(S)}\left[\mathbb{E}_{S'\sim\mathcal{D}_S}\left[f(z;S') \,\middle|\, |\Phi(S')| = \alpha\cdot|\Phi(S)|\right] - \mathbb{E}_{S'\sim\mathcal{D}_S}\left[w_z^\top \mathbf{1}_{S'} \,\middle|\, |\Phi(S')| = \alpha\cdot|\Phi(S)|\right]\right]\right| \\
&\le 2\cdot\frac{1}{2}\cdot C\varepsilon^{1/2}n^{1/4} = C\varepsilon^{1/2}n^{1/4}
\end{aligned}
$$

□

C. Proof of Lemma 2

Lemma 2. *Suppose Assumption 1 holds for some δ and Lemma 1 for some C. Then if $\delta > 2pC\varepsilon^{1/2}n^{1/4}$, the unique maximizer of (5) is the vector $\mathbf{1}_{\phi_p(S)}$, i.e., the indicator of the backdoored examples, where ϵ is as in Assumption 2.*

Proof. The result follows directly from Assumption 1 and Lemma 1. In particular, let ϕ_v be a feature whose corresponding support $\Phi_v(S)$ is of size p.

$$\begin{aligned} h(v)^\top \mathbf{W} v &= \left(\frac{1}{n}\cdot v - \frac{1}{n-p}\cdot(\mathbb{1}_n - v)\right)^\top \mathbf{W} v \\ &= \begin{bmatrix} h_v^\top w_{z_1} & h_v^\top w_{z_2} & \dots & h_v^\top w_{z_n} \end{bmatrix} \cdot v \\ &= \sum_{z\in\Phi_v(S)} h_v^\top w_z. \end{aligned}$$

First, from Lemma 1, we have that:

$$\sum_{z\in\Phi_v(S)} h_v^\top w_z \le \sum_{z\in\Phi_v(S)} s_v(z, \alpha\|v\|_1) + pC^*\varepsilon^{1/2}n^{1/4}$$

Now let v_p be the indicator vector for the poisoned examples. We similarly have from Lemma 1:

$$\sum_{z\in\Phi_p(S)} h_p^\top w_z \ge \sum_{z\in\Phi_p(S)} s_p(z, \alpha\|v\|_1) - pC^*\varepsilon^{1/2}n^{1/4}$$

Thus for any $v \neq v_p$ we have that

$$\begin{aligned} h_p^\top \mathbf{W} v_p - h_v^\top \mathbf{W} v &= \sum_{z\in\Phi_p(S)} h_p^\top w_z - \sum_{z\in\Phi_v(S)} h_v^\top w_z \\ &\ge \sum_{z\in\Phi_p(S)} s_{v_p}(z, \alpha\|v_p\|_1) - \sum_{z\in\Phi_v(S)} s_v(z, \alpha\|v\|_1) - 2pC^*\varepsilon^{1/2}n^{1/4} \end{aligned}$$

We now use Assumption 1 that states:

$$\sum_{z\in\Phi_p(S)} s_{\phi_p}(z, \alpha\cdot p) - \sum_{z\in\Phi(S)} s_\phi(z, \alpha\cdot p) \ge \delta^*$$

By combining these two inequalities, we obtain:

$$\begin{aligned} h_p^\top \mathbf{W} v_p - h_v^\top \mathbf{W} v &= \sum_{z\in\Phi_p(S)} h_p^\top w_z - \sum_{z\in\Phi_v(S)} h_v^\top w_z \\ &\ge \sum_{z\in\Phi_p(S)} s_{v_p}(z, \alpha\|v_p\|_1) - \sum_{z\in\Phi_v(S)} s_v(z, \alpha\|v\|_1) - 2pC^*\varepsilon^{1/2}n^{1/4} \\ &\ge \delta^* - 2pC^*\varepsilon^{1/2}n^{1/4} \end{aligned}$$

This concludes the proof that the solution of the optimization program in Equation (5) is the poison indicator vector v_p, as long as $\delta^* > 2pC^*\varepsilon^{1/2}n^{1/4}$.

□

D. Experimental Setup

Figure 6: We execute the poisoning attacks with three types of triggers: (a) one black pixel on top left corner (first two images), (b) 3x3 black square on top left corner (third and fourth images), and (c) 3-way triggers adapted from (Xie et al., 2020) (last four images).

D.1. Backdoor Attacks

Dirty-Label Backdoor Attacks. The most prominent type of backdoor attacks is a dirty-label attack (Gu et al., 2017). During a dirty-label attack, the adversary inserts a trigger into a subset of the training set, then flips the label of the poisoned samples to a particular target class y_b. We mount four different dirty-label attacks, by considering two different triggers, and two different levels of poisoning (cf. Exp. 1 to 4 in Table 1).

Clean-Label Backdoor Attacks. A more challenging attack is the clean-label attack (Shafahi et al., 2018; Turner et al., 2019)[10] where the adversary avoids changing the label of the poisoned samples. To mount a successful clean-label attack, the adversary poisons samples from the target class only, hoping to create a strong correlation between the target class and the trigger.

We perform two types of clean-label attacks. During the first type (Exp. 5 and 6 from Table 1), we perturb the image with an adversarial example before inserting the trigger, as presented in (Turner et al., 2019). During the second type of clean-label attacks (Exp. 7 and 8 from Table 1), we avoid adding the adversarial example, however, we poison more samples to have an effective attack.

Trigger. We conduct our experiments with two types of triggers. The first type is a fixed pattern inserted in the top left corner of the image. The trigger is unchanged between train and test time. This type of trigger has been used in multiple works (Gu et al., 2017; Turner et al., 2019). The other type of trigger is an m-way trigger, with m=3 (Xie et al., 2020). During training, one of three triggers is chosen at random for each image to be poisoned, and then the trigger is inserted into one of three locations in the image. At test time, all three triggers are inserted at the corresponding positions to reinforce the signal. We display in Figure 6 the triggers used to poison the dataset.

D.2. Training Setup

Training CIFAR models. In this paper, we train a large number of models on different subsets of CIFAR-10 in order to compute the datamodels. To this end, we use the ResNet-9 architecture (He et al., 2015)[11]. This smaller version of ResNets was optimized for fast training.

Training details. We fix the training procedure for all our models. We show the hyperparameter details in Table 5[12]. One augmentation was used for dirty-label attacks (Cutout (DeVries & Taylor, 2017)) to improve the performance of the model on CIFAR10. Similar to (Turner et al., 2019), we do not use any data augmentation when performing clean-label attacks.

Performance. In order to train a large number of models, we use the FFCV library for efficient data-loading (Leclerc et al., 2022). The speedup from using FFCV allows us to train a model to convergence in $\sim$40 seconds, and 100k models for each experiment using 16 V100 in roughly 1 day[13].

[10] We evaluate the clean-label attack as presented in (Turner et al., 2019)
[11] https://github.com/wbaek/torchskeleton/blob/master/bin/dawnbench/cifar10.py
[12] Our implementation and configuration files will be available in our code.
[13] We train 3 models in parallel on every V100.

Optimizer	Epochs	Batch Size	Peak LR	Cyclic LR Peak Epoch	Momentum	Weight Decay
SGD	24	1,024	0	5	0.9	4e-5

Table 5: Hyperparameters used to train ResNet-9 on CIFAR10.

Computing datamodels. We adopt the framework presented in (Ilyas et al., 2022) to compute the datamodels of each experiment. Specifically, we train 100k models on different subsets containing 50% of the training set chosen at random. We then compute the datamodels as described in (Ilyas et al., 2022).

Local Search. We approximate the solution of the problem outlined in (5) using a local search heuristic presented in (Kernighan & Lin, 1970). We iterate over ten sizes for the poison mask: {10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120}. For each size, we collect 1,000 different solutions by starting from different initialization of the solution.

D.3. Estimating Theoretical Quantities

Recall the average margin definition presented in (1). In particular:

$$g_\phi(k) = \mathbb{E}_{z\sim\Phi(S)}\left[\mathbb{E}_{S'\sim\mathcal{D}_S}\left[f(z;S')\middle||\Phi(S')| = k, z\notin S'\right]\right] \tag{12}$$

where S' is a subset of the training set, $f(z;S')$ is the margin of the model on example z when trained on the dataset S', $\Phi(S')$ is the subset of the set S' containing the poisoned feature, and k is the number of poisoned samples. Estimating the average margins requires training a large number of models on different subsets, and measure–for every sample z and every number of poisoned samples k–the margins of the trained model.

For the purpose of this paper, we leverage the datamodels computation framework to estimate these averages. In particular, to compute the datamodels weights, we train a large number of models on different subsets $S_1, S_2, \ldots, S_T$ of the training set S[14]. For every subset S_i, we record the number of poisoned samples in the subset, then we estimate the average margin by averaging the margins over the different subsets that contain k poisoned samples.

$$N_\phi(z,k) = \sum_{i=1}^{T} \mathbf{1}_{(|\Phi(S_i)|=k)\,\wedge\,(z\notin S_i)} \tag{13a}$$

$$\mathbb{E}_{S'\sim\mathcal{D}_S}\left[f(z;S')\middle||\Phi(S')| = k, z\notin S'\right] \approx \frac{1}{N_\phi(k)}\sum_{i=1}^{T} f(z;S_i)\cdot\mathbf{1}_{(|\Phi(S_i)|=k)\,\wedge\,(z\notin S_i)} \tag{13b}$$

$$g_\phi(k) \approx \frac{1}{|\Phi(S)|}\sum_{z\in\Phi(S)}\frac{1}{N_\phi(z,k)}\sum_{i=1}^{T} f(z;S_i)\cdot\mathbf{1}_{(|\Phi(S_i)|=k)\,\wedge\,(z\notin S_i)} \tag{13c}$$

By a training 100k models on different subsets of the dataset, we obtain reasonable estimates of the marginal effects for every sample $z = (x, y)$.

[14] We refer the reader to (Ilyas et al., 2022) for more details.

E. Omitted Plots

E.1. Average Margin Plots

In this section, we show for all the experiments the plots of the average margin for clean and poisoned samples as a function of the number of poisoned samples in the dataset (cf. Fig. 3 in the main paper).

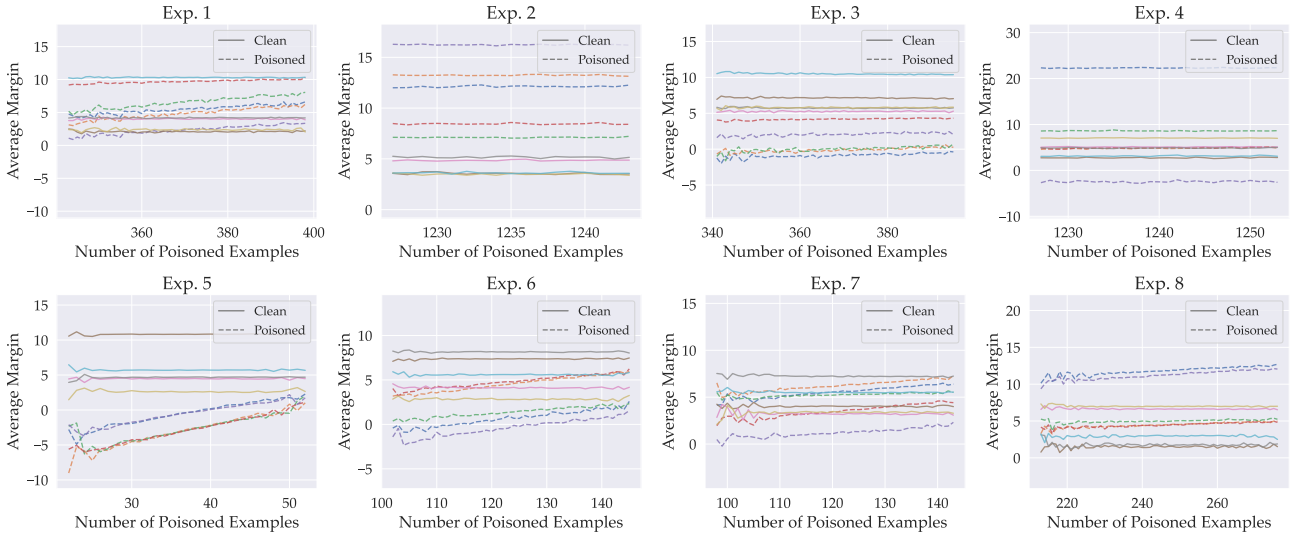


Figure 7: We plot for all the experiments the average margin for five clean samples (left) and five poisoned samples (right) as the number of poisoned samples in the training set increases. We observe that the average margin for *clean samples* (without the trigger) is constant when poisoning more samples in the dataset. In contrast, the average margin for *poisoned samples* (with the trigger) increases when the number of poisoned samples increases in the dataset, confirming our assumptions.

E.2. Estimated Backdoor Feature Strength Plots

In this section, we show for all the experiments the plots of the estimated backdoor feature strength, and the approximation we obtain using datamodels (cf. Figure 4 in the main paper).

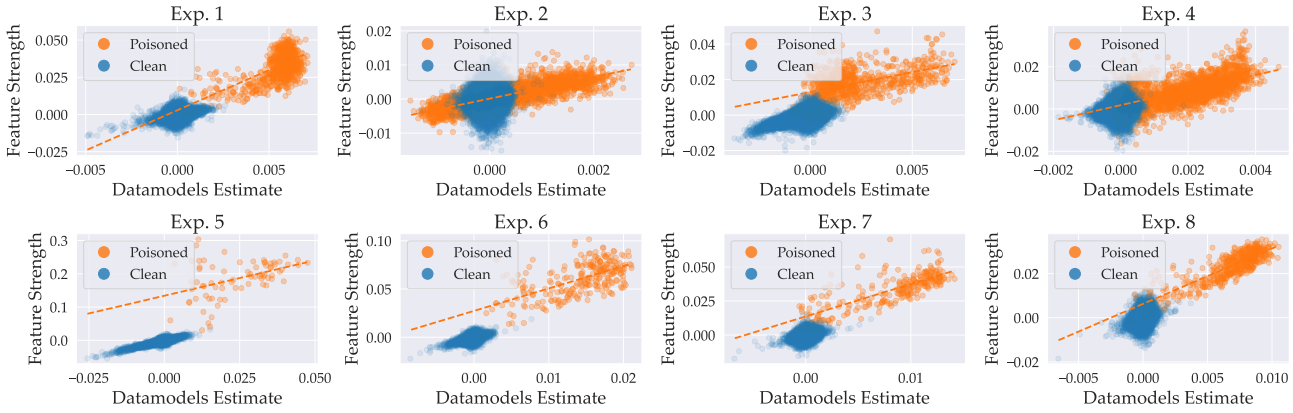


Figure 8: We plot for all the experiments the estimated backdoor feature strength and the approximation with datamodels presented in Equation 4. We observe for poisoned samples (in red) a good linear correlation between the strengths and the datamodels' approximation. Additionally, we observe no noticeable correlation for clean samples (in green).

E.3. Distribution of Datamodels Values

In this section, we plot for each experiment the distribution of the datamodels weights for all experiments. In particular, recall that the datamodels weight $w_z[i]$ represents the influence of the training sample z_i on the sample z. We show below the distribution of the effect of 1) poisoned samples on poisoned samples, 2) the poisoned samples on the clean samples and 4) the clean samples on the clean samples.

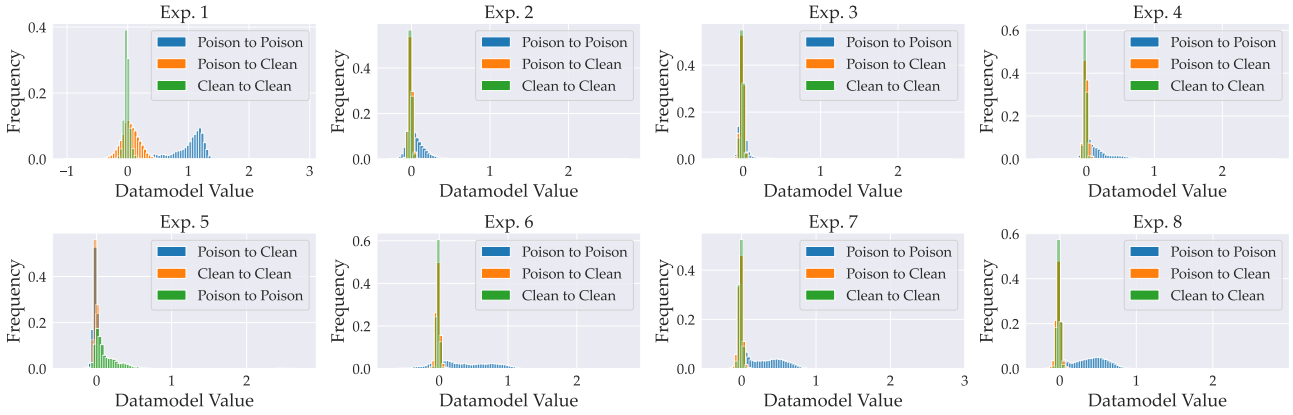


Figure 9: We plot the distribution of the datamodels weights for all the experiments. We clearly see that the effect of poisoned samples on other poisoned samples is generally higher than the effect of poisoned samples on clean samples, and than clean samples on each other.

E.4. Attack Success Rate (ASR)

In the main paper, we presented our results by measuring the accuracy of a model on a clean and a poisoned validation sets. Another relevant metric is the Attack Success Rate (ASR) which measures the probability that the model predicts the target class after inserting the trigger into an image. As we can see in Table 6, our defense leads to a low ASR in seven out of eight setups.

Table 6: We compare our method against multiple baselines in a wide range of experiments. We observe that our algorithm leads to a low ASR in all of our settings. Refer to Table 1 for the full experiments parameters.

Exp.	No Defense	AC	ISPL	SPECTRE	SS	Ours
1	87.94	88.26	**0.70**	87.67	73.78	**0.81**
2	96.38	96.32	**0.67**	-	95.40	**1.44**
3	50.49	51.33	**0.58**	55.68	10.44	**1.18**
4	99.21	99.02	**0.75**	-	95.85	**2.30**
5	15.66	5.35	**0.71**	7.66	**0.80**	**0.92**
6	58.57	45.44	**0.66**	46.78	**0.67**	**0.77**
7	26.09	23.64	9.90	18.48	9.17	**3.56**
8	50.62	44.82	26.07	44.14	24.72	**3.42**